

**PROYECTO FINAL
DE
CIBERSEGURIDAD**

Mireia Gómez

1. Introducción

- 1.1 Contexto empresarial (4Geeks Academy – entorno simulado)
- 1.2 Importancia de la ciberseguridad en infraestructuras Linux
- 1.3 Objetivos generales y específicos
- 1.4 Alcance del proyecto
- 1.5 Limitaciones del laboratorio
- 1.6 Metodología aplicada (visión general)

2. Marco Teórico

- 2.1 Seguridad en sistemas Linux
- 2.2 SSH – Arquitectura y riesgos
- 2.3 FTP – Riesgos y alternativas seguras
- 2.4 WordPress como superficie de ataque
- 2.5 Análisis Forense Digital
- 2.6 Gestión de Incidentes (NIST)
- 2.7 ISO 27001 – Fundamentos

3. Entorno del Laboratorio

- 3.1 Arquitectura de red
- 3.2 Configuración de máquinas virtuales
- 3.3 Versiones de software utilizadas

4. Fase 1 – Análisis Forense

- 4.1 Reconocimiento del sistema
- 4.2 Identificación del incidente
- 4.3 Línea temporal del ataque
- 4.4 Recolección de evidencias
- 4.5 Análisis de usuarios y sesiones
- 4.6 Análisis de servicios activos
- 4.7 Búsqueda de persistencia
- 4.8 Evaluación del impacto

5. Fase 1 – Mitigación y Hardening

- 5.1 Corrección SSH
- 5.2 Corrección FTP
- 5.3 Hardening WordPress
- 5.4 Configuración segura de Apache
- 5.5 Validación post-corrección

6. Fase 2 – Nueva Vulnerabilidad

- 6.1 Escaneo externo
- 6.2 Enumeración WordPress
- 6.3 Explotación controlada
- 6.4 Análisis del riesgo
- 6.5 Mitigación aplicada
- 6.6 Validación posterior

7. Análisis de Riesgos

- 7.1 Identificación de activos
- 7.2 Identificación de amenazas
- 7.3 Identificación de vulnerabilidades
- 7.4 Evaluación impacto/probabilidad
- 7.5 Matriz de riesgo
- 7.6 Tratamiento del riesgo

8. Plan de Respuesta a Incidentes

- 8.1 Preparation
- 8.2 Identification
- 8.3 Containment
- 8.4 Eradication
- 8.5 Recovery
- 8.6 Lessons Learned

9. Sistema de Gestión de Seguridad de la Información

- 9.1 Alcance del SGSI
- 9.2 Política de seguridad
- 9.3 Declaración de aplicabilidad
- 9.4 Controles implementados
- 9.5 Mejora continua
- 9.6 Auditorías internas

10. Impacto Legal y Cumplimiento Normativo

- 10.1 RGPD y brechas de seguridad
- 10.2 10.2 Obligación de notificación
- 10.3 10.3 Posibles sanciones económicas
- 10.4 10.4 Responsabilidad empresarial
- 10.5 10.5 Buenas prácticas regulatorias

11. Comparativa Estado Inicial vs Estado Final

- 11.1 Análisis técnico comparativo
- 11.2 Reducción de superficie de ataque
- 11.3 Nivel de madurez alcanzado

12. Recomendaciones Estratégicas

- 12.1 Técnicas
- 12.2 Organizativas
- 12.3 Formación del personal
- 12.4 Monitorización y SIEM
- 12.5 Implementación futura de MFA

13. Conclusiones

14. Glosario

15. Anexos

- Logs completos
- Escaneos Nmap
- Scripts
- Hashes
- Configuraciones antes/después

1. Introducción

1.1 Contexto empresarial

Este proyecto tiene como objetivo simular un escenario real de ciberseguridad, que representa la infraestructura tecnológica de en el que un servidor crítico de la empresa 4Geeks Academy ha sido comprometido. En este escenario de laboratorio, se proporciona una máquina virtual basada en Debian que simula un servidor empresarial expuesto. En el cual, se albergan una serie de servicios críticos como acceso remoto mediante SSH, servidor web Apache y una aplicación WordPress respaldada por una base de datos MariaDB.

A pesar de que se trata de un entorno académico controlado, el laboratorio reproduce situaciones reales que pueden ocurrir en cualquier organización que gestione infraestructuras Linux expuestas a la red. La configuración inicial del servidor presenta debilidades deliberadas que permiten simular un incidente de seguridad, facilitando el análisis forense, la detección de vulnerabilidades y la aplicación de medidas correctivas.

En este proyecto, se asume el rol de analista de ciberseguridad dentro de un equipo de respuesta a incidentes (SOC/Blue Team), para confrontar una situación realista en la que se debe investigar un posible compromiso del sistema, identificar la vía de entrada, mitigar el ataque y fortalecer la seguridad del entorno.

Este enfoque permite trasladar conceptos teóricos a un caso práctico, desarrollando competencias técnicas y estratégicas alineadas con estándares profesionales del sector.

1.2 Importancia de la ciberseguridad en infraestructuras Linux

Los sistemas Linux constituyen la base de una gran parte de las infraestructuras tecnológicas actuales, incluyendo servidores web, bases de datos, entornos cloud y sistemas empresariales críticos. Su estabilidad, escalabilidad y robustez los convierten en una elección predominante en entornos productivos.

Sin embargo, la seguridad de una infraestructura Linux no depende únicamente del sistema operativo, sino de su correcta configuración, gestión de accesos, políticas de actualización y control de servicios expuestos. Configuraciones inseguras, como permitir acceso root por SSH o utilizar contraseñas débiles, pueden convertir un sistema robusto en un objetivo vulnerable.

En entornos empresariales, un compromiso de servidor puede implicar:

- Pérdida de confidencialidad de datos.
- Alteración o destrucción de información.
- Interrupción de servicios (impacto en disponibilidad).
- Daño reputacional.
- Sanciones legales en caso de incumplimiento normativo.

Por ello, la ciberseguridad en sistemas Linux debe abordarse desde una perspectiva integral que incluya:

- Hardening del sistema.
- Monitorización continua.
- Gestión de vulnerabilidades.
- Respuesta estructurada ante incidentes.
- Cumplimiento de estándares internacionales.

Este proyecto permite comprender cómo pequeñas configuraciones inseguras pueden convertirse en vectores de ataque reales, reforzando la importancia de aplicar buenas prácticas desde el diseño inicial de la infraestructura.

1.3 Objetivos generales y específicos

Objetivo general

El objetivo principal del proyecto es analizar, contener y mitigar un incidente de seguridad simulado en un servidor Debian comprometido, aplicando metodologías profesionales de respuesta a incidentes y fortalecimiento de infraestructuras.

Objetivos específicos

- Identificar la vía de entrada utilizada por el atacante.
- Analizar registros del sistema para determinar evidencias del compromiso.
- Detectar configuraciones inseguras en servicios críticos.
- Realizar pruebas de pentesting controladas para identificar vulnerabilidades adicionales.
- Aplicar medidas de mitigación y hardening.
- Diseñar un plan de respuesta a incidentes basado en NIST SP 800-61.

- Proponer un Sistema de Gestión de Seguridad de la Información alineado con ISO 27001.
- Documentar todo el proceso bajo criterios forenses y profesionales.

Estos objetivos permiten abordar el incidente desde una perspectiva técnica, organizativa y estratégica.

1.4 Alcance del proyecto

El alcance del presente proyecto se limita al análisis y fortalecimiento del servidor Debian proporcionado en el laboratorio. Se evaluaron los siguientes componentes:

- Servicio SSH (acceso remoto).
- Servicio FTP.
- Servidor web Apache.
- Aplicación WordPress.
- Base de datos MariaDB.
- Configuraciones del sistema operativo.

El análisis incluyó:

- Revisión de logs.
- Escaneo de puertos y servicios.
- Evaluación de vulnerabilidades web.
- Pruebas de explotación controlada.
- Corrección de configuraciones inseguras.
- Validación posterior de mitigaciones.

No se realizaron pruebas fuera del entorno de laboratorio ni ataques contra sistemas externos. Tampoco se llevó a cabo explotación destructiva ni pruebas de denegación de servicio reales, dado que el enfoque del proyecto se centra en análisis forense y hardening.

1.5 Limitaciones del laboratorio

Al tratarse de un entorno académico simulado, existen ciertas limitaciones inherentes al laboratorio:

- La infraestructura se compone de una única máquina virtual, sin segmentación real de red empresarial.
- No se dispone de herramientas avanzadas de monitorización en tiempo real (SIEM corporativo).
- El entorno no simula tráfico real de usuarios concurrentes.
- No se cuenta con平衡adores de carga ni arquitectura distribuida.
- El análisis forense depende de una imagen local del disco sin integración con herramientas empresariales de cadena de custodia formal.

Estas limitaciones no restan valor al ejercicio, sino que permiten focalizar el aprendizaje en la identificación de vulnerabilidades y aplicación de buenas prácticas de seguridad.

1.6 Metodología aplicada (visión general)

El proyecto se desarrolló siguiendo una metodología estructurada basada en estándares internacionales y buenas prácticas del sector:

1. **Identificación del incidente:** Revisión de logs y servicios expuestos.
2. **Análisis forense inicial:** Inspección de historial de comandos y configuraciones críticas.
3. **Contención:** Bloqueo inmediato del vector de acceso identificado.
4. **Eradicación:** Eliminación de configuraciones inseguras y revisión de persistencia.
5. **Hardening:** Aplicación de medidas de fortalecimiento del sistema.
6. **Validación:** Pruebas externas para confirmar la efectividad de las mitigaciones.
7. **Planificación estratégica:** Diseño de un plan de respuesta basado en NIST y SGSI alineado con ISO 27001.

Esta metodología permitió abordar el incidente de manera sistemática, documentada y profesional, simulando un entorno real de trabajo en un equipo de seguridad.

2. MARCO TEÓRICO

2.1 Seguridad en sistemas Linux

2.1.1 Modelo de permisos UNIX

El modelo de seguridad en sistemas Linux se basa en el tradicional esquema de permisos UNIX, diseñado bajo un enfoque multiusuario. Cada archivo y directorio posee tres tipos de permisos fundamentales:

- **Lectura (r)**
- **Escritura (w)**
- **Ejecución (x)**

Estos permisos se aplican a tres categorías de usuarios:

- Propietario (user)
- Grupo (group)
- Otros (others)

La representación simbólica más común es:

-rwxr-xr-x

Donde:

- El primer bloque corresponde al propietario.
- El segundo al grupo.
- El tercero a otros usuarios.

Desde una perspectiva de seguridad, el modelo UNIX permite aplicar el principio de segregación de accesos. Sin embargo, configuraciones incorrectas como permisos 777 (lectura, escritura y ejecución para todos) representan una grave debilidad, ya que cualquier usuario podría modificar o ejecutar archivos críticos.

En el contexto del proyecto, se identificaron permisos inseguros en archivos del directorio WordPress, lo cual incrementaba la superficie de ataque del sistema.

2.1.2 Gestión de usuarios y privilegios

Linux opera bajo un modelo jerárquico de privilegios. El usuario **root** posee control absoluto sobre el sistema, mientras que los usuarios estándar tienen permisos limitados.

La gestión adecuada de usuarios implica:

- Creación controlada de cuentas.
- Asignación correcta de grupos.
- Uso de sudo para privilegios temporales.
- Auditoría periódica de accesos.

Permitir acceso remoto directo al usuario root representa una práctica insegura, ya que elimina una capa adicional de trazabilidad y control.

En el proyecto, el acceso root por SSH fue identificado como el vector inicial de compromiso.

2.1.3 Principio de mínimo privilegio

El principio de mínimo privilegio establece que cada usuario o proceso debe disponer únicamente de los permisos estrictamente necesarios para desempeñar su función.

Aplicar este principio reduce:

- Riesgo de escalada de privilegios.
- Impacto de credenciales comprometidas.
- Daños potenciales en caso de intrusión.

En el entorno analizado, la desactivación del acceso root por SSH y la eliminación de permisos 777 fueron medidas alineadas con este principio.

