

# Proyecto Final de Ciberseguridad 4Geeks Academy

💡 Informe Forense y Pentesting — Caso de Incidente en Servidor Debian en 4Geeks

Alumno: Simón A. Cervantes Martínez

Fecha última revisión: 14/07/2025



# Índice

## 1. Introducción

- Objetivo general del proyecto
- Contexto (simulación de empresa 4Geeks)
- Alcance del análisis
- Enfoque profesional asumido (SOC, Blue Team, análisis forense, etc.)
- Herramientas utilizadas (Kali Linux, Nmap, Autopsy, chkrootkit, etc.)

## 2. Metodología de análisis

- Enfoque técnico seguido (pasos NIST / ISO / ciclo forense)
- Fases: Reconocimiento, análisis, mitigación, hardening
- Consideraciones éticas y forenses (no alterar pruebas, documentación constante)

## 3. Fase 1: Corrección del hackeo

### 3.1 Identificación del incidente

- Servicios comprometidos
- Vía de entrada (logins sospechosos, logs revisados)
- Línea de tiempo del ataque

### 3.2 Recolección de evidencia

- Logs clave analizados (auth.log, syslog)
- Procesos sospechosos
- Cuentas de usuario maliciosas
- Archivos modificados
- Comandos clave usados

### 3.3 Escaneo y eliminación de malware

- Herramientas: chkrootkit, rkhunter, debsums, etc.
- Resultados del escaneo

### 3.4 Mitigación del ataque

- Exploit bloqueado

- Usuarios eliminados
- Backdoors desactivados
- Servicios detenidos temporalmente

### **3.5 Fortalecimiento inmediato**

- Actualización de paquetes
- Cambio de contraseñas
- Configuración de firewall y SSH

### **3.6 Evidencia de mitigación**

- Capturas o logs de configuración corregida
- Servicios reiniciados y validados

## **4. Fase 2: Detección y corrección de nueva vulnerabilidad**

### **4.1 Escaneo completo del sistema**

- Escaneo con nmap, nikto, linpeas, etc.
- Vulnerabilidades encontradas (puertos abiertos, servicios mal configurados)

### **4.2 Explotación controlada**

- Descripción detallada de la explotación (comandos, scripts, vectores)
- Pruebas de escalación (si aplica)
- Evidencia (salida de comandos, logs, capturas)

### **4.3 Corrección aplicada**

- Cambio de configuraciones
- Cierre de puertos
- Validación post-corrección

### **4.4 Validación de seguridad**

- Resultado tras parcheo y nueva revisión
- Checklist antes/después de hardening

## **5. Fase 3: Plan de respuesta a incidentes y SGSI**

### **5.1 Plan de respuesta a incidentes (NIST SP 800-61)**

- **Identificación:** Cómo detectar futuros incidentes
- **Contención:** Procedimientos para aislar el sistema
- **Erradicación:** Eliminación del ataque y su rastro
- **Recuperación:** Restauración y continuidad operativa
- **Lecciones aprendidas:** Cómo evitar la recurrencia

### **5.2 Protección de datos**

- Respaldo regular de sistemas
- Cifrado de datos sensibles
- Control de accesos (MFA, roles, auditoría)

### **5.3 Sistema de Gestión de Seguridad de la Información (SGSI – ISO 27001)**

- Análisis de riesgos
- Políticas de seguridad definidas
- Medidas de protección (firewall, acceso, políticas de actualización)
- Planes de mejora continua

## **6. Conclusiones generales**

- Estado inicial vs. estado final del sistema
- Impacto del trabajo realizado
- Lecciones técnicas aprendidas

## **7. Recomendaciones**

- Medidas futuras (2FA, SIEM, backups externos)
- Capacitación al personal
- Escaneos regulares automatizados

## **8. Glosario**

<b>Término</b>	<b>Definición</b>
Rootkit	Conjunto de herramientas usadas para ocultar procesos maliciosos
NIST	Instituto Nacional de Estándares y Tecnología (EE. UU.)
SGSI	Sistema de Gestión de Seguridad de la Información

## **9. Referencias**

- Guía NIST SP 800-61
- ISO 27001
- Documentación de herramientas: rkhunter, chkrootkit, nmap, etc.

## **10. Anexos**

- Salidas completas de herramientas (nmap, chkrootkit)
- Scripts utilizados (recolectar\_evidencia.sh, clonar\_disco.sh)
- Capturas de pantalla
- Archivos comprimidos (evidencia.tar.gz, imagen.dd)

## 1. Introducción

- Objetivo general del proyecto

El objetivo principal de este proyecto es **analizar, corregir y asegurar** un servidor Debian comprometido en un entorno simulado de la empresa 4Geeks Academy. El ejercicio pretende poner en práctica habilidades de **análisis forense, pentesting, y gestión de ciberseguridad** para restaurar el funcionamiento seguro del sistema.

