ITSE | EDUCACIÓN SUPERIOR DE CICLO CORTO

Innovación Digital

Ciberseguridad

Análisis de vulnerabilidades

Proyecto final: Ubuntu server

Carolina Guevara

Instructor: Jorge Macías

UBUNTU SERVER

Análisis de vulnerabilidades

Carolina Guevara

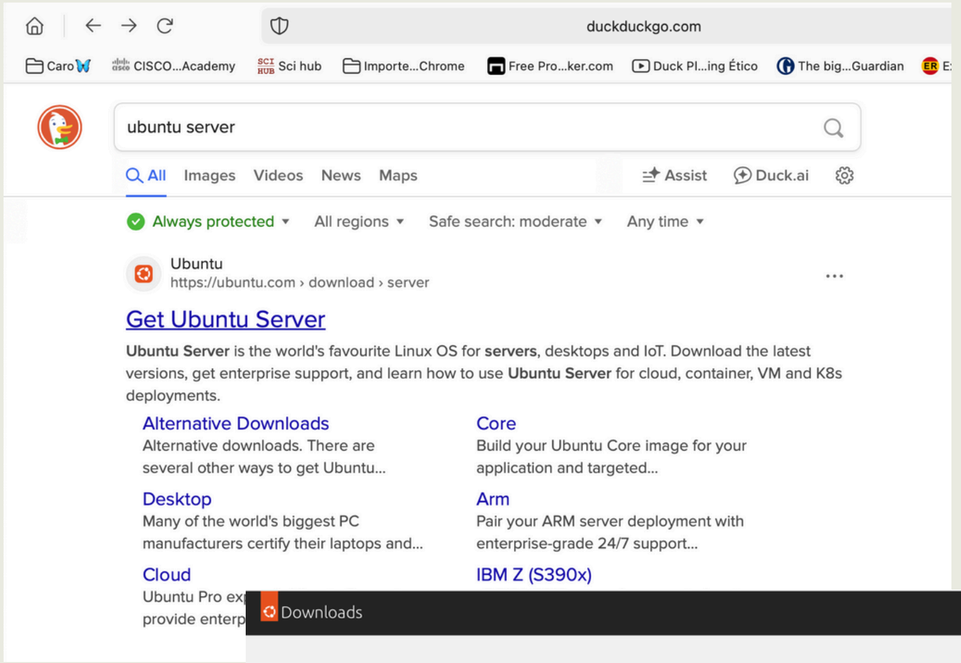
“Hazlo o no lo hagas, pero no lo intentes”.

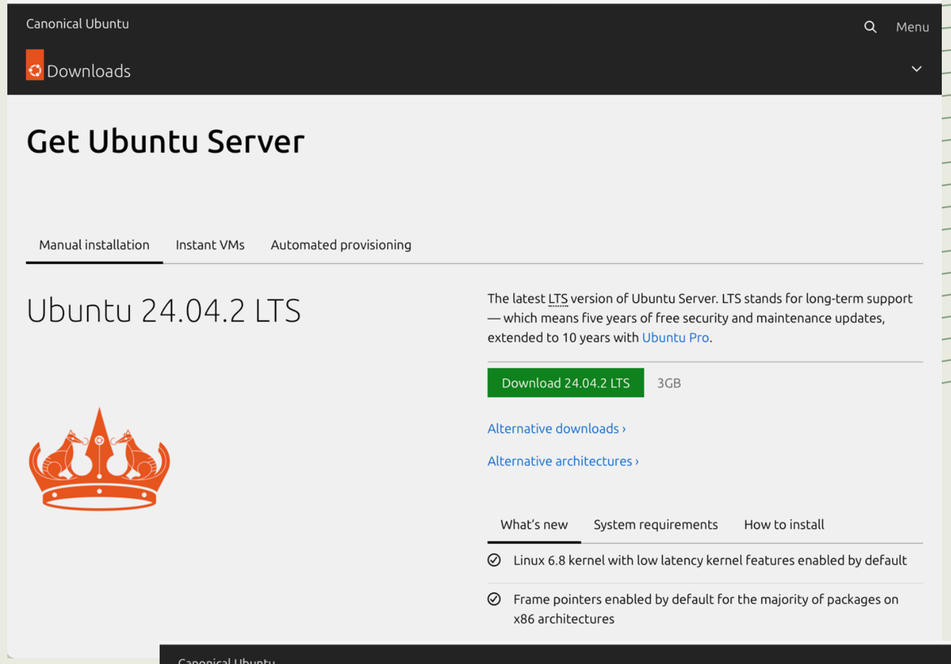


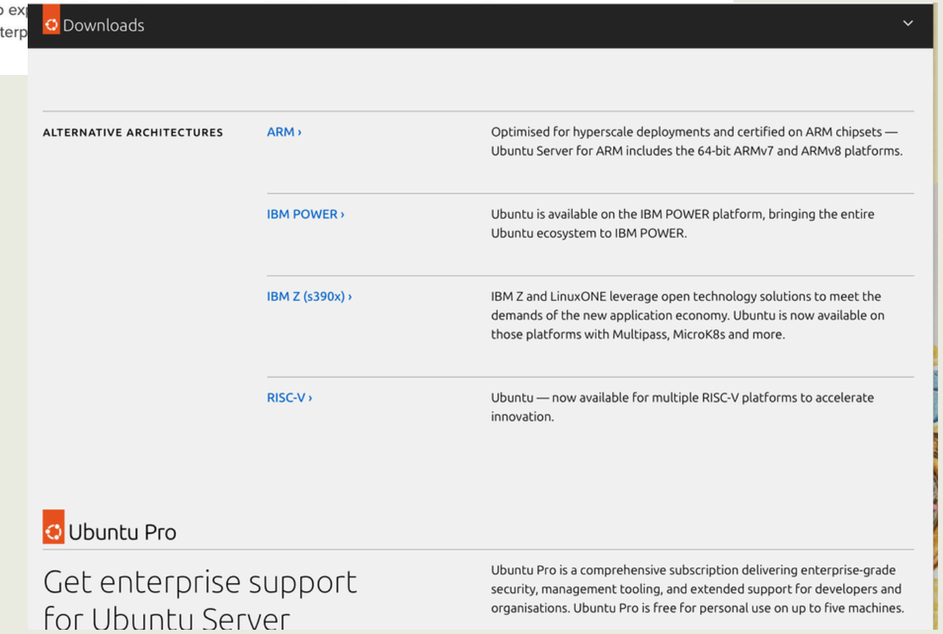
Índice

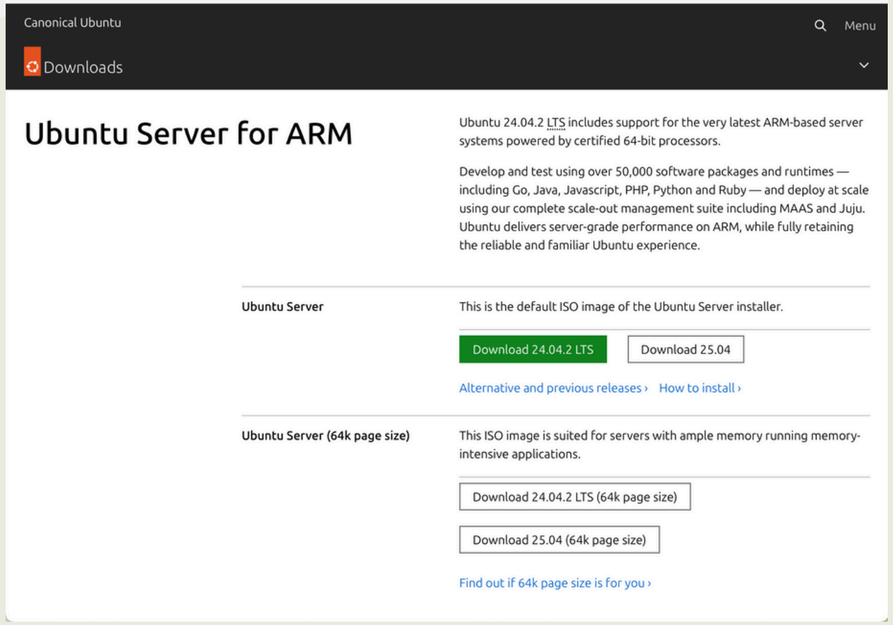
- Introducción
- **Parte I:** descarga de ISO, descarga de server en software
- **Parte II:** config, SFTP y config. servidor web
- **Parte III:** Análisis de vulnerabilidades: Zenmap y Nmap
- **Parte IV:** mejores prácticas de seguridad
- **Parte V:** Escaneo de seguimiento
- Conclusión