2.2 SSH – Arquitectura y riesgos

2.2.1 Funcionamiento del protocolo SSH

SSH (Secure Shell) es un protocolo criptográfico que permite acceso remoto seguro a sistemas a través de redes no confiables.

Opera típicamente sobre el puerto 22 y proporciona:

- Cifrado de extremo a extremo.
- Autenticación del cliente y del servidor.
- Integridad de datos.

SSH reemplazó protocolos inseguros como Telnet, que transmitían credenciales en texto plano.

2.2.2 Autenticación por contraseña vs clave pública

Existen dos métodos principales de autenticación en SSH:

Autenticación por contraseña

- El usuario introduce credenciales.
- Vulnerable a ataques de fuerza bruta.
- Riesgo elevado si se utilizan contraseñas débiles.

Autenticación por clave pública

- Se utiliza un par de claves (privada/pública).
- La clave privada permanece en el cliente.
- Mayor seguridad.
- Difícil de vulnerar mediante fuerza bruta.

En el proyecto, se deshabilitó la autenticación por contraseña para reducir el riesgo de ataques automatizados.

2.2.3 Riesgos de PermitRootLogin

La directiva:

PermitRootLogin yes

Permite acceso remoto directo al usuario root.

Riesgos asociados:

- Ataques automatizados contra cuentas privilegiadas.
- Dificultad de trazabilidad.
- Mayor impacto ante compromiso.

La configuración segura recomendada es:

PermitRootLogin no

PasswordAuthentication no

2.3 FTP – Riesgos y alternativas seguras

2.3.1 FTP vs SFTP

FTP (File Transfer Protocol) es un protocolo tradicional para transferencia de archivos. Opera por defecto en el puerto 21.

Problemas principales:

- Transmite credenciales en texto plano.
- No cifra el tráfico.
- Vulnerable a sniffing y ataques Man-in-the-Middle.

SFTP (SSH File Transfer Protocol):

- Opera sobre SSH.
- Cifra completamente la comunicación.
- Mucho más seguro.

2.3.2 Problemas históricos de FTP

Históricamente, FTP ha sido explotado mediante:

- Accesos anónimos mal configurados.
- Buffer overflows.
- Enumeración de usuarios.
- Subida de archivos maliciosos.

En el laboratorio se verificó que anonymous_enable=NO, mitigando uno de los riesgos clásicos.

2.4 WordPress como superficie de ataque

2.4.1 Arquitectura de WordPress

WordPress se compone de:

- Archivos PHP en /var/www/html
- Base de datos (MySQL/MariaDB)
- Directorio de plugins
- Directorio de uploads

Su popularidad lo convierte en un objetivo frecuente.

2.4.2 REST API

La REST API permite acceso programático a datos mediante:

/wp-json/wp/v2/

Puede exponer:

- Usuarios
- Posts
- Metadatos

Si no está protegida, facilita enumeración de usuarios.

2.4.3 Ataques comunes

- Fuerza bruta contra /wp-login.php
- Enumeración de usuarios
- Webshell subida vía plugin vulnerable
- Directory listing
- SQL Injection

En el proyecto se identificó y corrigió directory listing.

2.5 Análisis Forense Digital

2.5.1 Cadena de custodia

La cadena de custodia garantiza que la evidencia no ha sido alterada.

Incluye:

- Identificación de la evidencia.
- Documentación.
- Cálculo de hashes.
- Registro de accesos.

2.5.2 Hashes de integridad

Algoritmos como:

- MD5
- SHA-1
- SHA-256

Permiten verificar que un archivo no ha sido modificado.

2.5.3 Volatilidad de evidencias

Las evidencias más volátiles:

- Procesos en memoria
- Conexiones activas
- RAM

Las menos volátiles:

- Disco
- Logs
- Configuraciones

2.5.4 Logs en Linux (journald vs rsyslog)

Linux puede almacenar logs mediante:

- **journald** (systemd journal)
- **rsyslog**

Logs relevantes:

- `/var/log/auth.log`
- `journalctl -u ssh`

2.6 Gestión de Incidentes (NIST)

El modelo NIST SP 800-61 define cuatro fases:

1. Preparación
2. Detección y análisis
3. Contención, erradicación y recuperación
4. Actividades posteriores al incidente

Este modelo fue aplicado conceptualmente en el proyecto.

2.7 ISO 27001 – Fundamentos

2.7.1 Qué es un SGSI

Un SGSI es un sistema estructurado para gestionar la seguridad de la información.

2.7.2 Ciclo PDCA

- Plan
- Do
- Check
- Act

Modelo de mejora continua.

2.7.3 Anexo A

Incluye controles como:

- Control de accesos
- Seguridad física
- Gestión de incidentes
- Continuidad del negocio

3. ENTORNO DEL LABORATORIO

3.1 Arquitectura de red

El entorno del laboratorio fue diseñado como una red interna aislada con el objetivo de simular una infraestructura empresarial básica sin exponer los sistemas a Internet real.

Se utilizaron dos máquinas virtuales:

- **Servidor Debian** (máquina objetivo)
- **Kali Linux** (máquina atacante / auditor)

Ambas máquinas fueron configuradas en una red interna de VirtualBox denominada LabNetwork, permitiendo comunicación directa entre ellas sin acceso externo.

Esta configuración garantiza:

- Aislamiento del entorno.
- Seguridad durante pruebas de explotación.
- Simulación realista de un ataque interno o lateral.

◆ Segmentación lógica del laboratorio

Elemento	Rol	Sistema Operativo	IP
Debian Server	Servidor	Debian 12	192.168.50.10
Kali Linux	Máquina de pruebas	Kali Linux	192.168.50.20

3.2 Configuración de máquinas virtuales

Las máquinas virtuales fueron configuradas en VirtualBox con los siguientes parámetros:

- ◆ **Debian Server**

Parámetro	Valor
Sistema Operativo	Debian 12
CPU	1-2 vCPU
RAM	2GB
Disco	30GB
Red	Internal Network (LabNetwork)
Servicios activos	SSH, FTP, Apache, MariaDB

- ◆ **Kali Linux**

Parámetro	Valor
Sistema Operativo	Kali Lunx
CPU	1-2 vCPU
RAM	2GB
Disco	20-30GB
Red	Internal Network (LabNetwork)
Servicios activos	Auditor / Pentesting

3.3 Versiones de software utilizadas

El entorno incluía las siguientes versiones detectadas mediante escaneo:

Servicio	Puerto	Versión
OpenSSH	22	9.2P1 Debian 2+deb12u3
Vsftpd	21	3.0.3
Apache	80	2.4.62
MariaDB	3306	Versión incluida en Debian 12
WordPress	Detectado via headers	Instalación activa

4. FASE 1 – ANÁLISIS FORENSE

4.1 Reconocimiento del sistema

El primer paso del análisis forense consistió en identificar el entorno del sistema comprometido, confirmando la identidad del usuario, la configuración de red y el estado general del servidor.

Se ejecutaron comandos básicos de reconocimiento:

- **Identidad del sistema:**

- **ip a**

Permite identificar la dirección IP asignada al servidor, la interfaz de red activa y el rango de red utilizado.

- **whoami**

Permite verificar el usuario con el que se está trabajando inicialmente.

- **sudo -i**

Se utilizó para acceder al contexto de administración y poder revisar logs y configuraciones críticas.

- **date**

Se utilizó para registrar la fecha/hora del análisis, esencial en un contexto forense.

- **uptime**

Se utilizó para conocer cuánto tiempo llevaba activo el sistema, detectando si había reinicios recientes o actividad prolongada.

Estos comandos permitieron establecer el contexto inicial del análisis y asegurar que el sistema estaba accesible para continuar la investigación.

```
debian@debian:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noq
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 scope host noprefixroute
            valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu
oup default qlen 1000
    link/ether 08:00:27:00:9b:31 brd ff:ff:ff:ff:ff:ff
        inet 192.168.50.10/24 brd 192.168.50.255 scop
            valid_lft forever preferred_lft forever
        inet6 fe80::a00:27ff:fe00:9b31/64 scope link
            valid_lft forever preferred_lft forever
debian@debian:~$ sudo -i
[sudo] password for debian:
root@debian:~# whoami
root
root@debian:~#
```

The screenshot shows a terminal window with the following details:

- Terminal title: debian@debian:~
- User: root
- Date: Thu Feb 12 05:19:16 AM EST 2026
- Host name: debian
- Icon name: computer-vm
- Chassis: vm
- Machine ID: 41b6de202c3f48fdaa490411748aaaff
- Boot ID: 028f92cd8ef04967ad90ff4ad2295951
- Virtualization: oracle
- Operating System: Debian GNU/Linux 12 (bookworm)
- Kernel: Linux 6.1.0-25-amd64
- Architecture: x86_64
- Hardware Vendor: innoteck GmbH
- Hardware Model: VirtualBox
- Firmware Version: VirtualBox
- PRETTY_NAME="Debian GNU/Linux 12 (bookworm)"
- NAME="Debian GNU/Linux"
- VERSION_ID="12"
- VERSION="12 (bookworm)"
- VERSION_CODENAME=bookworm
- VERSION_ID="12"
- VERSION="12 (bookworm)"
- VERSION_CODENAME=bookworm
- ID=debian
- HOME_URL="https://www.debian.org/"
- SUPPORT_URL="https://www.debian.org/support"
- BUG_REPORT_URL="https://bugs.debian.org/"

At the bottom, the command `root@debian:~# uptime` is shown with the output: 05:19:43 up 10 min, 2 users, load average: 0.12, 0.07, 0.06

- Usuarios y sesiones:

```
root@debian:~# id
uid=0(root) gid=0(root) groups=0(root)
root@debian:~# who
debian  tty7          2026-02-12 05:44 (:)
debian  pts/1           2026-02-12 05:45
root@debian:~# w
05:51:17 up 7 min, 2 users, load average: 0.32, 0.24, 0.14
USER   TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
debian  tty7          :0              05:44   7:36  10.11s  0.27s x-session-manag
debian  pts/1           -              05:45   5.00s  0.04s  0.16s sudo -i
root@debian:~# last -a | head -n 30
debian  tty7    Thu Feb 12 05:44 still logged in :0
reboot system boot  Thu Feb 12 05:43 still running  6.1.0-25-amd64
debian  tty7    Tue Oct  8 17:28 - crash (491+13:14) :0
reboot system boot  Tue Oct  8 17:28 still running  6.1.0-25-amd64
debian  tty7    Tue Oct  8 16:48 - crash (00:48) :0
reboot system boot  Tue Oct  8 16:48 still running  6.1.0-25-amd64
debian  tty7    Tue Oct  8 16:44 - crash (00:03) :0
reboot system boot  Tue Oct  8 16:43 still running  6.1.0-25-amd64
debian  tty7    Mon Sep 30 15:13 - crash (8+01:29) :0
reboot system boot  Mon Sep 30 15:09 still running  6.1.0-25-amd64
debian  tty7    Mon Sep 30 09:49 - 12:27 (02:38) :0
reboot system boot  Mon Sep 30 09:48 - 12:28 (02:39) 6.1.0-23-amd64
debian  tty7    Sat Sep 28 16:40 - crash (1+17:08) :0
```

```
root@debian:~# ps auxwvf | tee /root/evidencias/comandos/ps_auxwvf.txt
USER      PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
root      2 0.0 0.0 0 0 ? S 05:43 0:00 [kthreadd]
root      3 0.0 0.0 0 0 ? I< 05:43 0:00 \_ [rcu_gp]
root      4 0.0 0.0 0 0 ? I< 05:43 0:00 \_ [rcu_par_
gp]
root      5 0.0 0.0 0 0 ? I< 05:43 0:00 \_ [slub_fiu_
shwq]
root      6 0.0 0.0 0 0 ? I< 05:43 0:00 \_ [netns]
root     10 0.0 0.0 0 0 ? I< 05:43 0:00 \_ [mm_percp_
u_wq]
root     11 0.0 0.0 0 0 ? I 05:43 0:00 \_ [rcu_task_
s_kthread]
root     12 0.0 0.0 0 0 ? I 05:43 0:00 \_ [rcu_task_
s_rude_kthread]
root     13 0.0 0.0 0 0 ? I 05:43 0:00 \_ [rcu_task_
s_trace_kthread]
root     14 0.0 0.0 0 0 ? S 05:43 0:00 \_ [ksoftirq_
d/0]
root     15 0.0 0.0 0 0 ? I 05:43 0:00 \_ [rcu_pree_
mpt]
root     16 0.0 0.0 0 0 ? S 05:43 0:00 \_ [migratio_
n/0]
root     18 0.0 0.0 0 0 ? S 05:43 0:00 \_ [cpuhp/0]
```

- Se verificó el estado del servicio SSH

mediante el análisis de los registros del sistema con journalctl. Como se observa en la captura de pantalla, el servicio SSH se encontraba activo y escuchando en el puerto 22, lo que confirma la exposición del servicio en el momento del análisis.

```
root@debian:~# sudo journalctl -u ssh --no-pager | tail -n 80
Sep 30 12:25:16 debian systemd[1]: Starting ssh.service - OpenBSD Secure Shell server...
Sep 30 12:25:16 debian sshd[51422]: Server listening on 0.0.0.0 port 22.
```

4.2 Identificación del incidente

Tras el reconocimiento inicial, se procedió a identificar evidencias del compromiso.