Descarga de Ubuntu server

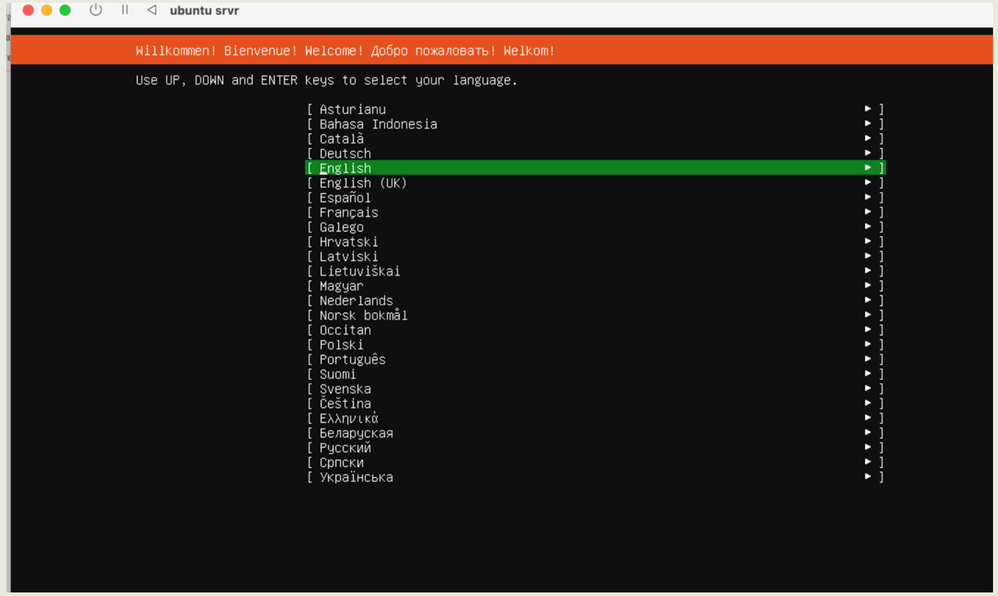






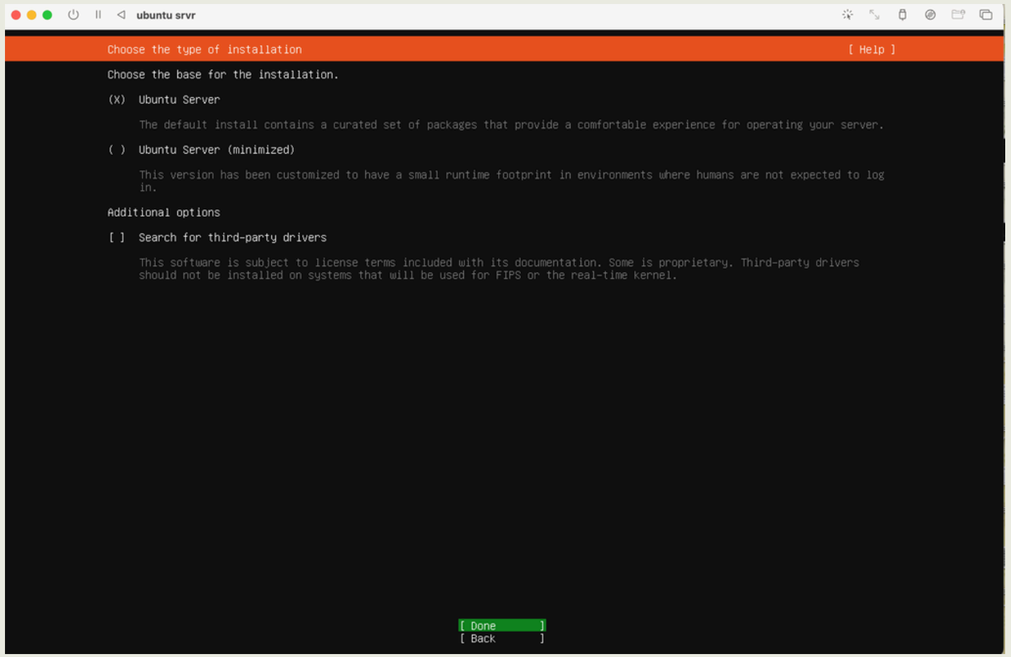


Descarga de Ubuntu server

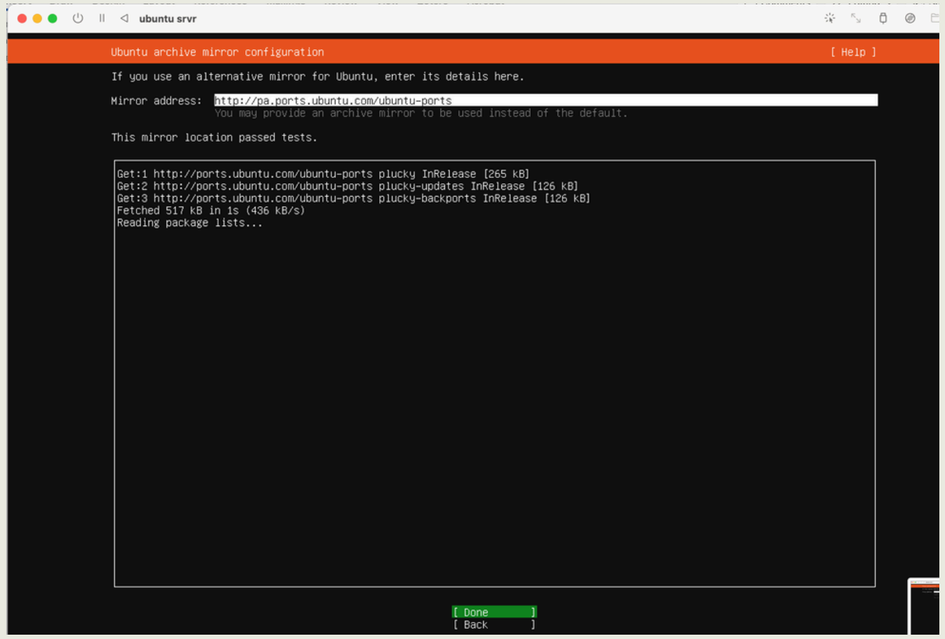


Para iniciar con la configuración del server debemos escoger el idioma, en mi caso escogí inglés ya que la documentación está en ese idioma.

Luego procedemos a escoger el tipo de instalación, como se necesita server escogemos el apartado que dice Ubuntu server.

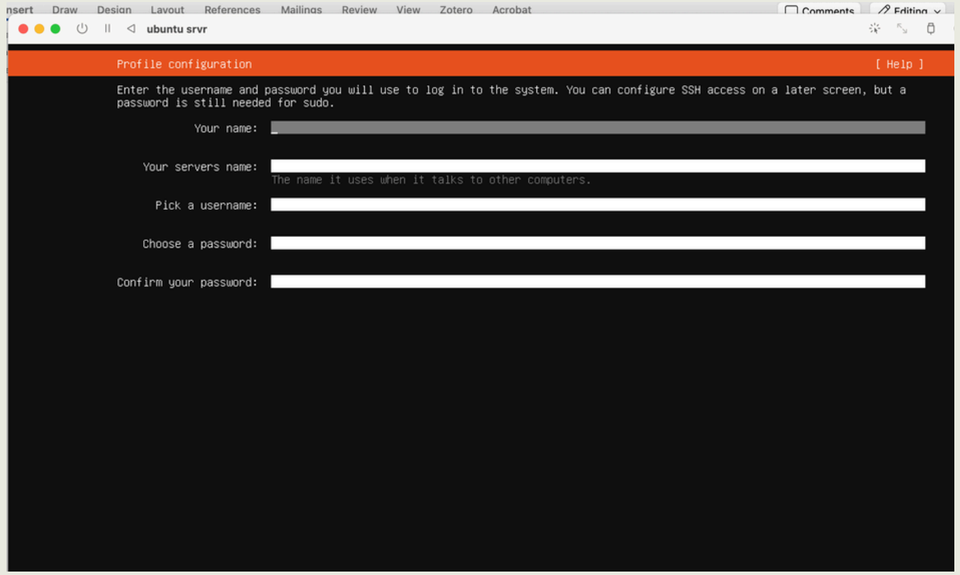


Descarga de Ubuntu server

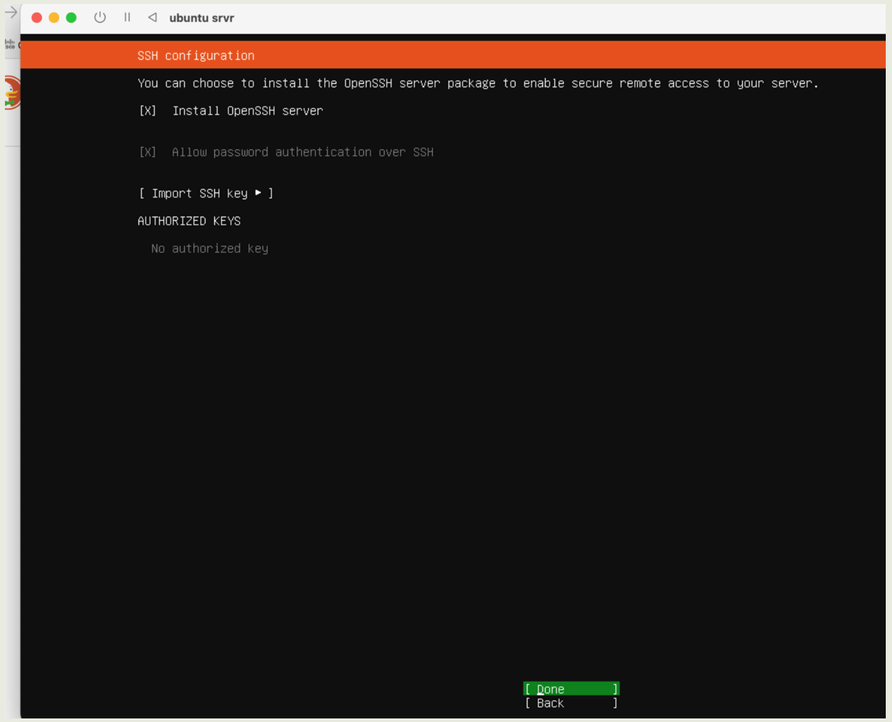


Paso importante:seguimos con la configuración y dejamos que se carguen los archivos mirror y seguimos en done,

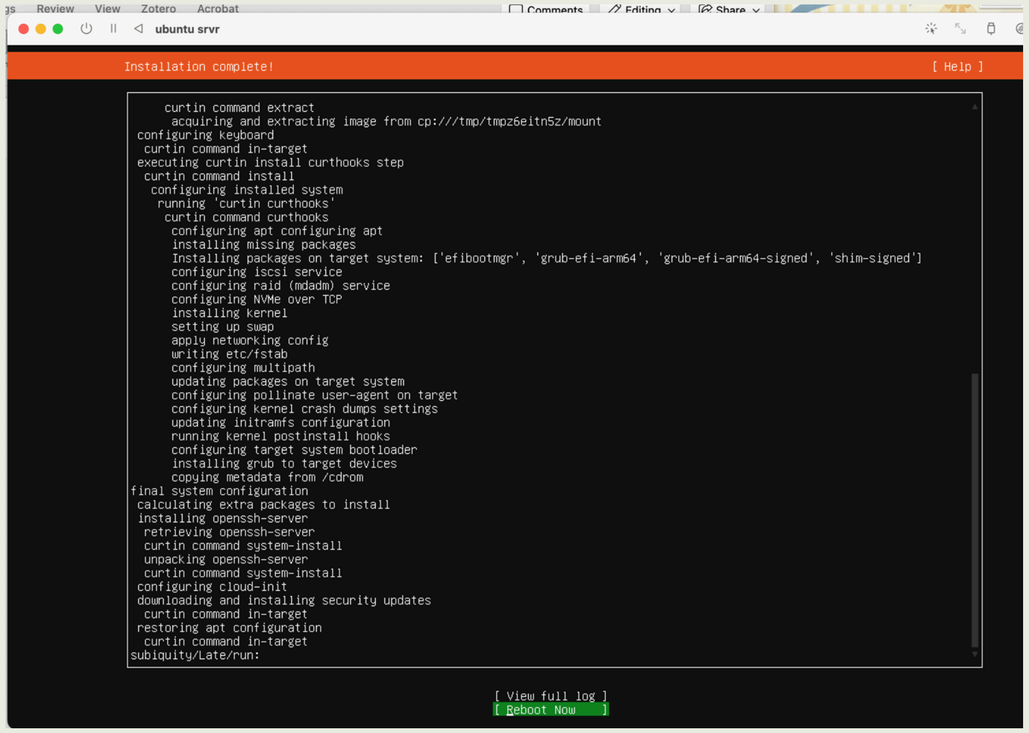
Escogemos nuestro usuario, cómo se llamará nuestro servidor y la contraseña del mismo.



Descarga de Ubuntu server



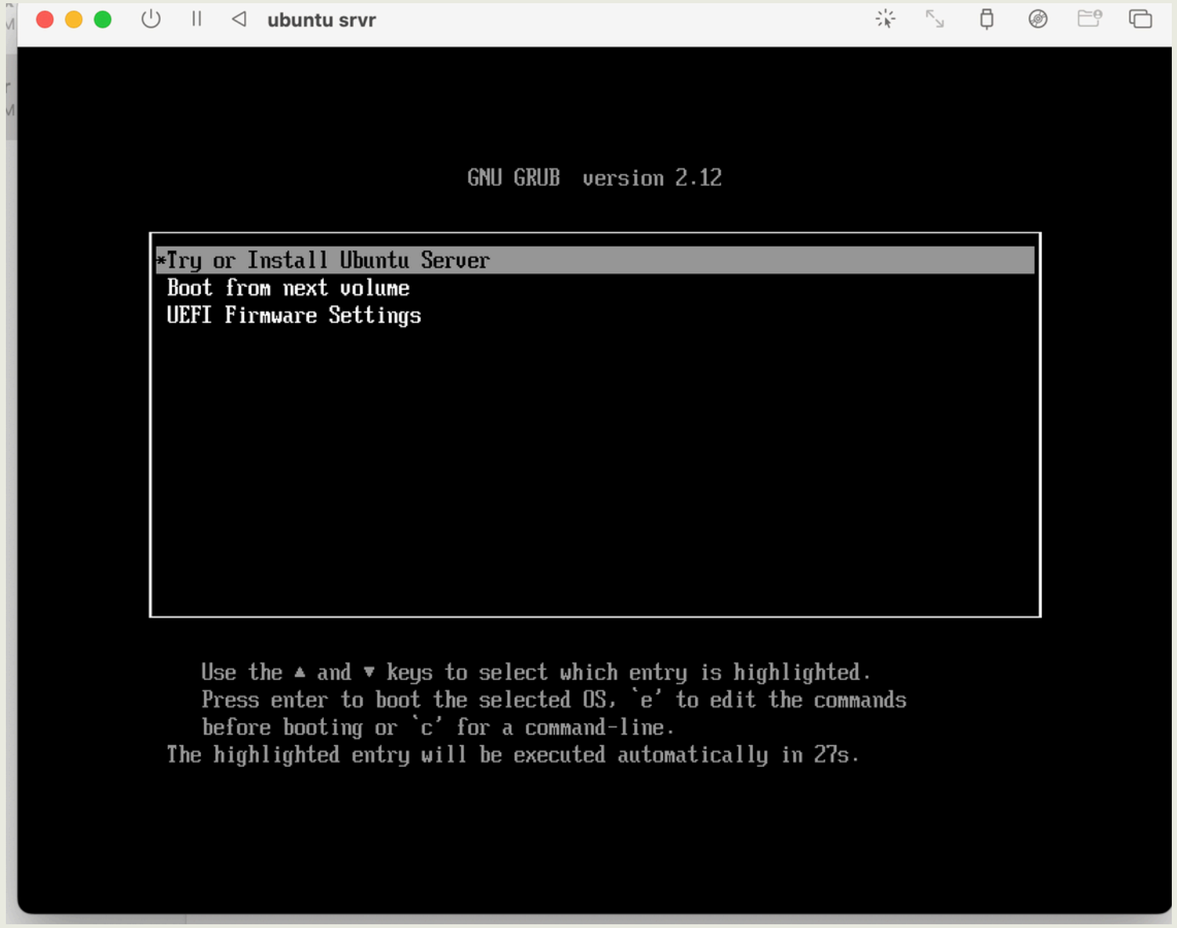
Luego escogemos el apartado de Install OpenSSH server para que al momento de que se realicen las descargas ya tengamos esa aplicaicón en nuestro servidor.



Finalizamos la descarga y configuración del servidor en nuestro software de virtualización, sabemos que la descarga fue correcta cuando vemos la siguiente línea:

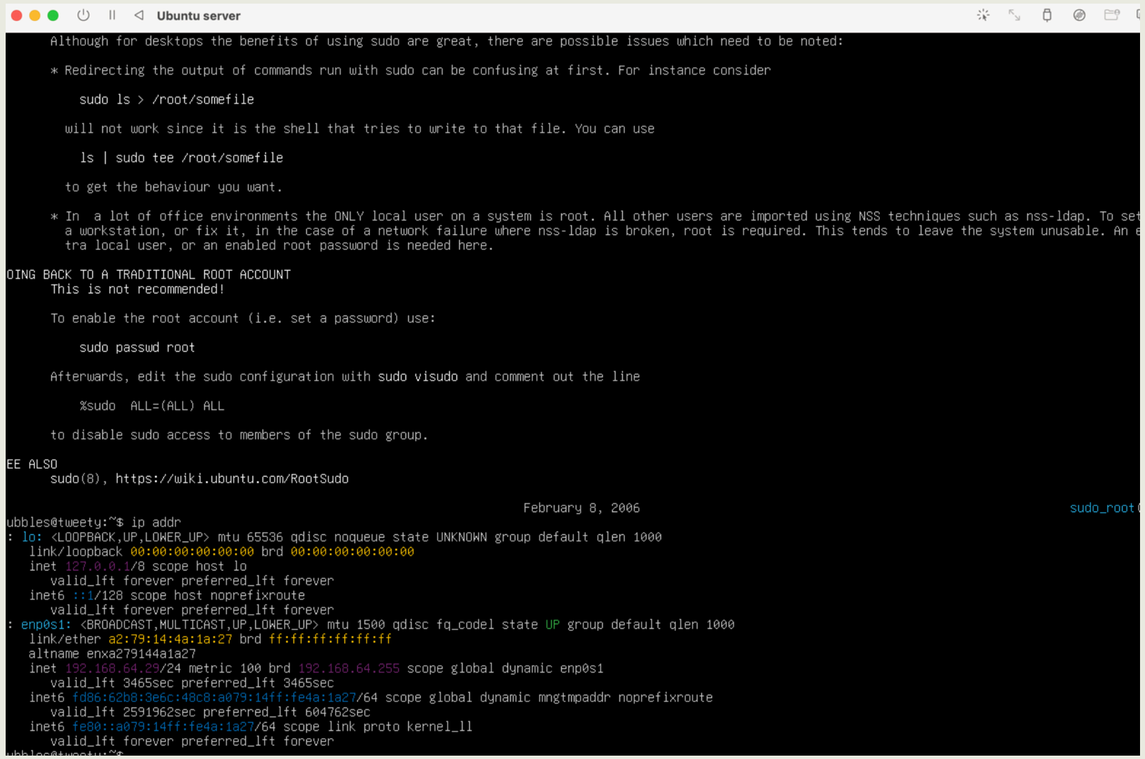
subiquity/late/run:

Descarga de Ubuntu server



Al realizar el reboot del server nos aparece este cuadro de diálogo, seleccionamos **boot from next volume** y ya estamos listos para empezar la configuración de nuestro servidor.