El indicador principal encontrado fue un acceso exitoso al usuario root vía SSH, registrado en el sistema:

Accepted password for root from 192.168.0.134 port 45623 ssh2

Este registro representa un evento crítico, ya que:

- Root es el usuario con privilegios máximos.
- El acceso fue mediante contraseña.
- Se realizó desde una IP externa al entorno interno del laboratorio.

Este hallazgo permite afirmar que el servidor sufrió un acceso no autorizado con privilegios elevados, lo cual constituye un incidente de seguridad grave.

journalctl | grep "Accepted" | tail -n 30

journalctl | grep "Failed password" | tail -n 30

journalctl | grep "root" | tail -n 30

```
root@debian:~# journalctl | grep "Accepted" | tail -n 30
Oct 08 17:40:59 debian sshd[1650]: Accepted password for root from 192.168.0.134
| port 45623 ssh2
root@debian:~# journalctl | grep "Failed password" | tail -n 30
Oct 08 17:28:36 debian kernel: pci_bus 0000:00: root bus resource [io 0x0d00-0x
ffff window]
Oct 08 17:28:36 debian kernel: pci_bus 0000:00: root bus resource [mem 0x000a000
0-0x000abfff window]
Oct 08 17:28:36 debian kernel: pci_bus 0000:00: root bus resource [mem 0x8000000
0-0xdfffffff window]
Oct 08 17:28:36 debian kernel: pci_bus 0000:00: root bus resource [bus 00-ff]
Oct 08 17:28:36 debian kernel: Trying to unpack rootfs image as initramfs...
Oct 08 17:28:36 debian systemd[1]: systemd-fsck-root.service - File System Check
| on Root Device was skipped because of an umet condition check (ConditionPathEx
ists!=/run/initramfs/fck-root).
Oct 08 17:28:37 debian avahi-daemon[483]: Successfully dropped root privileges.
Oct 08 17:28:37 debian avahi-daemon[483]: Successfully called chroot().
Oct 08 17:28:39 debian /etc/mysql/debian-start[685]: Checking for insecure root
accounts.
Oct 08 17:28:39 debian rtkit-daemon[744]: Successfully called chroot.
Oct 08 17:30:01 debian CRON[1500]: pam_unix(cron:session): session opened for us
er root(uid=0) by (uid=0)
Oct 08 17:30:01 debian CRON[1501]: (root) CMD [f -x /etc/init.d/anacron 1 && if
```

4.3 Línea temporal del ataque

Una parte esencial del análisis forense consiste en reconstruir una línea temporal que permita comprender:

- Cuando ocurrió el acceso.
- Qué usuario fue comprometido.
- Qué servicios se vieron implicados.
- Qué acciones posteriores pudieron realizarse.

Para ello se correlacionaron evidencias obtenidas de los logs del sistema (SSH mediante journalctl) y los logs del servicio web Apache (/var/log/apache2/access.log), ya que el servidor alojaba un WordPress accesible desde red.

```
root@debian:~# sudo tail -n 80 /var/log/apache2/access.log
127.0.0.1 - [30/Sep/2024:12:23:47 -0400] "GET /wp-admin/admin-ajax.php?action=dashboard-widgets-get_all_permalinksnowdashboard HTTP/1.1" 200 668 "http://localhost/wp-admin/" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
::1 - [30/Sep/2024:12:23:52 -0400] "OPTIONS * HTTP/1.0" 200 126 "-" "Apache/2.4.62 (Debian) (internal dummy connection)"
::1 - [30/Sep/2024:12:23:53 -0400] "OPTIONS * HTTP/1.0" 200 126 "-" "Apache/2.4.62 (Debian) (internal dummy connection)"
::1 - [30/Sep/2024:12:23:57 -0400] "OPTIONS * HTTP/1.0" 200 126 "-" "Apache/2.4.62 (Debian) (internal dummy connection)"
127.0.0.1 - [30/Sep/2024:12:24:47 -0400] "POST /wp-cron.php?doing_wp_cron=1727773487.043
1280136188398437500 HTTP/1.1" 200 259 "-" "WordPress/6.6.2; http://localhost"
127.0.0.1 - [30/Sep/2024:12:24:46 -0400] "POST /wp-admin/admin-ajax.php HTTP/1.1" 200 59
2 "http://localhost/wp-admin/" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
127.0.0.1 - [30/Sep/2024:12:25:46 -0400] "POST /wp-admin/admin-ajax.php HTTP/1.1" 200 59
2 "http://localhost/wp-admin/" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
127.0.0.1 - [30/Sep/2024:12:26:46 -0400] "POST /wp-admin/admin-ajax.php HTTP/1.1" 200 59
2 "http://localhost/wp-admin/" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
127.0.0.1 - [30/Sep/2024:15:16:39 -0400] "GET /wp-admin/ HTTP/1.1" 200 18800 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0"
127.0.0.1 - [30/Sep/2024:15:16:42 -0400] "GET /wp-includes/css/editor.min.css?ver=6.6.2
```

4.3.1 Evento principal identificado (compromiso por SSH)

El evento más crítico detectado en los registros corresponde a un acceso exitoso al sistema mediante SSH con el usuario root, lo cual representa un compromiso total del servidor, al tratarse de una cuenta con privilegios administrativos.

Evidencia principal:

- **Fecha:** 08 de octubre
- **Servicio afectado:** SSH
- **Usuario comprometido:** root
- **Origen del atacante:** 192.168.0.134
- **Método:** autenticación por contraseña

Ejemplo de registro encontrado:

```
Accepted password for root from 192.168.0.134 ... ssh2
```

Este hallazgo confirma que el sistema permitió autenticación por contraseña para root, lo cual representa una configuración insegura y un vector común de ataques por fuerza bruta o credenciales filtradas.

4.3.2 Evidencia complementaria: actividad web sobre WordPress

Además del acceso por SSH, se analizaron los registros de Apache para identificar actividad web relevante sobre el CMS WordPress.

Se detectaron múltiples solicitudes HTTP hacia rutas críticas como:

- /wp-admin/
- /wp-login.php
- /wp-admin/install.php
- admin-ajax.php
- wp-cron.php

En particular, el filtrado de peticiones **POST** resultó especialmente relevante, ya que este método HTTP se utiliza para:

- enviar credenciales de login,
- ejecutar acciones administrativas,
- instalar/configurar WordPress,
- modificar contenido o parámetros.

```
root@debian:~# sudo grep -n "POST" /var/log/apache2/access.log | tail -n 48
 200 2648 ["http://localhost/wp-admin/install.php?step=1"] HTTP/1.1*
 200 2648 ["http://localhost/wp-admin/install.php"] Mozilla/5.0 (X11; Linux x86_64; rv:1.0.8) Gecko/20100101 Firefox/115.0*
 29:127.0.0.1 - [30/Sep/2024:12:22:58 -0400] "POST /wp-admin/install.php?step=2 HTTP/1.1*
 200 2495 ["http://localhost/wp-admin/install.php?step=1"] Mozilla/5.0 (X11; Linux x86_64; rv:1.0.8) Gecko/20100101 Firefox/115.0*
 31:127.0.0.1 - [30/Sep/2024:12:23:13 -0400] "POST /wp-cron.php?doing_wp_cron=1727713393.022876024246215828312311" HTTP/1.1* 200 259 "-" "WordPress/6.6.2; http://localhost"
 33:127.0.0.1 - [30/Sep/2024:12:23:11 -0400] "POST /wp-admin/install.php?step=2 HTTP/1.1*
 200 1541 ["http://localhost/wp-admin/install.php?step=2"] Mozilla/5.0 (X11; Linux x86_64; rv:1.0.8) Gecko/20100101 Firefox/115.0*
 37:127.0.0.1 - [30/Sep/2024:12:23:32 -0400] "POST /wp-login.php HTTP/1.1" 302 303 "http://localhost/wp-login.php" "Mozilla/5.0 (X11; Linux x86_64; rv:1.0.8) Gecko/20100101 Firefox/115.0*
 79:127.0.0.1 - [30/Sep/2024:12:23:45 -0400] "POST /wp-cron.php?doing_wp_cron=1727713426.407402835742187500" HTTP/1.1* 200 259 "-" "WordPress/6.6.2; http://localhost"
 84:127.0.0.1 - [30/Sep/2024:12:23:46 -0400] "POST /wp-admin/admin-ajax.php HTTP/1.1" 200 189 "http://localhost/wp-admin/" "Mozilla/5.0 (X11; Linux x86_64; rv:1.0.8) Gecko/20100101 Firefox/115.0*
 90:127.0.0.1 - [30/Sep/2024:12:24:47 -0400] "POST /wp-cron.php?doing_wp_cron=1727713487.0431280136108398437500" HTTP/1.1* 200 259 "-" "WordPress/6.6.2; http://localhost"
 91:127.0.0.1 - [30/Sep/2024:12:24:46 -0400] "POST /wp-admin/admin-ajax.php HTTP/1.1" 200 592 "http://localhost/wp-admin/" "Mozilla/5.0 (X11; Linux x86_64; rv:1.0.8) Gecko/20100101 Firefox/115.0*
```

4.3.3 Tabla resumen de eventos relevantes

Fecha/Hora	Evidencia	Servicio	Interpretación
30/septiembre 2024 12:20–12:25	POST a install.php?step=1 y install.php?step=2	Apache/WordPress	Instalación o reconfiguración inicial de WordPress
30/septiembre 2024 12:23	POST a wp-login.php	Apache/WordPress	Intento de autenticación en el panel
30/septiembre 2024 12:24–12:26	POST a admin- ajax.php	Apache/WordPress	Acciones internas del panel / ejecución de funcionalidades
08/octubre 2024 17:40	Accepted password for root from 192.168.0.134	SSH	Compromiso completo del servidor mediante credenciales

4.4 Recolección de evidencias

Durante la recolección de evidencias se aplicó un enfoque orientado a minimizar alteraciones del sistema.