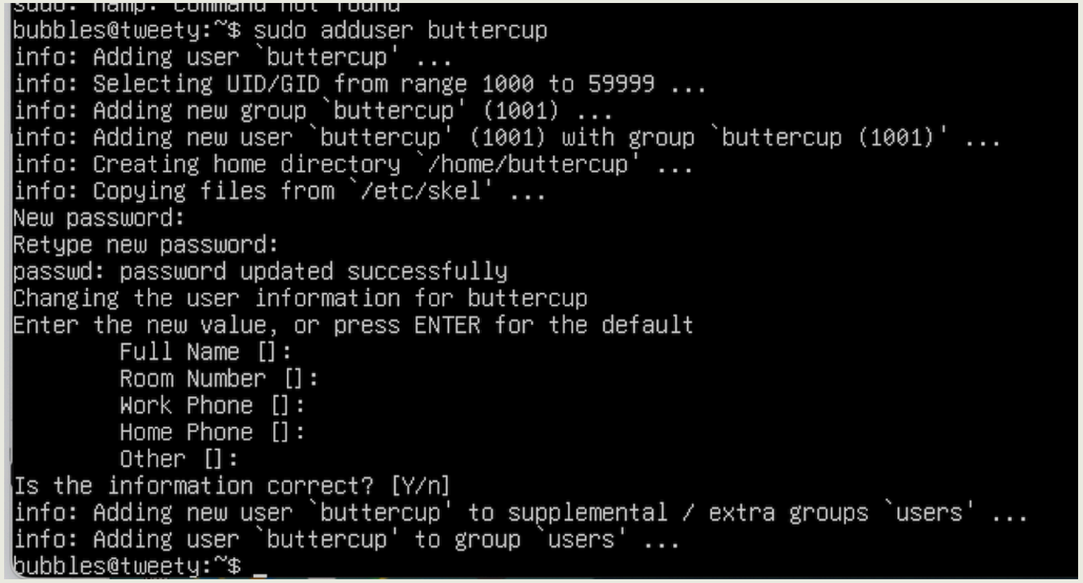
Descarga de Ubuntu server



Al iniciar el servidor descubrí la IP, la cual es:

192.168.64.29

Luego procedí a crear un usuario para tareas de rutina, el cual le coloqué el nombre de buttercup y lo añadí al grupo de users2



PARTE II

Configuración SFTP

Configuración del servidor Web

Configuración SFTP

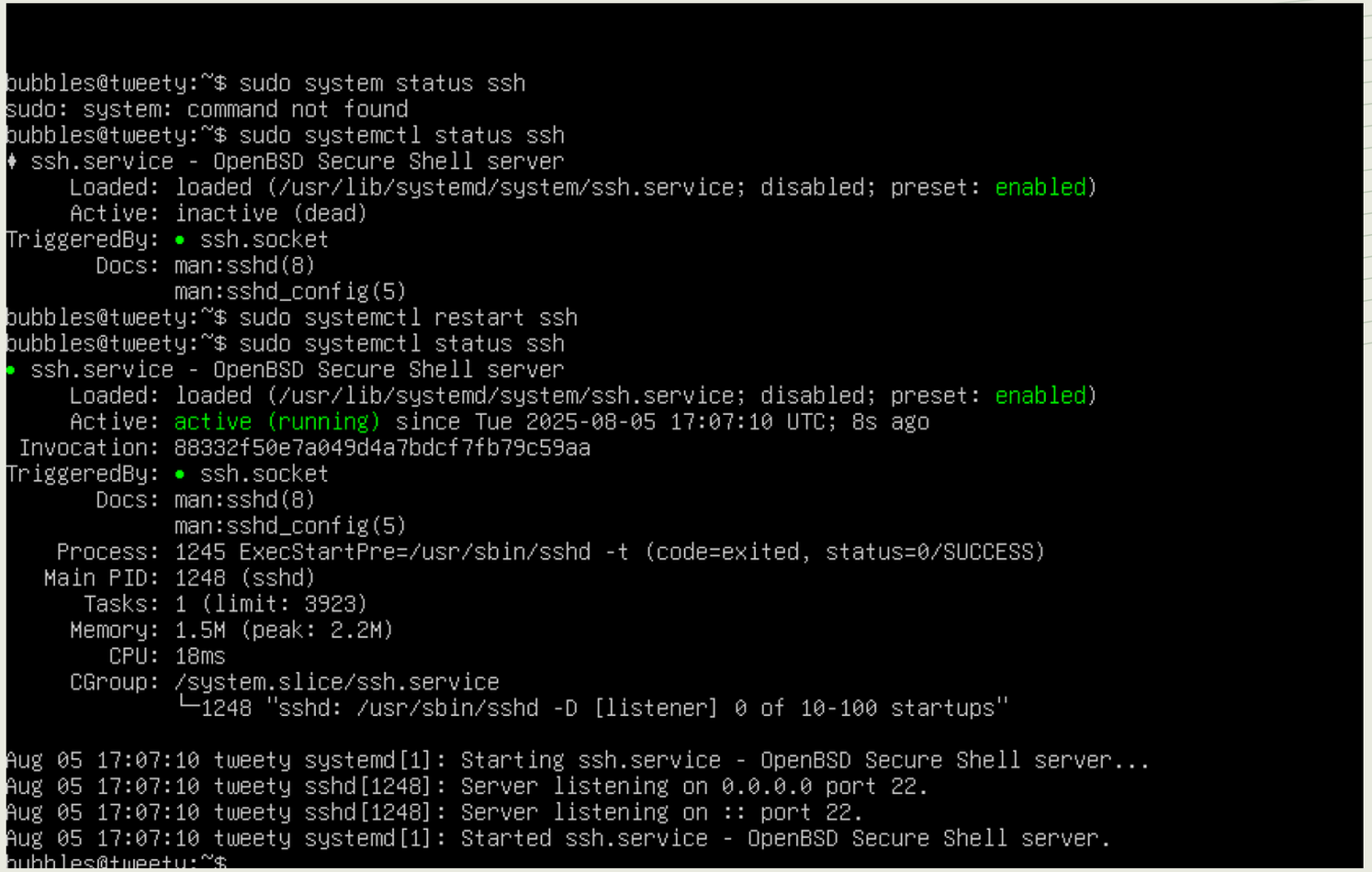
Se editó el archivo de configuración SSH (normalmente ubicado en '/etc/ssh/sshd_config') para habilitar SFTP.





Al acceder al archivo nano debemos ubicar la línea **# overrride default of no subsystems** y revisar que el comentario **Subsystem sftp /usr/lib/openssh/sftp-server** esté habilitado es decir sin #. Luego para restringir el acceso SFTP a un grupo específico de usuarios, puedes agregar las siguientes líneas al final del archivo:

Configuración SFTP

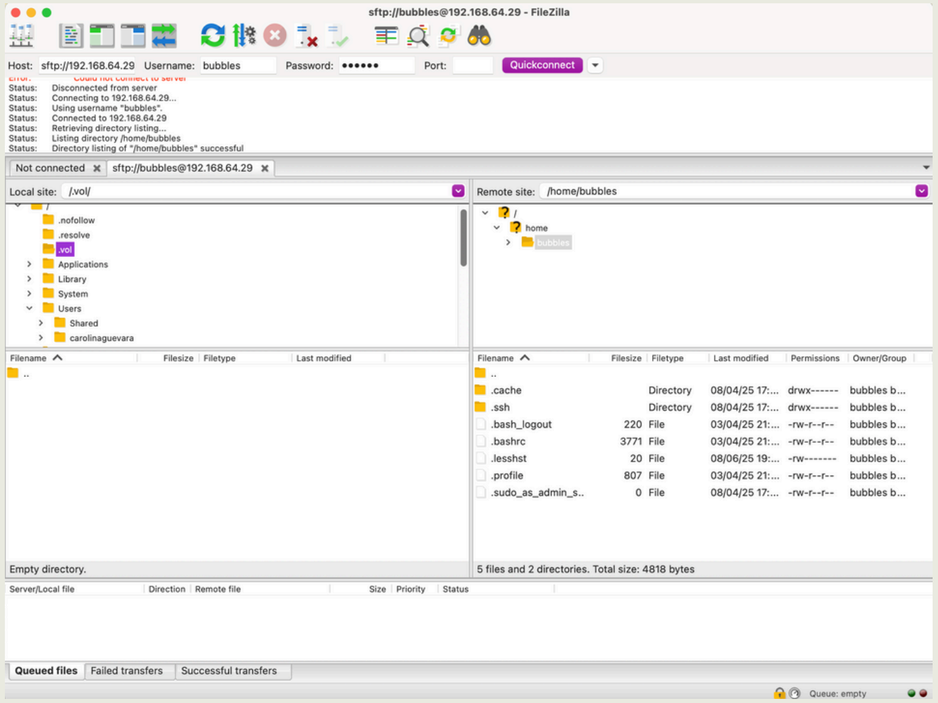


Procedí a confirmar que el servicio de SSH estuviera enable y activo para poder realizar las comunicaciones con otras aplicaciones.

Pasos adicionales

Se puede realizar el reinicio del servicio SSH, esto para que se apliquen los cambios.

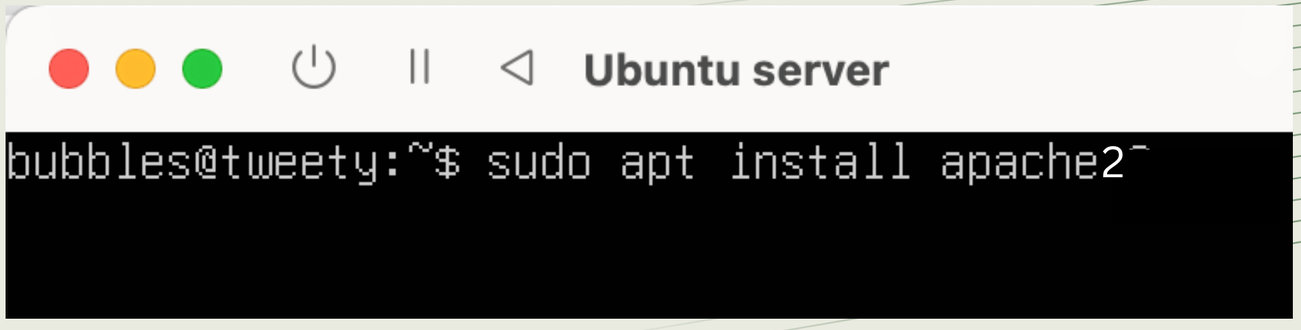
```
bubbles@tweety:~$ sudo systemctl restart ssh
bubbles@tweety:~$ _
```



Probé la la conexion SFTP con FileZilla

Configuración de servidor web

Procedí a utilizar los comandos **sudo apt install apache2** para para servir contenido web a través de Internet.




```
apache2 is already the newest version (2.4.63-1ubuntu1.1).
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 35
bubbles@tweety:~$
```

Al realizar la descarga del apache2 debe aparecerte los documentos de la BD, en mi caso ya lo había instalado.

Configuración de servidor web

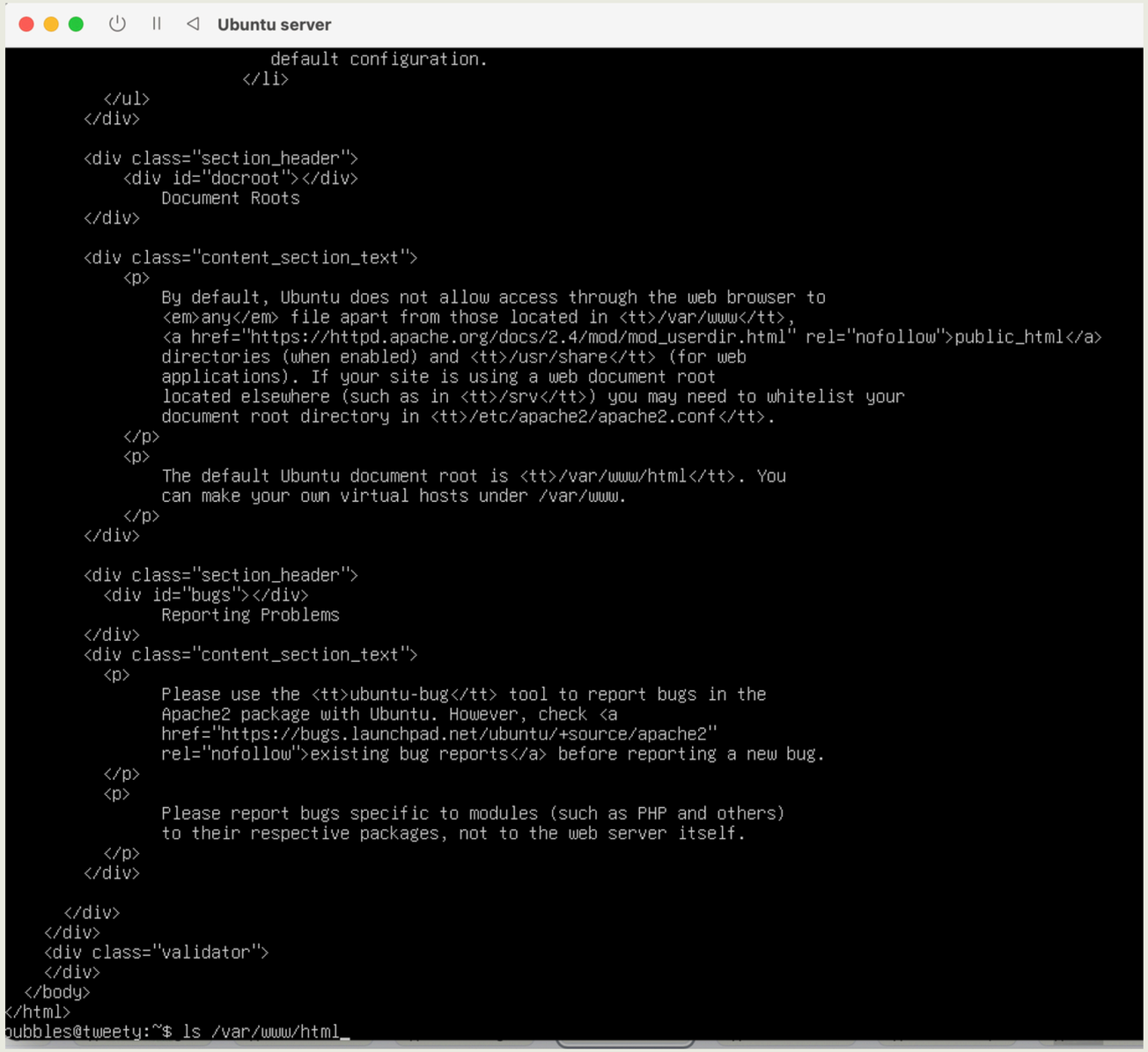
Creación de documento para la página web con los permisos necesarios.

```
bubbles@tweety:~$ sudo mkdir -p /var/www/html
[sudo] password for bubbles:
bubbles@tweety:~$ sudo chown -R www-data:www-data /var/www/html
bubbles@tweety:~$ sudo chmod -R 755 /var/www
bubbles@tweety:~$
```

```
bubbles@tweety:~$ ls /var/www/html
index.html
bubbles@tweety:~$ sudo chown -R www-data:www-data /var/www/html
[sudo] password for bubbles:
Sorry, try again.
[sudo] password for bubbles:
bubbles@tweety:~$ sudo chmod -R 755 /var/www
bubbles@tweety:~$
```

Página web creada con el siguiente comando:

Configuración de servidor web



Al crear la página web dentro del servidor se hace visible la información dentro de la web creada previamente, la misma se le colocó el nombre de **index.html y se localiza en ls/var/www/html.**