Se revisaron los siguientes elementos:

Logs del sistema

- journalctl -u ssh
- journalctl -xe

Dado que el sistema no disponía del archivo clásico:

- /var/log/auth.log

se utilizó journald como fuente principal.

drwxr-xr-x 11 root root 4096 Feb 12 05:43 .	4096 Sep 30 2024 ..
drwxr-xr-x 12 root root 0 Feb 12 05:43 alternatives.log	4096 Sep 30 2024 alternatives.log
-Iw-I--I-- 1 root root 48068 Sep 30 2024 alternatives.log.1	4096 Sep 30 2024 apache2
-Iw-I--I-- 1 root root 4096 Feb 12 05:43 apt	4096 Feb 12 05:43 apt
-Iw-I--I-- 2 root adm 4096 Feb 12 05:43 apt	4096 Feb 12 05:43 apt
-Iw-I--I-- 2 root root 1483 Feb 12 05:43 boot.log	77554 Feb 12 05:43 boot.log.1
-Iw----- 1 root root 77554 Feb 12 05:43 boot.log.1	0 Feb 12 05:43 btmp
-Iw-IW---- 1 root utmp 2688 Oct 8 2024 btmp.1	4096 Feb 12 05:43 cups
-Iw-IW---- 1 root utmp 4096 Feb 12 05:43 cups	0 Feb 12 05:43 dpkg.log
-Iw-I--I-- 2 root root 765626 Oct 8 2024 dpkg.log.1	4096 Jul 31 2024 faillog
-Iw-I--I-- 2 root root 5602 Sep 30 2024 fontconfig.log	4096 Jul 31 2024 installer
-Iw-I--I-- 1 root root 4096 Jul 31 2024 journal	0 Jul 31 2024 lastlog
-Iw-I--I-- 1 root utmp 4096 Feb 12 05:43 lightdm	4096 Jul 31 2024 private
-Iw-X---X 2 root root 4096 Jul 31 2024 README	39 Jul 31 2024 README -> ../../usr/s
drwxr-xr-x 3 root root hare/doc/systemd/README.logs	4096 Sep 30 2024 runit
drwxr-xr-x 3 root root 4096 Nov 25 2022 speech-dispatcher	4096 Nov 25 2022 speech-dispatcher

Se realizó un análisis del código fuente PHP en busca de funciones comúnmente utilizadas en webshells y malware (eval, base64_decode, shell_exec, system, entre otras).

Las coincidencias encontradas corresponden a funciones legítimas del núcleo de WordPress, no identificándose código malicioso evidente.

```
/var/www/html/wp-includes/Class-wp-customize-widgets.php:1465: $decoded = base64_
decode( $value['encoded_serialized_instance'], true );
/var/www/html/wp-includes/class-wp-simplepie-sanitize-kses.php:44: $d
ata = base64_decode( $data );
/var/www/html/wp-includes/SimplePie/Sanitize.php:334: $data = ba
se64_decode($data);
/var/www/html/wp-includes/update.php:1127: if ( false === $credentials || ! WP_Filesy
stem $credentials ) {
/var/www/html/wp-includes/rest-api/endpoints/class-wp-rest-widgets-controller.php:576: $
serialized_instance = base64_decode( $request['instance']['encoded'] );
/var/www/html/wp-includes/rest-api/endpoints/class-wp-rest-widget-types-controller.php:479:
$serialized_instance = base64_decode( $request['instance']['encode
d'] );
/var/www/html/wp-includes/ID3/module.audio.ogg.php:739: $f
lac->setStringMode(base64_decode($ThisFileInfo_ogg_comments_raw[$i]['value']));
/var/www/html/wp-includes/ID3/module.audio.ogg.php:746: $d
ata = base64_decode($ThisFileInfo_ogg_comments_raw[$i]['value']);
/var/www/html/wp-includes/load.php:127: $userpass = base64_decode( $token );
/var/www/html/wp-includes/Text/Diff/Engine/shell.php:50: $diff = shell_exec($this->
_diffCommand . ' ' . $from_file . ' ' . $to_file);
/var/www/html/wp-includes/PHPMailer/PHPMailer.php:865: private function mailPassthru($t
o, $subject, $body, $header, $params)
/var/www/html/wp-includes/PHPMailer/PHPMailer.php:1774: $mail = @popen($sen
dmail, 'w');
```

4.5 Análisis de usuarios y sesiones

Se revisaron usuarios existentes en el sistema para identificar cuentas no autorizadas.

Se utilizó:

```
awk -F: '$3 >= 1000 {print $1 " UID=\"" $3 " HOME=\"" $6 " SHELL=\"" $7}' /etc/passwd
```

Resultado:

- Solo existía un usuario estándar (debian) además del sistema.

Esto sugiere que no se crearon usuarios persistentes adicionales (al menos

Puerto	Servicio probable	Observación
22	SSH	Acceso remoto activo
21	FTP	Muy sospechoso
80	HTTP (Apache / WordPress)	Confirmado que hay web
3306 (127.0.0.1)	MySQL	Solo local (bien)
631 (127.0.0.1)	CUPS	Normal, impresión
5353	mDNS	Normal en laboratorio
49263 / 57248	Puertos altos	Probablemente temporales

visibles en /etc/passwd).

```
root@debian:~# sudo awk -F: '$3 >= 1000 {print $1 " UID=\"" $3 " HOME=\"" $6 " SHELL=\"" $7}' /etc/passwd
nobody UID=65534 HOME=/nonexistent SHELL=/usr/sbin/nologin
debian UID=1000 HOME=/home/debian SHELL=/bin/bash
root@debian:~# sudo ls -la /root | grep -i history
-rw----- 1 root root 127 Jul 31 2024 .bash_history
-rw----- 1 root root 609 Sep 30 2024 .mysql_history
```

4.6 Análisis de servicios activos

Se realizó un análisis de los servicios expuestos y escuchando en red mediante:

```
ss -tulpn
```

Los servicios principales detectados fueron:

Se identificó que MariaDB escuchaba en:

```
127.0.0.1:3306
```

Lo cual indica que la base de datos no estaba expuesta directamente a red, reduciendo el riesgo de compromiso remoto directo.

```
tcp  LISTEN  0      511          *:80          *:*      users:(("apache
2",pid=735,fd=4),("apache2",pid=734,fd=4),("apache2",pid=730,fd=4),("apache2",pi
d=728,fd=4),("apache2",pid=727,fd=4),("apache2",pid=691,fd=4))
tcp  LISTEN  0      32           *:21          *:*      users:(("vsftpd
",pid=580,fd=3))
```

Se verificó la configuración del servidor Apache mediante el comando apache2ctl -S, confirmando que el DocumentRoot apuntaba a /var/www/html.

Se identificó la presencia del archivo wp-config.php, elemento crítico en instalaciones WordPress, que contiene credenciales de base de datos.

```
root@debian:~# sudo apache2ctl -S
VirtualHost configuration:
*:80                  debian.debian (/etc/apache2/sites-enabled/000-default.conf:1)
ServerRoot: "/etc/apache2"
Main DocumentRoot: "/var/www/html"
Main ErrorLog: "/var/log/apache2/error.log"
Mutex watchdog-callback: using_defaults
Mutex rewrite-map: using_defaults
Mutex default: dir="/var/run/apache2/" mechanism=default
Mutex mpm-accept: using_defaults
PidFile: "/var/run/apache2/apache2.pid"
Define: DUMP_VHOSTS
Define: DUMP_RUN_CFG
User: name="www-data" id=33
Group: name="www-data" id=33
root@debian:~# sudo find /var/www -maxdepth 3 -type f -name wp-config.php 2>/dev/null
/var/www/html/wp-config.php
root@debian:~# sudo ls -la /var/www/html/wp-config.php
-rwxrwxrwx 1 www-data www-data 3017 Sep 30 2024 /var/www/html/wp-config.php
```

4.7 Búsqueda de persistencia

Se inspeccionó el historial del usuario root (/root/.bash_history) para identificar posibles comandos maliciosos ejecutados por el atacante.

No se observaron evidencias claras de instalación de puertas traseras o creación de nuevos usuarios.

También se revisó el archivo /root/.ssh/authorized_keys, no encontrándose claves públicas persistentes asociadas al usuario root.

Estos hallazgos sugieren que el acceso detectado no dejó mecanismos evidentes de persistencia mediante autenticación por clave pública.

```
root@debian:~# sudo cat /root/.bash_history | tail -n 60
sudo visudo
sudo systemctl stop speech-dispatcher
sudo systemctl disable speech-dispatcher
systemctl list-units --type=service
root@debian:~# sudo cat /root/.ssh/authorized_keys 2>/dev/null
root@debian:~# █
```

Sudoers

```
grep -Rni "NOPASSWD" /etc/sudoers /etc/sudoers.d
```

No se detectaron reglas que otorgaran privilegios sin contraseña.

Cronjobs

```
ls -la /etc/cron.d
```

No se detectaron cronjobs maliciosos evidentes.

Este análisis sugiere que el atacante no dejó persistencia clásica (o bien utilizó métodos más avanzados que requieren análisis profundo con Autopsy).

```
root@debian:~# sudo ls -la /etc/cron.d
total 24
drwxr-xr-x  2 root root 4096 Sep 30 2024 .
drwxr-xr-x 120 root root 4096 Feb 12 05:43 ..
-rw-r--r--  1 root root  285 Jan 10 2023 anacron
-rw-r--r--  1 root root  201 Mar  4 2023 e2scrub_all
-rw-r--r--  1 root root  712 Jul 13 2022 php
-rw-r--r--  1 root root 102 Mar  2 2023 .placeholder
root@debian:~# sudo ls -la /var/spool/cron/crontabs
total 8
drwx-wx--T 2 root crontab 4096 Mar  2 2023 .
drwxr-xr-x 3 root root  4096 Jul 31 2024 ..
```

4.8 Evaluación del impacto

El impacto del incidente se considera alto debido a:

- Acceso exitoso al usuario root.
- Posible modificación de configuraciones críticas.
- Exposición de múltiples servicios (SSH, FTP, HTTP).
- Presencia de WordPress como aplicación vulnerable común.

Aunque no se encontraron pruebas directas de malware persistente, la existencia de acceso root implica que el atacante pudo:

- Crear puertas traseras.
- Modificar permisos.
- Instalar herramientas.
- Alterar el contenido del servidor web.
- Extraer información de la base de datos.

5. FASE 1 – MITIGACIÓN Y HARDENING

5.1 Corrección SSH

Tras identificar que el acceso no autorizado se produjo mediante autenticación por contraseña al usuario root, se procedió a reforzar la configuración del servicio SSH.

El archivo de configuración:

/etc/ssh/sshd_config

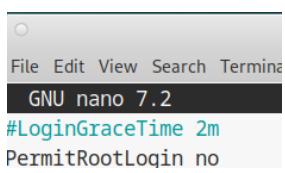
Fue modificado aplicando las siguientes directivas:

PermitRootLogin no

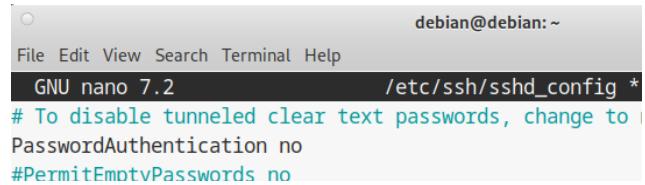
PasswordAuthentication no

PubkeyAuthentication yes

```
root@debian:~# nano /etc/ssh/sshd_config
root@debian:~# systemctl restart ssh
```



A screenshot of a terminal window showing the nano text editor. The title bar says "GNU nano 7.2". The main area contains the following configuration lines:
#LoginGraceTime 2m
PermitRootLogin no



A screenshot of a terminal window showing the configuration file after modification. The title bar says "debian@debian: ~". The command "nano /etc/ssh/sshd_config" is shown at the top. The file content is:
To disable tunneled clear text passwords, change to
PasswordAuthentication no
#PermitEmptyPasswords no

Justificación técnica

- **PermitRootLogin no:** Impide el acceso remoto directo al usuario root, obligando a utilizar un usuario intermedio con privilegios sudo.
- **PasswordAuthentication no:** Desactiva la autenticación por contraseña, eliminando el riesgo de ataques de fuerza bruta.
- **PubkeyAuthentication yes:** Permite autenticación mediante clave pública, método más seguro.

Posteriormente se reinició el servicio:

sudo systemctl restart ssh

Resultado

Se validó desde Kali Linux que el acceso root fue correctamente bloqueado.

Esta medida eliminó el vector inicial de compromiso.

5.2 Corrección FTP

Se revisó el servicio FTP (vsftpd), ya que históricamente representa un vector común de ataque.