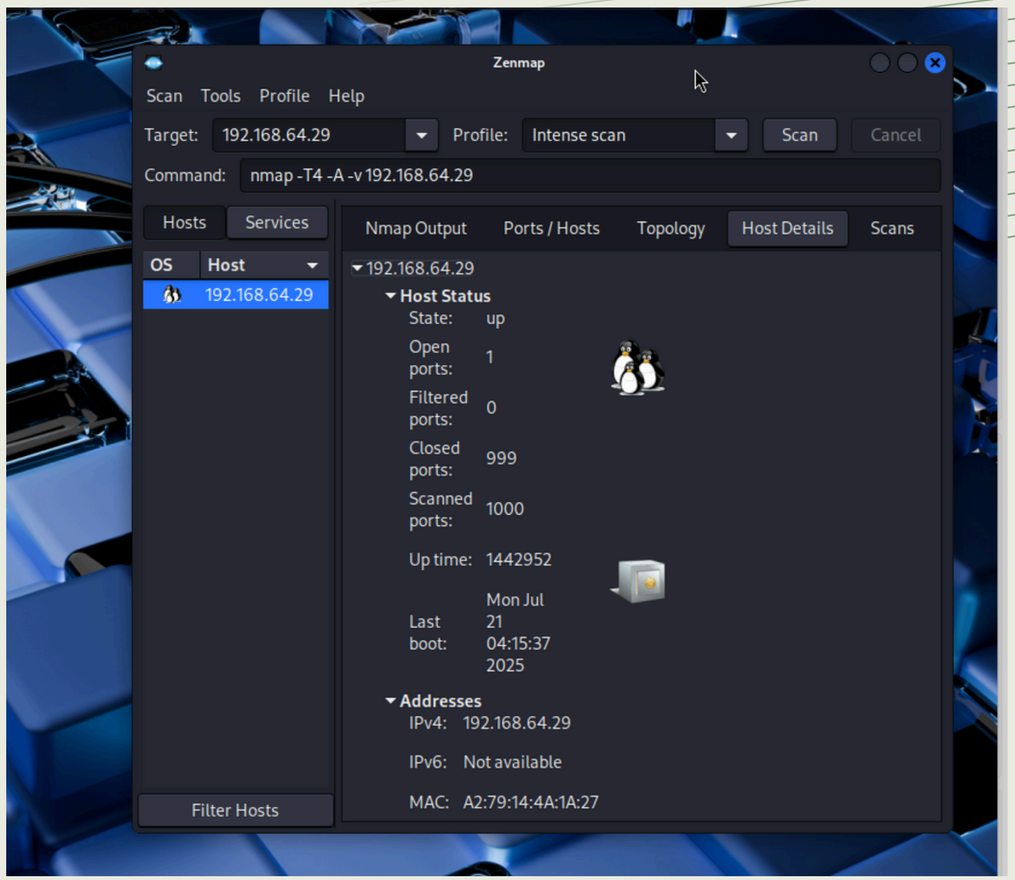
PARTE III
Análisis de vulnerabilidades

Herramientas de análisis:
Zenmap
Nmap

Escaneos iniciales

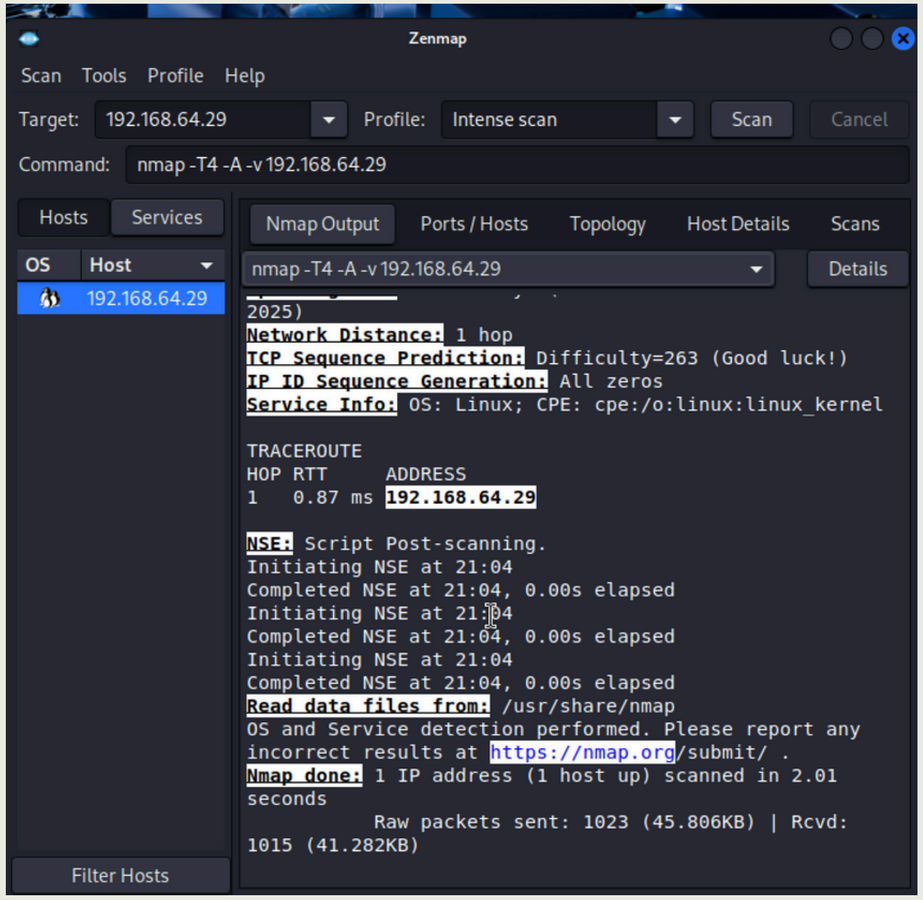
Herramientas de análisis: Zenmap

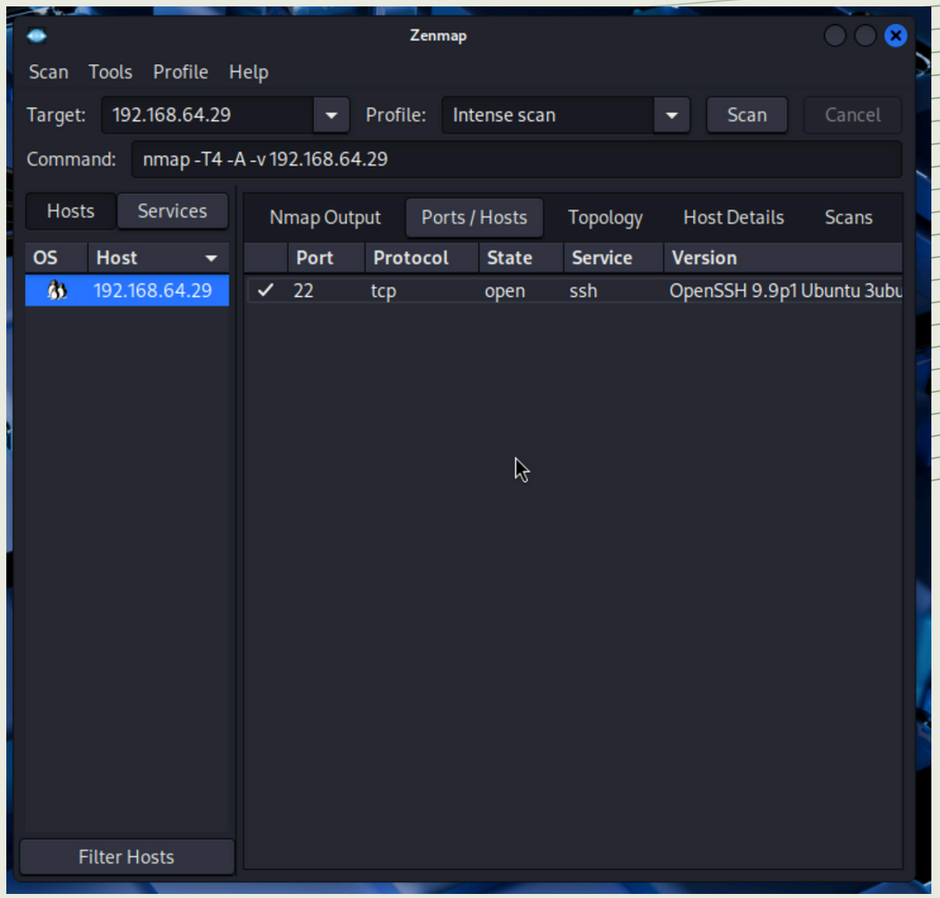
Realicé un descubrimiento del host **(IP del servidor de Ubuntu)** utilizando la herramienta Zenmap que está dentro de las aplicaciones del Sistema Operativo Kali Linux



Herramientas de análisis: Zenmap

Realicé un descubrimiento del host **(IP del servidor de Ubuntu)** utilizando la herramienta Zenmap que está dentro de las aplicaciones del Sistema Operativo Kali Linux

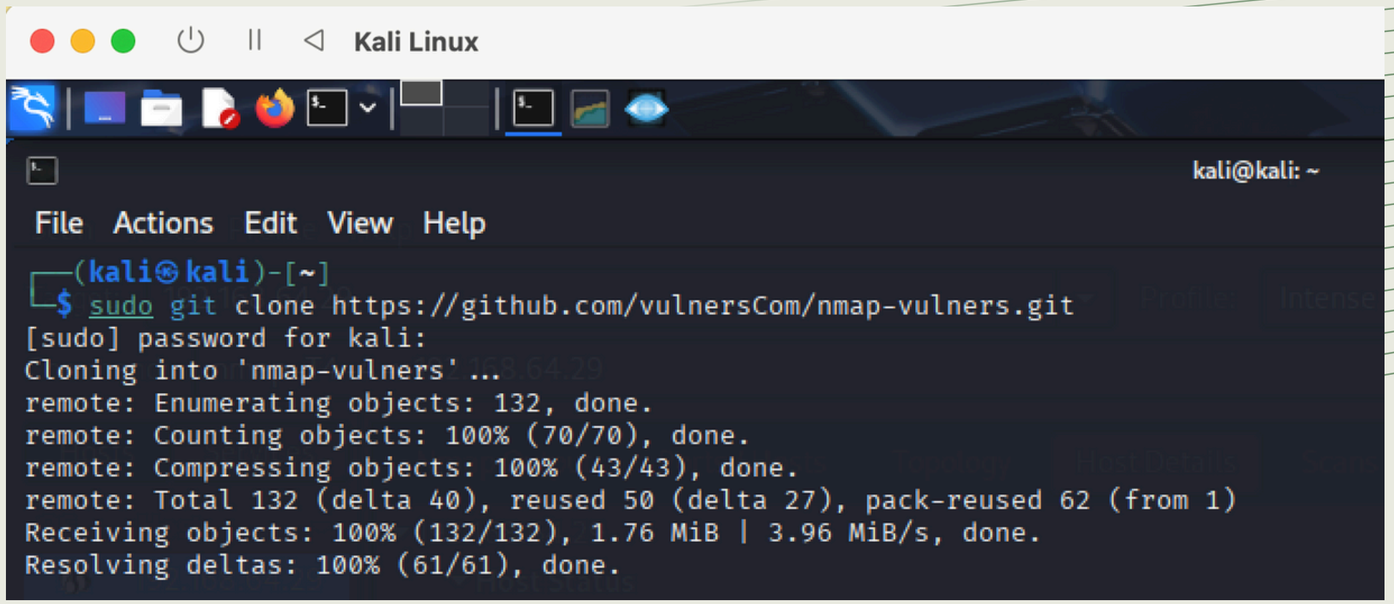


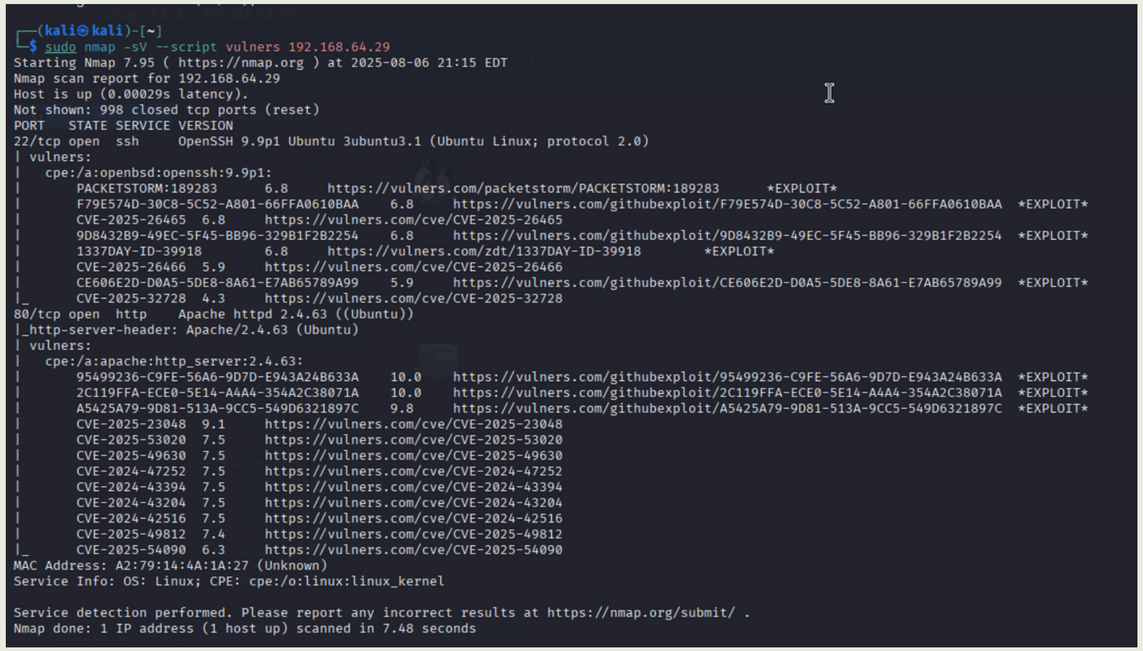


Luego realicé un Intense Scan en Zenmap para obtener puertos abiertos y los detalles del host que en mi caso es la IP: 192.168.64.29.

Herramientas de análisis: Nmap

Instalación del **script nmap-vulners**, para realizar el escaneo descargando un repositorio de GitHub, donde la URL es el repositorio oficial de nmap-vulners, un script de Nmap para buscar vulnerabilidades conocidas de forma automática.





Al descargar la librería, procedí a realizar un escaneo de vulnerabilidades con el comando: sudo nmap -sV —script vulners (IP del servidor de Ubuntu) y el resultado fueron las vulnerabilidades que se encontraron en el escaneo las cuales inician con las siglas CVE que fue la librería que se descargó previamente.

PARTE IV

Vulnerabilidades encontradas

Encontradas en el puerto 22

CVE-2025-26465 Detail

Una vulnerabilidad en OpenSSH permite ataques de intermediario (man-in-the-middle) cuando está habilitada la opción VerifyHostKeyDNS.

Description

A vulnerability was found in OpenSSH when the VerifyHostKeyDNS option is enabled. A machine-in-the-middle attack can be performed by a malicious machine impersonating a legit server. This issue occurs due to how OpenSSH mishandles error codes in specific conditions when verifying the host key. For an attack to be considered successful, the attacker needs to manage to exhaust the client's memory resource first, turning the attack complexity high.