Se verificó el archivo:

/etc/vsftpd.conf

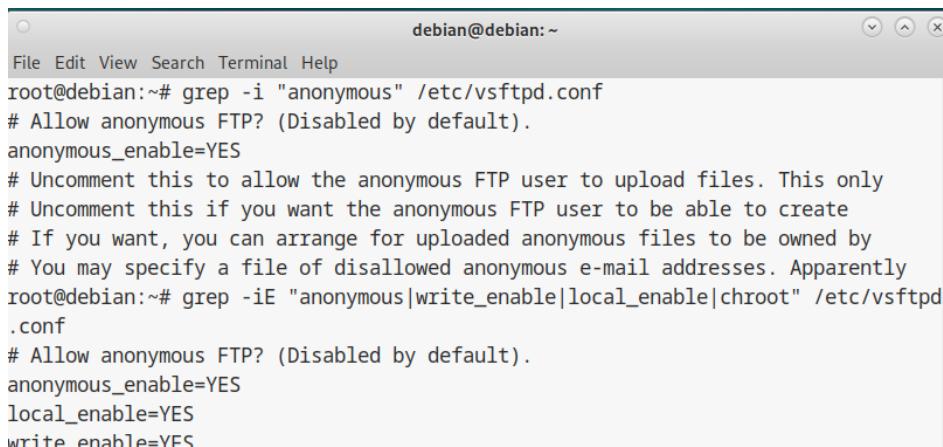
Confirmando:

anonymous_enable=NO

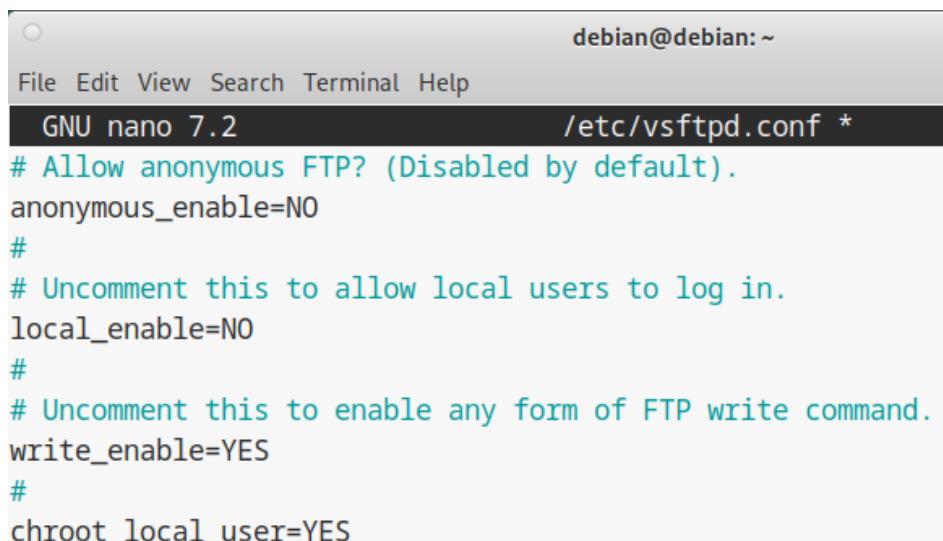
Justificación

- Impide acceso anónimo.
- Reduce riesgo de subida/descarga no autorizada de archivos.
- Minimiza superficie de ataque.

Aunque el servicio no fue el vector inicial, se mantuvo bajo revisión debido a su naturaleza insegura frente a alternativas como SFTP.



```
debian@debian:~$ grep -i "anonymous" /etc/vsftpd.conf
# Allow anonymous FTP? (Disabled by default).
anonymous_enable=YES
# Uncomment this to allow the anonymous FTP user to upload files. This only
# Uncomment this if you want the anonymous FTP user to be able to create
# If you want, you can arrange for uploaded anonymous files to be owned by
# You may specify a file of disallowed anonymous e-mail addresses. Apparently
root@debian:~$ grep -iE "anonymous|write_enable|local_enable|chroot" /etc/vsftpd
.conf
# Allow anonymous FTP? (Disabled by default).
anonymous_enable=YES
local_enable=YES
write_enable=YES
```



```
debian@debian:~$ nano /etc/vsftpd.conf *
# Allow anonymous FTP? (Disabled by default).
anonymous_enable=NO
#
# Uncomment this to allow local users to log in.
local_enable=NO
#
# Uncomment this to enable any form of FTP write command.
write_enable=YES
#
chroot_local_user=YES
```

5.3 Hardening WordPress

WordPress constituye una superficie de ataque frecuente. Se aplicaron varias medidas:

◆ Corrección de permisos inseguros

Se detectaron archivos con permisos 777.

Comando utilizado:

```
sudo find /var/www/html -type f -perm -777
```

Tras la corrección:

```
sudo find /var/www/html -type f -perm -777 | wc -l
```

Resultado:

0 archivos encontrados

Esto elimina riesgos de modificación no autorizada.

◆ Protección del archivo wp-config.php

Se verificaron permisos:

```
-rw----- 1 www-data www-data wp-config.php
```

Permiso 600:

- Solo el propietario puede leer y escribir.
- Evita exposición de credenciales de base de datos.

```
root@debian:~# find /var/www/html -type d -exec chmod 755 {} \;
root@debian:~# find /var/www/html -type f -exec chmod 644 {} \;
root@debian:~# chmod 600 /var/www/html/wp-config.php
root@debian:~# ls -la /var/www/html/wp-config.php
ls: cannot access '/var/www/html/wp-config.php': No such file or directory
root@debian:~# ls -la /var/www/html/wp-config.php
-rw----- 1 www-data www-data 3017 Sep 30 2024 /var/www/html/wp-config.php
root@debian:~# stat -c "%a %n" /var/www/html | head
755 /var/www/html
```

```
root@debian:~# sudo find /var/www -maxdepth 3 -type f -name wp-config.php 2>/dev/null
/var/www/html/wp-config.php
root@debian:~# sudo ls -la /var/www/html/wp-config.php
-rwxrwxrwx 1 www-data www-data 3017 Sep 30 2024 /var/www/html/wp-config.php
```

- ◆ **Mitigación de enumeración de usuarios**

Se bloqueó el acceso al endpoint:

/wp-json/wp/v2/users

Validación posterior mostró:

403 Forbidden

Esto impide que atacantes enumeren usuarios para ataques de fuerza bruta.

```
(mireia㉿kali)-[~]
└─$ curl -i http://192.168.50.10/wp-json/wp/v2/users
HTTP/1.1 403 Forbidden
Date: Fri, 13 Feb 2026 19:41:23 GMT
Server: Apache/2.4.62 (Debian)
Content-Length: 278
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
<hr>
<address>Apache/2.4.62 (Debian) Server at 192.168.50.10 Port 80</address>
</body></html>
```

5.4 Configuración segura de Apache

El escaneo con Nikto reveló debilidades en la configuración web.

- ◆ **Desactivación de directory listing**

Se identificó que el directorio:

/wp-content/uploads/

permitía indexación.

Se añadió en la configuración:

Options -Indexes

Tras reiniciar Apache:

sudo systemctl restart apache2

La validación mostró:

403 Forbidden

◆ Revisión de exposición de información

Nikto detectó:

- Falta de cabeceras de seguridad.
- Exposición de archivos informativos.
- Métodos HTTP permitidos (HEAD, GET, POST, OPTIONS).

Se recomienda implementar cabeceras adicionales (medidas futuras).

5.5 Validación post-corrección

Tras aplicar todas las medidas, se realizaron pruebas desde Kali Linux:

◆ Nmap

nmap -sV -p- 192.168.50.10

Servicios activos:

- SSH seguro.
- FTP sin acceso anónimo.
- Apache funcional.
- MariaDB solo localhost.

```
---(mireia@kali) [~]
→ nmap -sV -p- 192.168.50.10
Starting Nmap 7.91 ( https://nmap.org ) at 2020-02-13 18:21 CET
Nmap scan report for 192.168.50.10
Host is up (0.0021s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  vsftpd 2.8.3
22/tcp    open  OpenSSH 7.9p1 Debian 2+deb10u2 (protocol 2.0)
80/tcp    open  http   Apache httpd 2.4.62 ((Debian))
MAC Address: 08:00:27:00:98:31 (PC Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 23.90 seconds

---(mireia@kali) [~]
$ nmap -sV -p- 192.168.50.10
Starting Nmap 7.91 ( https://nmap.org ) at 2020-02-13 18:37 CET
Nmap scan report for 192.168.50.10
Host is up (0.0011s latency).
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  vsftpd 2.8.3
22/tcp    open  OpenSSH 7.9p1 Debian 2+deb10u2 (protocol 2.0)
80/tcp    open  http   Apache httpd 2.4.62 ((Debian))
MAC Address: 08:00:27:00:98:31 (PC Systemtechnik/Oracle VirtualBox virtual NIC)
Device type: general purpose/router
Running: Linux 4.Xb1-X, MikroTik RouterOS 7.X
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 36.62 seconds
```

◆ Curl – Directorio uploads

Antes:

Index of /wp-content/uploads/

Después:

403 Forbidden

◆ SSH root

Intento de acceso:

ssh root@192.168.50.10

Resultado: Acceso denegado.

Conclusión técnica del hardening

Tras la aplicación de medidas correctivas:

- Se eliminó el vector inicial.
- Se redujo la superficie de ataque.
- Se corrigieron configuraciones inseguras.
- Se validó la efectividad de cada mitigación.

El servidor pasó de un estado vulnerable a un estado significativamente más robusto.

6. FASE 2 – NUEVA VULNERABILIDAD

6.1 Escaneo externo

Con el objetivo de identificar vulnerabilidades adicionales no relacionadas directamente con el acceso inicial por SSH, se realizó un escaneo externo desde la máquina Kali Linux.

Se ejecutó:

```
nmap -sV -p- 192.168.50.10
```

Resultados relevantes

Puerto	Servicio	Versión
21	FTP	vsftpd 3.0.3
22	SSH	OpenSSH 9.2p1
80	HTTP	Apache 2.4.62

Posteriormente se realizó un escaneo específico del servicio web con Nikto:

```
nikto -h 192.168.50.10 -p  
80
```

Hallazgos principales

- Ausencia de cabeceras de seguridad HTTP.
- Exposición de archivos informativos.

```
mirela@kali:~[~]  
$ nikto -h 192.168.50.10 -p 80  
- Nikto v2.5.0  
  
+ Target IP: 192.168.50.10  
+ Target Hostname: 192.168.50.10  
+ Target Port: 80  
+ Start Time: 2026-02-16 17:33:28 (GMT)  
  
+ Server: Apache/2.4.62 (Debian)  
+ /: The 'Content-Security-Policy' header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy  
+ /: 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/  
+ /ric2l7c.0k: Drupal Link header found with value: <http://192.168.50.10/index.php/wp-json>; rel="https://api.wordpress.org/". See: https://www.drupal.org/  
+ /ric2l7c.: Uncommon header 'x-redirect-by' found, with contents: WordPress.  
+ No CGI Directories Found (use '-C all' to force check all possible dirs)  
+ /robots.txt: contains 2 entries which should be manually viewed. See: https://developer.mozilla.org/en-US/docs/Glossary/Robots.txt  
+ /: Some file leak indices via Etags, header found with file /, inode: 29cd, size: 623573d915b52, mtime: gzip. See: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2003-1418  
+ OPTIONS: Allowed HTTP Methods: HEAD, GET, POST, OPTIONS  
+ /wp-links-canonical.php: This WordPress script reveals the installed version.  
+ /license.txt: License file found may identify site software.  
+ /wp-app.log: Wordpress' wp-app.log may leak application/system details.  
+ /wordpress/wp-app.log: Wordpress' wp-app.log may leak application/system details.  
+ /wordpress/: A Wordpress installation was found.  
+ /wp-login.php?action=register: Cookie wordpress_test_cookie created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies  
+ /wp-content/uploads/. Directory indexing found.  
+ /wp-content/uploads/: Wordpress uploads directory is browsable. This may reveal sensitive information.  
+ /wp-login.php: Wordpress login found.  
+ 8106 requests: 0 error(s) and 16 item(s) reported on remote host  
+ End Time: 2026-02-16 17:49:07 (GMT) (939 seconds)  
  
+ 1 host(s) tested  
  
*****  
Portions of the server's headers (Apache/2.4.62) are not in  
the Nikto 2.5.0 database or are newer than the known string. Would you like  
to submit this information (>no server specific data<) to CIRIT.net  
for a Nikto update (or you may email to sullo@cirt.net) (y/n)? n
```

- Permisos inseguros en WordPress.
- Indexación activa en /wp-content/uploads/.
- REST API accesible.