LinkNIST

LinkVulners

Link NIST Link Vulners

CVE-2025-26466 Detail

Description

A vulnerability was found in OpenSSH when the VerifyHostKeyDNS option is enabled. A machine-in-the-middle attack can be performed by a malicious machine impersonating a legit server. This issue occurs due to how OpenSSH mishandles error codes in specific conditions when verifying the host key. For an attack to be considered successful, the attacker needs to manage to exhaust the client's memory resource first, turning the attack complexity high.

Encontradas en el puerto 22

Afecta a OpenSSH anterior a la versión 10.0, donde DisableForwarding no deshabilita correctamente el reenvío.

CVE-2025-32728 Detail

Description

In sshd in OpenSSH before 10.0, the DisableForwarding directive does not adhere to the documentation stating that it disables X11 and agent forwarding.

Link NIST

Link vulners

Encontradas en el puerto 80

CVE-2025-23048 Detail

Description

In some mod_ssl configurations on Apache HTTP Server 2.4.35 through to 2.4.63, an access control bypass by trusted clients is possible using TLS 1.3 session resumption. Configurations are affected when mod_ssl is configured for multiple virtual hosts, with each restricted to a different set of trusted client certificates (for example with a different SSLCACertificateFile/Path setting). In such a case, a client trusted to access one virtual host may be able to access another virtual host, if SSLStrictSNIVHostCheck is not enabled in either virtual host.

Link NIST

Link Vulners

Servidor HTTP Apache permite ataques DoS debido a problemas de memoria.

Eludir el control de acceso en el servidor HTTP Apache afecta a mod_ssl con la reanudación de sesión TLS 1.3.

Link NIST Link Vulners

CVE-2025-53020 Detail

Description

Late Release of Memory after Effective Lifetime vulnerability in Apache HTTP Server. This issue affects Apache HTTP Server: from 2.4.17 up to 2.4.63. Users are recommended to upgrade to version 2.4.64, which fixes the issue.

Encontradas en el puerto 80

CVE-2024-47252 Detail

Description

Insufficient escaping of user-supplied data in mod_ssl in Apache HTTP Server 2.4.63 and earlier allows an untrusted SSL/TLS client to insert escape characters into log files in some configurations. In a logging configuration where CustomLog is used with "%{varname}x" or "%{varname}c" to log variables provided by mod_ssl such as SSL_TLS_SNI, no escaping is performed by either mod_log_config or mod_ssl and unsanitized data provided by the client may appear in log files.

Eludir el control de acceso en el servidor HTTP Apache afecta a mod_ssl con la reanudación de sesión TLS 1.3.

Link NIST Link Vulners

Link NIST

Link Vulners

Vulnerabilidad de denegación de servicio en el servidor HTTP Apache mod_proxy_http2 debido a clientes no confiables.

CVE-2025-49630 Detail

Description

In certain proxy configurations, a denial of service attack against Apache HTTP Server versions 2.4.26 through to 2.4.63 can be triggered by untrusted clients causing an assertion in mod_proxy_http2. Configurations affected are a reverse proxy is configured for an HTTP/2 backend, with ProxyPreserveHost set to "on".

Encontradas en el puerto 80

CVE-2024-43394 Detail

Description

Server-Side Request Forgery (SSRF) in Apache HTTP Server on Windows allows to potentially leak NTLM hashes to a malicious server via mod_rewrite or apache expressions that pass unvalidated request input. This issue affects Apache HTTP Server: from 2.4.0 through 2.4.63. Note: The Apache HTTP Server Project will be setting a higher bar for accepting vulnerability reports regarding SSRF via UNC paths. The server offers limited protection against administrators directing the server to open UNC paths. Windows servers should limit the hosts they will connect over via SMB based on the nature of NTLM authentication.

La falsificación de solicitudes del lado del servidor en el servidor HTTP Apache en Windows puede filtrar hash NTLM a través de entradas no validadas.

Link NIST

Link Vulners

Link NIST

Link Vulners

CVE-2024-43204 Detail

Description

SSRF in Apache HTTP Server with mod_proxy loaded allows an attacker to send outbound proxy requests to a URL controlled by the attacker. Requires an unlikely configuration where mod_headers is configured to modify the Content-Type request or response header with a value provided in the HTTP request. Users are recommended to upgrade to version 2.4.64 which fixes this issue.

La vulnerabilidad SSRF en mod_headers del servidor HTTP Apache requiere una actualización a la versión 2.4.64.

Encontradas en el puerto 80

CVE-2024-42516 Detail

Description

HTTP response splitting in the core of Apache HTTP Server allows an attacker who can manipulate the Content-Type response headers of applications hosted or proxied by the server can split the HTTP response. This vulnerability was described as CVE-2023-38709 but the patch included in Apache HTTP Server 2.4.59 did not address the issue. Users are recommended to upgrade to version 2.4.64, which fixes this issue.

Link NIST

Link Vulners

Las versiones del servidor HTTP Apache hasta la 2.4.63 son vulnerables a la desincronización HTTP para las actualizaciones TLS

La falsificación de solicitudes del lado del servidor en el servidor HTTP Apache en Windows puede filtrar hash NTLM a través de entradas no validadas.

Link NIST Link Vulners

CVE-2025-49812 Detail

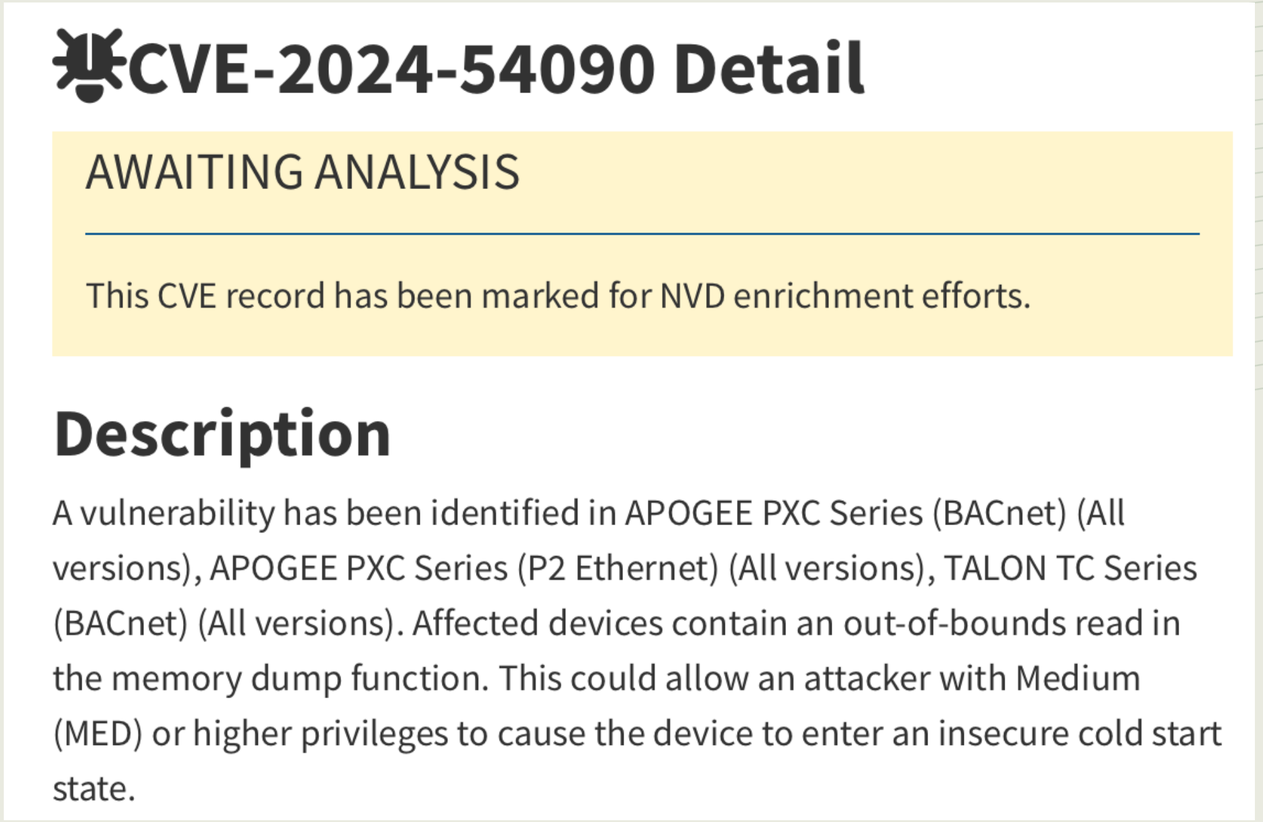
Description

In some mod_ssl configurations on Apache HTTP Server versions through to 2.4.63, an HTTP desynchronisation attack allows a man-in-the-middle attacker to hijack an HTTP session via a TLS upgrade. Only configurations using "SSLEngine optional" to enable TLS upgrades are affected. Users are recommended to upgrade to version 2.4.64, which removes support for TLS upgrade.

Encontradas en el puerto 80

Link NIST

Link Vulners



CVE-2024-54090 Detail

AWAITING ANALYSIS

This CVE record has been marked for NVD enrichment efforts.

Description

A vulnerability has been identified in APOGEE PXC Series (BACnet) (All versions), APOGEE PXC Series (P2 Ethernet) (All versions), TALON TC Series (BACnet) (All versions). Affected devices contain an out-of-bounds read in the memory dump function. This could allow an attacker with Medium (MED) or higher privileges to cause the device to enter an insecure cold start state.

Una vulnerabilidad en los dispositivos APOGEE y TALON permite un arranque en frío inseguro mediante una lectura fuera de límites.

PARTE V

¿Cómo resolverlas?

Soluciones para las vulnerabilidades encontradas en el puerto 22

CVE-2025-26465

CVE-2025-26466

Para garantizar la seguridad continua, se recomienda actualizar los sistemas afectados a la versión OpenSSH 9.9p2 lo antes posible.