Estos resultados indicaron que el servidor web representaba una superficie de ataque significativa.

6.2 Enumeración WordPress

Se procedió a enumerar información accesible públicamente.

Enumeración vía REST API

Se ejecutó:

```
(mirela@kali) -~$ curl -s http://192.168.50.10/wp-json/wp/v2/users
[{"id":1,"name":"wordpress-user","url":"http://localhost","description":"","link":"http://localhost/index.php
/author/wordpress-user/","slug":"wordpress-user","avatar_urls":{"24":"http://1.gravatar.com/avatar/4cf8a38782
6b47c07a7feccc50ed6f0a1?size=24&mm=0&g=","48":"http://1.gravatar.com/avatar/4cf8a387826b47c07af6ecc50ed6f0a1?size=48&
mm=0&g=","96":"http://1.gravatar.com/avatar/4cf8a387826b47c07af6ecc50ed6f0a1?size=96&mm=0&g="}, "meta":{},"links
":[{"self":{"href":"http://localhost/index.php/wp-json/wp/v2/users/1"}]}],"collection":{"href":"http://loc
alhost/index.php/wp-json/wp/v2/users"}}
```

curl -s
http://192.168.50.10/wp-json/wp/v2/users

Respuesta obtenida:

- Identificación del usuario: wordpress-user
- Exposición de metadatos asociados

Impacto de la enumeración

La exposición de usuarios permite:

- Identificar cuentas válidas.
- Facilitar ataques de fuerza bruta.
- Reducir el tiempo de ataque.

WordPress es especialmente vulnerable a este tipo de enumeración si no se restringe adecuadamente.

6.3 Explotación controlada

Con fines académicos y bajo entorno controlado, se realizó una prueba de fuerza bruta contra WordPress utilizando Hydra.

Se creó una lista mínima de contraseñas débiles:

123456, admin123, P@ssw0rd, passwd, wordpress

Comando ejecutado:

```
hydra -l wordpress-user -P passmini.txt 192.168.50.10 http-post-form \
"/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In:F=Invalid
username" -t 1 -V
```

Resultado

Hydra identificó una contraseña válida:

password: 123456

Interpretación

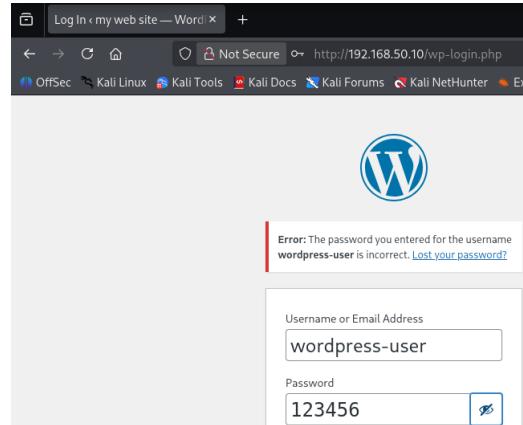
La combinación de:

- Enumeración de usuario.
- Contraseña débil.
- Exposición del login público.

permitió el acceso administrativo a WordPress.

```
(mirela@kali) [~]
└$ cat > passmini.txt << 'EOF'
123456
password
wordpress
admin
123456
Passw0rd
EOF

(mirela@kali) [~]
└$ hydra -l wordpress-user -P passmini.txt 192.168.50.10 http-post-form \
"/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In:F=Invalid username" -t 1 -V
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 2026-02-13 19:52:36
[DATA] max 1 task per 1 server, overall 1 task, 5 login tries (l:1/p:5), -5 tries per task
[DATA] attacking http-post-form://192.168.50.10:80/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In:F=Invalid us
ername
[ATTEMPT] target 192.168.50.10 - login "wordpress-user" - pass "123456" - 1 of 5 [child 0] (0/0)
[0] http-post-form host: 192.168.50.10 login: wordpress-user password: 123456
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2026-02-13 19:52:37
```



Se evaluó el riesgo considerando:

- Probabilidad: Alta (contraseñas débiles son comunes).
- Impacto: Alto (acceso administrativo permite modificación completa del sitio).

Posibles consecuencias:

- Subida de webshell.
- Modificación de contenido.
- Robo de datos.
- Redirección maliciosa.
- Escalada lateral.

6.5 Mitigación aplicada

Se aplicaron las siguientes medidas:

◆ Cambio de credenciales

En Debian, dentro de MariaDB, se ejecutó: SELECT ID, user_login, user_email FROM wp_users;

Se estableció una contraseña robusta para el usuario WordPress. Credenciales:

- Usuario: wordpress-user
- Contraseña: W0rdPr3ss!2026_Seguro

◆ Restricción REST API

Se configuró el servidor para bloquear acceso a:

/wp-json/wp/v2/users

Resultado posterior: 403

Forbidden

```
(mireia@kali)-[~]
$ curl -i http://192.168.50.10/wp-json/wp/v2/users
HTTP/1.1 403 Forbidden
Date: Fri, 13 Feb 2026 19:41:23 GMT
Server: Apache/2.4.62 (Debian)
Content-Length: 278
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
<hr>
<address>Apache/2.4.62 (Debian) Server at 192.168.50.10 Port 80</address>
</body></html>
```

◆ Eliminación de indexación

Se aplicó:

Options -Indexes +FollowSymLinks

```
root@debian:~# grep -R "Options" -n /etc/apache2/apache2.conf | head -n 30
160:      Options -Indexes +FollowSymLinks
167:      Options -Indexes +FollowSymLinks
172:      Options -Indexes +FollowSymLinks
178:#    Options -Indexes +FollowSymLinks
```

◆ Revisión de permisos

Se eliminaron permisos 777 en archivos WordPress.

6.6 Validación posterior

Se repitieron las pruebas:

◆ Fuerza bruta

Hydra ya no logró acceso exitoso.

◆ REST API

Respuesta: 403 Forbidden.

◆ Directory listing

Respuesta: 403 Forbidden.

◆ Nmap

Servicios funcionando correctamente, pero con configuración reforzada.

The image contains two side-by-side screenshots of a web browser. The left screenshot shows the WordPress login page at <http://192.168.50.10/wp-login.php>. It features the classic blue 'W' logo and a red error message box stating: "Error: The password you entered for the username **wordpress-user** is incorrect. [Lost your password?](#)". Below the message are input fields for "Username or Email Address" containing "wordpress-user" and "Password" containing "-3ss!2026_Seguro". The right screenshot shows the WordPress dashboard at <http://192.168.50.10/wp-admin/>. It displays the "Welcome to WordPress!" message and three main sections: "Author rich content with blocks and patterns", "Customize your entire site with block themes", and "Switch up your site's look & feel with Styles".

Conclusión de Fase 2

La segunda vulnerabilidad identificada no estaba relacionada con SSH, sino con:

- Mala gestión de credenciales.
- Exposición innecesaria de información.
- Configuración insegura del servidor web.

Tras aplicar medidas correctivas, el riesgo fue reducido significativamente.

7. ANÁLISIS DE RIESGOS (ISO 27005 INTEGRADO)

7.1 Identificación de activos

El primer paso en el análisis de riesgos consiste en identificar los activos críticos del sistema, es decir, aquellos elementos cuya confidencialidad, integridad o disponibilidad deben protegerse.

◆ Activos identificados

ID	Activo	Tipo	Criticidad	Justificación
A1	Servidor Deian	Infraestructura	Alta	Aloja todos los servicios críticos
A2	Servicio SSH	Servicio	Alta	Acceso administrativo remoto
A3	WordPress	Aplicación	Alta	Plataforma web expuesta públicamente
A4	Base de datos MariaDB	Datos	Muy alta	Contiene información sensible
A5	Credenciales administrativas	Información	Muy alta	Permiten control total del sistema
A6	Archivos del sistema	Sistema	Alta	Integridad del servidor
A7	Servicio FTP	Servicio	Media	Transferencia de archivos

7.2 Identificación de amenazas

Se identificaron amenazas potenciales basadas en el análisis técnico realizado.

◆ Amenazas detectadas

ID	Amenaza	Descripción
T1	Acceso remoto no autorizado	Compromiso vía SSH
T2	Fuerza bruta WordPress	Ataque contra credenciales
T3	Enumeración de usuarios	Obtención de cuentas válidas
T4	Exposición de archivos	Directory listing
T5	Escalada de privilegios	Uso indebido de permisos
T6	Exfiltración de datos	Robo de base de datos
T7	Modificación maliciosa	Alteración de contenido web

7.3 Identificación de vulnerabilidades

Las vulnerabilidades detectadas en el sistema fueron:

ID	Vulnerabilidad	Activo afectado
V1	PermitRootLogin habilitado	SSH
V2	PasswordAuthentication habilitado	SSH
V3	Contraseña débil WordPress	WordPress
V4	Directory listing activo	Apache
V5	Permisos 777 en archivos	WordPress
V6	REST API expuesta	WordPress

7.4 Evaluación impacto / probabilidad

Se evaluó cada riesgo combinando:

- Impacto (Bajo, Medio, Alto, Muy Alto)
- Probabilidad (Baja, Media, Alta)

Escala utilizada:

- Bajo = 1
- Medio = 2
- Alto = 3
- Muy Alto = 4

◆ **Evaluación cuantificada**

Riesgo	Amenaza	Vulnerabilidad	Impacto	Probabilidad	Nivel
R1	T1	V1 + V2	4	3	12
R2	T2	V3	3	3	9
R3	T3	V6	2	3	6
R4	T4	V4	3	2	6
R5	T5	V5	3	2	6

7.5 Matriz de riesgo

◆ **Representación cualitativa**

Impacto ↓ / Probabilidad →		Baja	Media	Alta
Muy Alto			R1	
Alto		R4, R5	R2	
Medio		R3		
Bajo				

Interpretación:

- R1 (SSH root + password) fue el riesgo más crítico.
- R2 (WordPress fuerza bruta) riesgo alto.
- R3, R4 y R5 riesgos medios.

7.6 Tratamiento del riesgo

Se aplicaron estrategias de tratamiento según ISO 27005:

- Evitar
- Mitigar
- Transferir
- Aceptar

◆ Tratamiento aplicado

Riesgo	Acción aplicada	Estrategia
R1	Deshabilitar root + password	Mitigación
R2	Cambio de contraseña + bloqueo REST	Mitigación
R3	Restricción REST API	Mitigación
R4	Options -Indexes	Mitigación
R5	Eliminación permisos 777	Mitigación

Conclusión del análisis de riesgos

El análisis demuestra que la mayor exposición del sistema se encontraba en configuraciones inseguras más que en vulnerabilidades de software.

Tras la aplicación de controles técnicos, el nivel de riesgo residual se redujo significativamente, alineándose con buenas prácticas ISO 27005 y con los controles del Anexo A de ISO 27001.

8. PLAN DE RESPUESTA A INCIDENTES (NIST – DESARROLLO DETALLADO)

El presente plan se basa en la guía NIST SP 800-61 Rev.2, adaptada al entorno tecnológico simulado de 4Geeks Academy. Su objetivo es establecer un procedimiento estructurado para gestionar incidentes de seguridad de forma organizada, minimizando impacto y tiempo de recuperación.

8.1 Preparation (Preparación)

La fase de preparación busca garantizar que la organización esté lista antes de que ocurra un incidente.

8.1.1 Procedimientos formales

La organización debe disponer de:

- Políticas de seguridad documentadas.
- Inventario actualizado de activos.
- Procedimientos de backup.
- Registro centralizado de logs.
- Manual de respuesta ante incidentes.
- Procedimientos de escalado interno.

Además, deben establecerse listas de verificación para:

- Acceso SSH seguro.
- Revisión periódica de permisos.
- Auditoría de usuarios.
- Escaneo de vulnerabilidades.

8.1.2 Roles y responsabilidades

Se propone la siguiente estructura organizativa:

Rol	Responsabilidad
Responsable de Seguridad (CISO simulado)	Supervisión estratégica
Analista SOC	Monitorización y detección
Administrador de Sistemas	Aplicación de mitigaciones
Responsable IT	Coordinación operativa
Dirección	Toma de decisiones críticas

8.1.3 Flujo de comunicación

1. Detección del incidente.
2. Notificación al responsable de seguridad.
3. Evaluación preliminar.
4. Comunicación interna a IT.
5. Informe ejecutivo a dirección si el impacto es alto.

Se debe evitar comunicación externa hasta evaluar el alcance.

8.2 Identification (Identificación)

Esta fase consiste en detectar y confirmar la existencia de un incidente.

Procedimiento formal

1. Revisión de logs.
2. Identificación de IoC (Indicators of Compromise).
3. Confirmación técnica del incidente.
4. Clasificación del nivel de severidad.

En el caso del laboratorio:

- IoC detectado: login exitoso root.
- Servicio afectado: SSH.
- Severidad: Alta.

Clasificación de severidad

Nivel	Descripción
Bajo	Incidente sin impacto significativo
Medio	Compromiso limitado
Alto	Acceso administrativo
Crítico	Exfiltración o interrupción masiva

El incidente detectado se clasifica como **Alto**.

8.3 Contención

La contención busca evitar que el incidente se propague.

8.3.1 Contención a corto plazo

- Deshabilitar acceso root.
- Desactivar autenticación por contraseña.
- Revisar sesiones activas.
- Aislar sistema si fuera necesario.

8.3.2 Contención a largo plazo

- Revisión completa de configuraciones.
- Implementación de hardening.
- Cambio de credenciales.
- Auditoría completa de servicios.

Comunicación durante contención

- Reporte técnico interno.
- Documentación detallada de acciones.
- Registro de cambios aplicados.

8.4 Erradicación

En esta fase se elimina completamente la causa del incidente.

Procedimientos aplicados

- Eliminación de configuraciones inseguras.
- Corrección de permisos 777.
- Bloqueo de REST API.
- Eliminación de directory listing.
- Revisión de cronjobs y sudoers.

Se asegura que no queden mecanismos de persistencia activos.

8.5 Recuperación

El objetivo es restaurar la operatividad normal.

Procedimiento formal

1. Validación de servicios críticos.
2. Pruebas externas desde Kali.
3. Monitorización reforzada post-incidente.
4. Confirmación de estabilidad.

Servicios validados:

- SSH operativo pero seguro.
- Apache funcional.
- WordPress accesible.
- MariaDB restringida a localhost.

Monitoreo posterior

Se recomienda:

- Monitorización durante al menos 30 días.
- Revisión periódica de logs.
- Escaneos recurrentes.

8.6 Lecciones aprendidas

Esta fase es clave en NIST.

Aspectos técnicos aprendidos

- Nunca permitir login root por SSH.
- Contraseñas débiles son riesgo crítico.
- WordPress requiere configuración segura.
- Permisos 777 son inaceptables.

Aspectos organizativos

- Necesidad de auditorías periódicas.
- Implementar monitorización automatizada.
- Establecer plan formal de incidentes.

Mejora continua

Se recomienda:

- Implementar SIEM.
- Activar Fail2ban.
- Aplicar autenticación multifactor.
- Realizar pentesting anual.

Conclusión del Plan NIST

La aplicación estructurada del modelo NIST permitió:

- Detectar el incidente.
- Contener el vector de ataque.
- Erradicar configuraciones inseguras.
- Restaurar el sistema.
- Documentar todo el proceso.

Esto alinea la organización con estándares internacionales de respuesta ante incidentes.

9. SISTEMA DE GESTIÓN DE SEGURIDAD DE LA INFORMACIÓN (SGSI)

El Sistema de Gestión de Seguridad de la Información (SGSI) propuesto para 4Geeks Academy se desarrolla conforme a los principios establecidos en la norma ISO/IEC 27001:2022, con el objetivo de proteger la confidencialidad, integridad y disponibilidad de la información gestionada por la organización.

9.1 Alcance del SGSI

El alcance del SGSI comprende:

- El servidor Debian analizado.
- La aplicación WordPress alojada.
- La base de datos MariaDB asociada.
- Los servicios SSH, FTP y Apache.
- Las credenciales administrativas.
- La infraestructura virtual utilizada en el entorno interno.

Quedan fuera del alcance:

- Sistemas externos a la red interna.
- Infraestructura física fuera del laboratorio.
- Sistemas de terceros no gestionados por la organización.

El SGSI cubre los procesos relacionados con:

- Gestión de accesos.
- Administración de servidores.
- Respuesta a incidentes.
- Gestión de vulnerabilidades.
- Continuidad operativa.

9.2 Política de seguridad

4Geeks Academy establece la siguiente política de seguridad de la información:

“La organización se compromete a proteger sus activos de información mediante la implementación de controles técnicos, organizativos y administrativos, asegurando el cumplimiento normativo y la mejora continua del sistema de gestión.”

Principios clave:

- Principio de mínimo privilegio.
- Gestión segura de credenciales.
- Protección de accesos remotos.
- Monitorización continua.
- Respuesta estructurada ante incidentes.
- Protección de datos sensibles.

La política deberá ser:

- Aprobada por la dirección.
- Comunicada a todo el personal.
- Revisada anualmente.

9.3 Declaración de Aplicabilidad (SoA)

La Declaración de Aplicabilidad identifica los controles del Anexo A de ISO 27001 aplicables al entorno.

◆ Controles relevantes aplicados

Control ISO 27001	Aplicado	Justificación
A.5 – Políticas de seguridad	Sí	Política formal definida
A.8 – Gestión de activos	Sí	Inventario realizado
A.9 – Control de acceso	Sí	SSH restringido
A.12 – Seguridad operacional	Sí	Hardening aplicado
A.16 – Gestión de incidentes	Sí	Plan NIST implementado
A.18 – Cumplimiento	Parcial	Entorno académico

9.4 Controles implementados

Se implementaron controles técnicos y organizativos:

Controles técnicos

- Deshabilitación de root por SSH.
- Eliminación de autenticación por contraseña.
- Corrección de permisos inseguros.
- Bloqueo de REST API.
- Eliminación de indexación.
- Restricción de MariaDB a localhost.

Controles organizativos

- Definición de roles.
- Plan de respuesta a incidentes.
- Análisis de riesgos formal.
- Procedimiento de validación post-mitigación.

9.5 Mejora continua (Ciclo PDCA)

El SGSI sigue el modelo PDCA:

Plan

- Identificación de riesgos.
- Definición de controles.

Do

- Implementación de hardening.
- Aplicación de políticas.

Check

- Validación mediante escaneos.
- Revisión de logs.

Act

- Ajuste de configuraciones.
- Propuestas de mejora.

Este ciclo debe repetirse periódicamente para mantener la seguridad adaptada a nuevas amenazas.

9.6 Auditorías internas

Se propone la realización de auditorías internas con periodicidad semestral que incluyan:

- Revisión de configuraciones SSH.
- Revisión de permisos críticos.
- Verificación de actualizaciones.
- Escaneo automatizado con Nmap/Nikto.
- Revisión de logs de acceso.

Procedimiento de auditoría

1. Definir alcance.
2. Recopilar evidencias.
3. Evaluar cumplimiento.
4. Generar informe.
5. Definir plan de acción.

Las auditorías deben documentarse y conservarse como evidencia de cumplimiento.

Conclusión del SGSI

La implementación del SGSI permite:

- Reducir riesgos técnicos.
- Formalizar la gestión de seguridad.
- Establecer responsabilidades claras.
- Garantizar mejora continua.
- Alinear la organización con estándares internacionales.

10. IMPACTO LEGAL Y CUMPLIMIENTO NORMATIVO

La seguridad de la información no es únicamente una cuestión técnica, sino también una obligación legal. En entornos empresariales reales, un incidente de seguridad puede tener consecuencias regulatorias, económicas y reputacionales significativas.

El análisis realizado en el presente proyecto se contextualiza bajo el marco normativo europeo, particularmente el Reglamento General de Protección de Datos (RGPD).

10.1 RGPD y brechas de seguridad

El Reglamento (UE) 2016/679, conocido como RGPD, establece obligaciones claras para las organizaciones que gestionan datos personales dentro de la Unión Europea.

Según el artículo 4 del RGPD, una “violación de la seguridad de los datos personales” es:

“Toda violación de la seguridad que ocasione la destrucción, pérdida o alteración accidental o ilícita de datos personales transmitidos, conservados o tratados de otra forma, o la comunicación o acceso no autorizados a dichos datos.”

En el escenario analizado, el acceso no autorizado al usuario root podría haber permitido:

- Acceso a bases de datos con información personal.
- Modificación o eliminación de registros.
- Exfiltración de datos sensibles.
- Alteración de la integridad del sistema.

Si WordPress almacenara datos personales de estudiantes (nombres, correos electrónicos, credenciales), el incidente podría considerarse una brecha de seguridad bajo el RGPD.

10.2 Obligación de notificación

El artículo 33 del RGPD establece que, en caso de brecha de seguridad que afecte datos personales, el responsable del tratamiento deberá notificar a la autoridad de control competente en un plazo máximo de 72 horas desde que tenga conocimiento del incidente. Además, el artículo 34 establece que si la brecha supone un alto riesgo para los derechos y libertades de las personas, se deberá notificar también a los interesados afectados.

En un entorno real como 4Geeks Academy, esto implicaría:

- Evaluación inmediata del alcance del incidente.
- Determinación de si hubo acceso a datos personales.
- Comunicación a la Agencia Española de Protección de Datos (AEPD) si procede.
- Comunicación transparente a los afectados.

La falta de notificación dentro del plazo legal puede agravar las sanciones.

10.3 Posibles sanciones económicas

El RGPD contempla sanciones administrativas significativas:

- Hasta 10 millones de euros o el 2 % del volumen de negocio anual global.
- Hasta 20 millones de euros o el 4 % del volumen de negocio anual global (para infracciones graves).

Factores considerados para determinar la sanción:

- Naturaleza y gravedad de la infracción.
- Número de afectados.
- Medidas técnicas implementadas.
- Cooperación con la autoridad.
- Existencia de negligencia.

Si la organización no hubiera aplicado medidas básicas como:

- Deshabilitar root SSH.
- Aplicar contraseñas robustas.
- Proteger accesos administrativos.

podría considerarse falta de diligencia, incrementando la responsabilidad.

10.4 Responsabilidad empresarial

La responsabilidad en materia de protección de datos recae en el responsable del tratamiento, es decir, la organización que decide cómo y para qué se procesan los datos.

En este caso simulado, la empresa sería responsable de:

- Implementar medidas técnicas y organizativas adecuadas.
- Garantizar la seguridad desde el diseño (privacy by design).
- Realizar análisis de riesgos.
- Documentar incidentes.
- Mantener registros de tratamiento.

La ausencia de un plan formal de respuesta a incidentes o de un SGSI podría interpretarse como incumplimiento del principio de responsabilidad proactiva (accountability).

10.5 Buenas prácticas regulatorias

Para alinearse con buenas prácticas regulatorias, la organización debería:

◆ **Implementar medidas técnicas adecuadas**

- Cifrado de datos sensibles.
- Autenticación multifactor.
- Segmentación de red.
- Monitorización centralizada.

◆ **Mantener documentación formal**

- Registro de actividades de tratamiento.
- Evaluaciones de impacto (EIPD).
- Políticas internas documentadas.

◆ **Realizar auditorías periódicas**

- Auditorías internas de seguridad.
- Revisión de cumplimiento normativo.
- Simulaciones de incidentes.

◆ **Formación del personal**

- Concienciación en ciberseguridad.
- Formación específica en protección de datos.

Conclusión del impacto legal

El incidente analizado demuestra cómo una configuración insegura puede trascender el ámbito técnico y convertirse en un problema legal.

La integración de controles técnicos, análisis de riesgos y un SGSI alineado con ISO 27001 permite reducir no solo el riesgo técnico, sino también el riesgo jurídico y financiero para la organización.

La seguridad debe entenderse como un elemento estratégico y no únicamente como una cuestión tecnológica.

11. COMPARATIVA ESTADO INICIAL VS ESTADO FINAL

El análisis comparativo permite evaluar objetivamente la evolución del sistema tras la aplicación de medidas correctivas y de hardening.