CVE-2025-32728

CWE



Phase(s)	Mitigation
Architecture and Design	**Strategy:** ***Libraries or Frameworks*** Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. For example, use anti-CSRF packages such as the OWASP CSRFGuard. [REF-330] Another example is the ESAPI Session Management control, which includes a component for CSRF. [REF-45]
Implementation	Ensure that the application is free of cross-site scripting issues (CWE-79), because most CSRF defenses can be bypassed using attacker-controlled script.
Architecture and Design	Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable (CWE-330). [REF-332] **Note:** Note that this can be bypassed using XSS (CWE-79).
Architecture and Design	Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation. **Note:** Note that this can be bypassed using XSS (CWE-79).
Architecture and Design	Use the "double-submitted cookie" method as described by Felten and Zeller: When a user visits a site, the site should generate a pseudorandom value and set it as a cookie on the user's machine. The site should require every form submission to include this value as a form value and also as a cookie value. When a POST request is sent to the site, the request should only be considered valid if the form value and the cookie value are the same. Because of the same-origin policy, an attacker cannot read or modify the value stored in the cookie. To successfully submit a form on behalf of the user, the attacker would have to correctly guess the pseudorandom value. If the pseudorandom value is cryptographically strong, this will be prohibitively difficult. This technique requires Javascript, so it may not work for browsers that have Javascript disabled. [REF-331] **Note:** Note that this can probably be bypassed using XSS (CWE-79), or when using web technologies that enable the attacker to read raw headers from HTTP requests.

Soluciones para las vulnerabilidades encontradas en el puerto 80

CVE-2025-23048 CWE

Potential Mitigations

Phase(s)	Mitigation
Architecture and Design; Operation	Very carefully manage the setting, management, and handling of privileges. Explicitly manage trust zones in the software.
Architecture and Design	**Strategy: *Separation of Privilege*** Compartmentalize the system to have "safe" areas where trust boundaries can be unambiguously drawn. Do not allow sensitive data to go outside of the trust boundary and always be careful when interfacing with a compartment outside of the safe area. Ensure that appropriate compartmentalization is built into the system design, and the compartmentalization allows for and reinforces privilege separation functionality. Architects and designers should rely on the principle of least privilege to decide the appropriate time to use privileges and the time to drop privileges.

CVE-2024-42516 CWE

Potential Mitigations

Phase(s)	Mitigation
Architecture and Design	**Strategy: *Attack Surface Reduction*** Consider using language-theoretic security (LangSec) techniques that characterize inputs using a formal language and build "recognizers" for that language. This effectively requires parsing to be a distinct layer that effectively enforces a boundary between raw input and internal data representations, instead of allowing parser code to be scattered throughout the program, where it could be subject to errors or inconsistencies that create weaknesses. [REF-1109] [REF-1110] [REF-1111]
Architecture and Design	**Strategy: *Libraries or Frameworks*** Use an input validation framework such as Struts or the OWASP ESAPI Validation API. Note that using a framework does not automatically address all input validation problems; be mindful of weaknesses that could arise from misusing the framework itself (CWE-1173).
Architecture and Design; Implementation	**Strategy: *Attack Surface Reduction*** Understand all the potential areas where untrusted inputs can enter the product, including but not limited to: parameters or arguments, cookies, anything read from the network, environment variables, reverse DNS lookups, query results, request headers, URL components, e-mail, files, filenames, databases, and any external systems that provide data to the application. Remember that such inputs may be obtained indirectly through API calls.
Implementation	**Strategy: *Input Validation*** Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if the input is only expected to contain colors such as "red" or "blue." Do not rely exclusively on looking for malicious or malformed inputs. This is likely to miss at least one undesirable input, especially if the code's environment changes. This can give attackers enough room to bypass the intended validation. However, denylists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. **Effectiveness: High**

Soluciones para las vulnerabilidades encontradas en el puerto 80

CVE-2025-49812

CWE

Potential Mitigations

Phase(s)	Mitigation
Architecture and Design	**Strategy:** ***Libraries or Frameworks*** Use an authentication framework or library such as the OWASP ESAPI Authentication feature.

CVE-2024-54090

CWE

Potential Mitigations

Phase(s)	Mitigation
Implementation	**Strategy:** ***Input Validation*** Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if the input is only expected to contain colors such as "red" or "blue." Do not rely exclusively on looking for malicious or malformed inputs. This is likely to miss at least one undesirable input, especially if the code's environment changes. This can give attackers enough room to bypass the intended validation. However, denylists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright. To reduce the likelihood of introducing an out-of-bounds read, ensure that you validate and ensure correct calculations for any length argument, buffer size calculation, or offset. Be especially careful of relying on a sentinel (i.e. special character such as NUL) in untrusted inputs.
Architecture and Design	**Strategy:** ***Language Selection*** Use a language that provides appropriate memory abstractions.

Soluciones para las vulnerabilidades encontradas en el puerto 80

CVE-2025-53020

CWE

Potential Mitigations

Phase(s)	Mitigation
Implementation	**Strategy:** ***Libraries or Frameworks*** Choose a language or tool that provides automatic memory management, or makes manual memory management less error-prone. For example, glibc in Linux provides protection against free of invalid pointers. When using Xcode to target OS X or iOS, enable automatic reference counting (ARC) [REF-391]. To help correctly and consistently manage memory when programming in C++, consider using a smart pointer class such as std::auto_ptr (defined by ISO/IEC ISO/IEC 14882:2003), std::shared_ptr and std::unique_ptr (specified by an upcoming revision of the C++ standard, informally referred to as C++ 1x), or equivalent solutions such as Boost.
Architecture and Design	Use an abstraction library to abstract away risky APIs. Not a complete solution.
Architecture and Design; Build and Compilation	The Boehm-Demers-Weiser Garbage Collector or valgrind can be used to detect leaks in code. **Note:** This is not a complete solution as it is not 100% effective.

CVE-2024-47252

CWE

Potential Mitigations

Phase(s)	Mitigation
	Developers should anticipate that escape, meta and control characters/sequences will be injected/removed/manipulated in the input vectors of their product. Use an appropriate combination of denylists and allowlists to ensure only valid, expected and appropriate input is processed by the system.
Implementation	**Strategy:** ***Input Validation*** Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a list of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if the input is only expected to contain colors such as "red" or "blue." Do not rely exclusively on looking for malicious or malformed inputs. This is likely to miss at least one undesirable input, especially if the code's environment changes. This can give attackers enough room to bypass the intended validation. However, denylists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright.
Implementation	**Strategy:** ***Output Encoding*** While it is risky to use dynamically-generated query strings, code, or commands that mix control and data together, sometimes it may be unavoidable. Properly quote arguments and escape any special characters within those arguments. The most conservative approach is to escape or filter all characters that do not pass an extremely strict allowlist (such as everything that is not alphanumeric or white space). If some special characters are still needed, such as white space, wrap each argument in quotes after the escaping/filtering step. Be careful of argument injection (CWE-88).
Implementation	**Strategy:** ***Input Validation*** Inputs should be decoded and canonicalized to the application's current internal representation before being validated (CWE-180). Make sure that the application does not decode the same input twice (CWE-174). Such errors could be used to bypass allowlist validation schemes by introducing dangerous inputs after they have been checked.

Soluciones para las vulnerabilidades encontradas en el puerto 80

CVE-2025-49630

CWE

Potential Mitigations

Phase(s)	Mitigation
Implementation	Make sensitive open/close operation non reachable by directly user-controlled data (e.g. open/close resources)
Implementation	**Strategy:** ***Input Validation*** Perform input validation on user data.

CVE-2024-43204

low: Apache HTTP Server: SSRF with mod_headers setting Content-Type header (CVE-2024-43204)

SSRF in Apache HTTP Server with mod_proxy loaded allows an attacker to send outbound proxy requests to a URL controlled by the attacker. Requires an unlikely configuration where mod_headers is configured to modify the Content-Type request or response header with a value provided in the HTTP request.

Users are recommended to upgrade to version 2.4.64 which fixes this issue.

Soluciones generales para el sistema

Actualizaciones y Parches

Mantén el sistema actualizado:

- Usa `sudo apt update` y `sudo apt upgrade -y` regularmente para asegurarte de que todos los paquetes estén actualizados.

Configuración del Firewall

Instala y configura UFW (Uncomplicated Firewall):

- Habilita UFW: `sudo ufw enable`
- Configura reglas para permitir solo el tráfico necesario (por ejemplo, SSH y HTTP).
- `sudo ufw allow 22` (para SSH)
- `sudo ufw allow 80` (para HTTP)

Seguridad en SSH

Desactiva el acceso root por SSH:

- Edita el archivo `/etc/ssh/sshd_config` y establece `PermitRootLogin no`.

Cambia el puerto SSH por defecto (22):

- Modifica `Port 22` a otro número no estándar (por ejemplo, `Port 2222`).

Usa la autenticación basada en clave:

- Desactiva `PasswordAuthentication` y asegura que los usuarios tengan claves SSH configuradas.

Configurar Apache

Actualiza el servidor web:

- Asegúrate de que tu versión de Apache esté actualizada.

Desactiva módulos innecesarios:

- Usa `sudo a2dismod <nombre_del_modulo>` para desactivar módulos que no necesites.

Configura headers de seguridad:

- Agrega cabeceras como `X-Content-Type-Options`, `X-Frame-Options`, y `Content-Security-Policy` en la configuración de tu servidor.

Soluciones generales para el sistema

Revisión de Seguridad

Utiliza herramientas de escaneo:

- Puedes usar herramientas como `lynis` o `rkhunter` para escanear tu sistema y obtener informes sobre posibles vulnerabilidades.

Revisión regular de logs:

- Monitorea logs `/var/log/auth.log` para detectar intentos de acceso no autorizados y otras actividades inusuales.

Seguridad de las Aplicaciones

Asegúrate de que las aplicaciones web estén actualizadas:

- Verifica que las aplicaciones que estés ejecutando (por ejemplo, CMS) estén en su última versión.

Implementa medidas contra inyección de SQL y vulnerabilidades en XSS.

Monitoreo y Alertas

Configura herramientas de monitoreo:

- Usa herramientas como `fail2ban` para prevenir ataques de fuerza bruta.

“Difícil de ver es. Siempre en movimiento el futuro está”.

Nada es tan incierto como el futuro, por eso es necesario asumir que el mañana está lleno de probabilidades que no siempre podemos controlar...