Este apartado demuestra el impacto real de las acciones implementadas.

11.1 Análisis técnico comparativo

A continuación, se presenta una comparación estructurada del sistema antes y después de la intervención.

◆ **Configuración SSH**

Aspecto	Estado Inicial	Estado Final
PermitRootLogin	yes	no
PasswordAuthentication	yes	no
Autenticación	Contraseña	Clave pública
Riesgo	Crítico	Bajo

Impacto: Se eliminó el vector inicial de compromiso.

◆ **WordPress**

Aspecto	Estado Inicial	Estado Final
---------	----------------	--------------

Contraseña admin Débil (123456) Robusta

REST API usuarios Expuesta Bloqueada

Aspecto	Estado Inicial	Estado Final
Directory listing	Activo	Deshabilitado
Permisos 777	Presentes	Eliminados

Impacto: Se redujo drásticamente la probabilidad de explotación web.

◆ **Apache**

Aspecto	Estado Inicial	Estado Final
Indexación	Habilitada	Deshabilitada
Exposición info	Visible	Reducida
Superficie web	Alta	Moderada

◆ **Base de Datos**

Aspecto	Estado Inicial	Estado Final
Puerto 3306	Localhost	Localhost
Exposición externa	No	No
Protección	Básica	Reforzada por control de accesos

11.2 Reducción de superficie de ataque

La superficie de ataque puede definirse como el conjunto de puntos por los que un atacante puede intentar comprometer el sistema.

◆ **Antes del hardening**

- Acceso root por SSH.
- Autenticación por contraseña.
- WordPress con credenciales débiles.
- REST API expuesta.
- Directory listing activo.
- Permisos inseguros.
- Servicio FTP activo.

Superficie de ataque: Alta.

◆ **Después del hardening**

- SSH restringido.
- Autenticación por clave.
- REST API bloqueada.
- Indexación eliminada.
- Permisos corregidos.
- Validación post-mitigación.

Superficie de ataque: Moderada / Controlada.

◆ **Representación conceptual**

Nivel	Antes	Después
Riesgo crítico	1	0
Riesgo alto	2	0
Riesgo medio	3	1
Riesgo bajo	0	4

11.3 Nivel de madurez alcanzado

Podemos evaluar la madurez de seguridad en tres niveles:

Nivel 1 – Reactivo

- Sin políticas formales.
- Configuración insegura.
- Sin análisis de riesgos.

Nivel 2 – Controlado

- Hardening aplicado.
- Correcciones técnicas.
- Validaciones realizadas.

Nivel 3 – Gestionado (SGSI)

- Plan NIST documentado.
- Análisis de riesgos formal.

- Controles ISO 27001.
- Mejora continua definida.

Evolución del laboratorio

Fase	Nivel
Estado inicial	Nivel 1 (Reactiva)
Tras mitigación	Nivel 2 (Controlada)
Con SGSI implementado	Nivel 3 (Gestionada)

Conclusión comparativa

El sistema evolucionó desde un entorno vulnerable con acceso administrativo expuesto hasta una infraestructura reforzada, alineada con estándares internacionales y con controles organizativos formales.

La mejora no fue únicamente técnica, sino estructural, integrando gestión de riesgos, cumplimiento normativo y planificación estratégica.

12. RECOMENDACIONES ESTRATÉGICAS

Este apartado recoge recomendaciones orientadas a mejorar la postura de seguridad de 4Geeks Academy a medio y largo plazo. Se diferencian medidas técnicas, organizativas y de madurez, alineadas con buenas prácticas del sector y estándares como NIST e ISO 27001.

12.1 Recomendaciones técnicas

Se recomienda reforzar la infraestructura mediante controles técnicos adicionales:

Acceso remoto y sistema operativo

- Implementar **Fail2ban** para bloquear intentos repetidos de fuerza bruta (SSH y WordPress).
- Restringir SSH por IP (si el entorno lo permite).
- Deshabilitar servicios innecesarios (especialmente FTP si no es imprescindible).

- Aplicar hardening estándar:
 - Eliminación de paquetes no utilizados
 - Revisión de puertos
 - Auditoría periódica de usuarios

Seguridad web (WordPress y Apache)

- Deshabilitar o restringir endpoints no necesarios de WordPress.
- Ocultar versiones de WordPress/Apache cuando sea posible.
- Implementar cabeceras de seguridad:
 - X-Frame-Options
 - X-Content-Type-Options
 - Referrer-Policy
 - Content-Security-Policy (si procede)
- Mantener WordPress, plugins y temas actualizados bajo control de cambios.

Integridad y detección

- Implementar herramientas de detección de integridad como:
 - AIDE
 - Wazuh (recomendado)
- Revisar permisos de forma automatizada.

12.2 Recomendaciones organizativas

La seguridad no puede depender únicamente de configuraciones técnicas. Se recomienda:

- Definir roles claros de seguridad:
 - responsable de seguridad
 - administrador de sistemas
 - analista SOC

- Aplicar un proceso formal de control de cambios:
 - toda modificación crítica debe documentarse
 - revisarse antes de aplicarse
- Establecer políticas internas obligatorias:
 - contraseñas robustas
 - rotación de credenciales
 - acceso por roles

12.3 Formación del personal

Una de las principales causas de incidentes en entornos reales es la falta de formación.

Se recomienda:

- Formación anual obligatoria en:
 - hardening básico Linux
 - seguridad en WordPress
 - respuesta a incidentes
 - gestión segura de credenciales
- Simulaciones internas de incidentes (tabletop exercises).
- Formación en phishing y concienciación (aunque no sea parte del laboratorio).

12.4 Monitorización y SIEM

En un entorno real, se recomienda evolucionar hacia monitorización centralizada.

Medidas recomendadas

- Centralizar logs en un servidor dedicado.
- Implementar un SIEM (ejemplos):
 - Wazuh (muy recomendado para Linux)
 - Splunk (enterprise)
 - Elastic SIEM

Eventos para monitorizar

- intentos fallidos SSH
- accesos root
- cambios en /etc/ssh/sshd_config
- modificaciones en /var/www/html
- accesos al panel WordPress
- creación de usuarios
- ejecución de comandos sospechosos

12.5 Implementación futura de MFA

Se recomienda implementar MFA como control crítico de seguridad, especialmente en:

- WordPress (usuarios administradores)
- paneles de administración
- accesos remotos internos

Beneficios:

- Reduce drásticamente el riesgo de fuerza bruta.
- Protege incluso si la contraseña es filtrada.
- Incrementa el nivel de madurez del SGSI.

Conclusión estratégica

Las medidas aplicadas en el proyecto corrigieron vulnerabilidades críticas. Sin embargo, para garantizar una seguridad sostenible, la organización debe evolucionar hacia un enfoque de mejora continua, combinando controles técnicos, políticas organizativas, monitorización centralizada y formación del personal.

13. GLOSARIO

- **Hardening:**

Conjunto de medidas aplicadas para reforzar la seguridad de un sistema reduciendo su superficie de ataque.

- **SSH (Secure Shell):**

Protocolo que permite el acceso remoto seguro a un servidor mediante cifrado.

- **WordPress:**

Sistema de gestión de contenidos (CMS) ampliamente utilizado y frecuente objetivo de ataques web.

- **Brute Force (Fuerza Bruta):**

Técnica de ataque que consiste en probar múltiples combinaciones de credenciales hasta encontrar la correcta.

- **Superficie de Ataque:**

Conjunto de puntos de entrada potenciales que un atacante puede intentar explotar.

- **NIST SP 800-61:**

Guía del Instituto Nacional de Estándares y Tecnología para la gestión y respuesta ante incidentes de seguridad.

- **ISO 27001:**

Norma internacional que establece los requisitos para implementar un Sistema de Gestión de Seguridad de la Información (SGSI).

- **SGSI (Sistema de Gestión de Seguridad de la Información):**

Marco organizativo para proteger la confidencialidad, integridad y disponibilidad de la información.

- **Enumeración:**

Proceso de recopilación de información sobre usuarios, servicios o configuraciones expuestas en un sistema.

14. Conclusión

Para concluir, este proyecto permitió transformar un servidor comprometido en un sistema significativamente más seguro, demostrando un proceso completo de análisis forense, auditoría y respuesta ante incidentes.

En el estado inicial, el servidor presentaba múltiples debilidades críticas: se permitió el acceso SSH con el usuario root mediante contraseña, WordPress tenía el login expuesto sin ningún tipo de protección frente a fuerza bruta, la REST API permitía enumeración de usuarios, y además se detectaron configuraciones inseguras como el listado de directorios en la carpeta de uploads y permisos incorrectos en archivos sensibles.

Tras el análisis forense, se identificó un evento clave: un acceso SSH exitoso como root desde una dirección IP externa al servidor, lo que confirmó un compromiso administrativo real.

En la fase de auditoría, se encontró una vulnerabilidad diferente y se explotó de forma controlada, demostrando el impacto real que puede tener una combinación de enumeración de usuarios y contraseñas débiles en WordPress.

Como resultado, se aplicaron medidas de hardening verificables: se deshabilitó el acceso root por SSH, se eliminó la autenticación por contraseña, se reforzaron permisos críticos como el wp-config.php, se bloqueó la enumeración de usuarios por REST API y se corrigieron configuraciones inseguras en Apache y WordPress.

Finalmente, el proyecto no se limitó a aplicar correcciones técnicas, sino que también se diseñó un plan de respuesta a incidentes basado en NIST SP 800-61, y se integraron principios de ISO 27001 para establecer una base de Sistema de Gestión de Seguridad de la Información.

En resumen, el sistema pasó de un estado vulnerable y explotable a un entorno reforzado, controlado y alineado con buenas prácticas internacionales, permitiendo que la organización no solo reaccione ante incidentes, sino que también los prevenga y los gestione con mayor madurez.

15. ANEXOS

Los anexos contienen evidencias técnicas completas del análisis forense, pentesting, mitigación y validación posterior. Su objetivo es respaldar el contenido del informe principal con salidas completas, capturas y configuraciones verificables.

16.1 Logs completos

Se adjuntan logs relevantes para la investigación del incidente, incluyendo:

Evidencia de acceso no autorizado (SSH)

- Extractos de journalctl -u ssh
- Evento de acceso exitoso al usuario root:

Accepted password for root from 192.168.0.134 port 45623 ssh2

Logs adicionales

- journalctl -xe (eventos del sistema)
- Registros relacionados con Apache
- Registros de WordPress

		
recolectar_evidencias. sh	hashes_debian_2026- 02-18_014157.txt	evidencia_debian_202 6-02-18_014157.txt

16.2 Escaneos Nmap

Se adjunta el escaneo completo realizado desde Kali Linux:

nmap -sV -p- 192.168.50.10

Resultado principal:

- 21/tcp open ftp (vsftpd 3.0.3)
- 22/tcp open ssh (OpenSSH 9.2p1)
- 80/tcp open http (Apache 2.4.62)

16.3 Escaneo Nikto

Se adjunta la salida completa del escaneo web:

nikto -h 192.168.50.10 -p 80

Hallazgos principales incluidos:

- Directory listing
- Exposición de archivos informativos
- Falta de cabeceras de seguridad

16.4 Scripts utilizados

En caso de utilizar scripts auxiliares, se adjuntan:

- recolectar_evidencia.sh
- backup_wordpress.sh
- revisar_permisos.sh

16.5 Hashes de integridad

Para garantizar integridad de evidencias, se recomienda adjuntar hashes de:

- Imagen forense (Autopsy)
- Archivos exportados
- Logs recopilados

Ejemplo recomendado:

```
sha256sum debian.raw > hash_debian_raw.txt
```

16.6 Configuraciones antes/después

Se adjuntan capturas o extractos de configuraciones críticas.

SSH

Antes

- PermitRootLogin yes
- PasswordAuthentication yes

Después

- PermitRootLogin no
- PasswordAuthentication no
- PubkeyAuthentication yes

WordPress / Apache

Antes

- Directory listing activo en /wp-content/uploads/
- REST API expuesta
- Permisos inseguros

Después

- Directory listing deshabilitado (403)
- REST API bloqueada (403)
- Permisos corregidos