- **Restaurar y proteger un servidor comprometido** (Debian).
- Simula un entorno empresarial en 4Geeks Academy.
- Fases: Análisis forense del incidente, búsqueda de nuevas vulnerabilidades, y plan de respuesta conforme a normas (NIST / ISO 27001).

- Contexto (simulación de empresa 4Geeks)

El servidor Debian simula un servidor de producción utilizado por 4Geeks Academy para alojar servicios críticos como bases de datos, FTP, y un sitio web WordPress. Se sospecha que ha sido víctima de un ciberataque, lo que ha generado la necesidad de una investigación exhaustiva, corrección de vulnerabilidades y fortalecimiento general del sistema.

- Alcance del análisis

**Vulnerabilidades a revisar específicamente (por instrucciones para profesores):**

### 1. MySQL

- Usuario con **contraseña débil**.
- Solución esperada: uso de contraseñas fuertes, restricción de accesos.

### 2. Servidor FTP

- Permisos inseguros / **acceso anónimo habilitado**.
- Solución: desactivar acceso anónimo o reforzar la configuración.

### 3. SSH

- Autenticación débil o acceso root habilitado.
- Solución: deshabilitar root login, forzar autenticación por clave pública.

### 4. Puertos abiertos innecesarios

- Escaneo con **nmap**.
- Solución: cerrar puertos y desactivar servicios innecesarios.

## 5. wp-config.php

- **Permisos mal configurados.**
  - Solución: aplicar chmod 600.

## 6. Directorio web listable

- Apache/Nginx con Options Indexes habilitado.
- Solución: deshabilitar con .htaccess o ajustes en apache2.conf.

- 
- Enfoque profesional asumido (SOC, Blue Team, análisis forense, etc.)

Rol asumido: **Blue Team / SOC Analyst / Forense Digital**

Se adopta un enfoque profesional orientado al análisis forense y la defensa del sistema, centrado en la recolección de evidencia, mitigación de amenazas, y protección proactiva.

- Herramientas utilizadas:

<b>Herramienta</b>	<b>Sistema</b>	<b>Uso principal</b>
nmap	Kali	Escaneo de red y servicios
chkrootkit	Debian	Detección de rootkits
rkhunter	Debian	Revisión de rootkits y backdoors
debsums	Debian	Verificación de integridad de paquetes
grep, journalctl	Debian	Análisis de logs
netstat, ss	Debian	Detección de conexiones y puertos activos
autopsy	Kali (opcional)	Ánalisis de imagen forense (si se genera)
hydra, wpscan	Kali	Pentesting de servicios
fail2ban, ufw	Debian	Fortalecimiento de seguridad

Este documento ofrece una guía estructurada para ejecutar el proyecto final de ciberseguridad de 4Geeks Academy, dividido en tres fases clave:

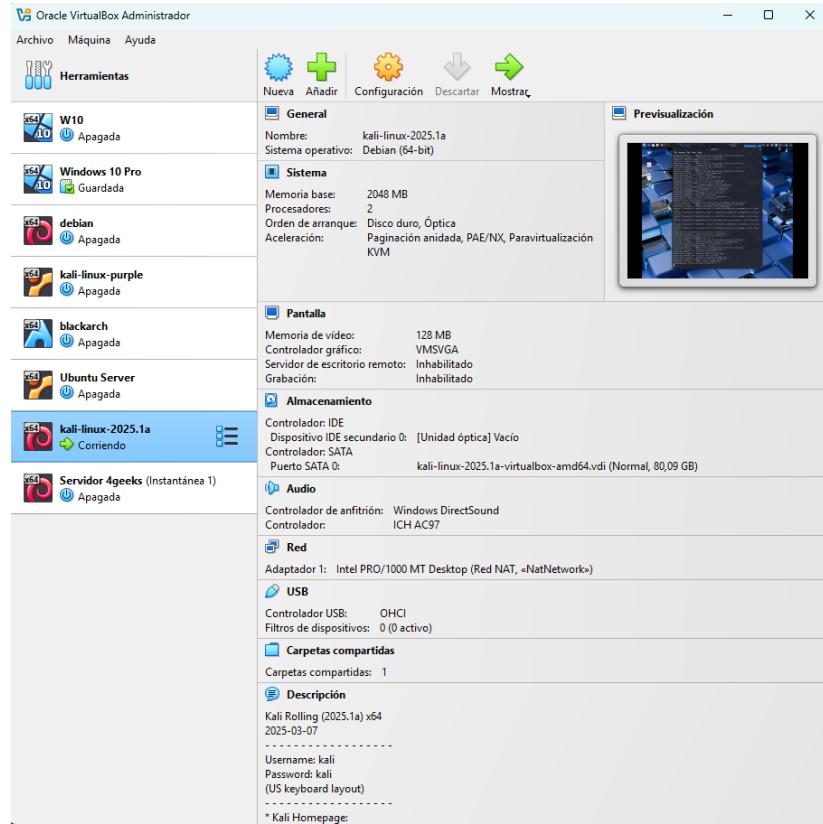
**Fase 1 (Corrección de un hackeo),**

**Fase 2 (Pentesting de una nueva vulnerabilidad) y**

**Fase 3 (Plan de respuesta a incidentes e implementación de SGSI).**

Se detallan los pasos a seguir en cada fase, con evidencias gráficas, herramientas recomendadas, referencias teóricas y entregables esperados. Cada fase concluye con la elaboración de informes profesionales (forense, pentest, plan de recuperación) y una presentación ejecutiva para comunicar resultados.

Para comenzar usaremos dos máquinas virtuales en nuestro Oracle VirtualBox Administrador, (**Kali-linux-2025.1a**) que hará la función de nuestra máquina anfitrión para esclarecer lo pasado en el servidor 4geeks infectado (**Servidor 4geeks**).



Hemos exportado la máquina debian, **Servidor 4geeks** y sacada una Instantánea para tener un punto de retorno y siempre el estado 0 de la máquina, tal como se nos entregó.

En cuanto a nuestro Kali-linux usaremos la última versión y lo primero que haremos es actualizar los paquetes y herramientas para que el sistema este preparado.

```
sudo apt-get dist-upgrade
```

```
sudo apt update && sudo apt full-upgrade -y && sudo apt autoremove -y
```

para ver la versión del sistema usamos y su ip en un red nat con nuestro otro equipo:

```
cat /etc/issue
```

```
grep VERSION /etc/os-release
```

```
uname -a
```

```
ip a
```

```

kali@kali: ~
File Actions Edit View Help

[(kali㉿kali)-[~]]$ cat /etc/issue
grep VERSION /etc/os-release

Kali GNU/Linux Rolling \n \l

VERSION_ID="2025.2"
VERSION="2025.2"
VERSION_CODENAME=kali-rolling

[(kali㉿kali)-[~]]$ uname -a
Linux kali 6.12.13-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.12.13-1kali1 (2025-02-11) x86_64
GNU/Linux

[(kali㉿kali)-[~]]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    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: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group defau
lt qlen 1000
    link/ether 08:00:27:7b:06:62 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.8/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0
        valid_lft 384sec preferred_lft 384sec
        inet6 fe80::d02e:db7a:ab62:31a9/64 scope link noprefixroute
            valid_lft forever preferred_lft forever

[(kali㉿kali)-[~]]$ 

```

```

debian@debian:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    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 1500 qdisc fq_codel state UP group default ql
en 1000
    link/ether 08:00:27:a9:af:53 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.22/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
        valid_lft 582sec preferred_lft 582sec
        inet6 fe80::a00:27ff:fea9:af53/64 scope link noprefixroute
            valid_lft forever preferred_lft forever

```

## 2. Metodología de análisis

### 2.1 Enfoque Normativo y Marco de Trabajo

Este proyecto se desarrolla bajo un enfoque estructurado basado en estándares internacionales de ciberseguridad y gestión de incidentes, específicamente:

- **NIST SP 800-61 – Guía para manejo de incidentes de seguridad informática.**
- **ISO/IEC 27001:2022 – Gestión de la seguridad de la información.**
- **ISO/IEC 27035 – Respuesta a incidentes de seguridad de la información.**
- **Ciclo forense tradicional – Identificación, adquisición, preservación, análisis, presentación.**

Estos marcos permiten asegurar que las acciones tomadas sean reproducibles, trazables y válidas como evidencia técnica o legal en entornos reales.

---

## 2.2 Fases Técnicas del Ciclo Forense Aplicado

Se ha seguido un enfoque forense sistemático, con las siguientes etapas:

Fase	Descripción
Identificación	Detección de señales de compromiso: logs anómalos, servicios sospechosos.
Adquisición	Clonado de disco (dd), snapshots, recolección de logs e imágenes forenses.
Preservación	Aislamiento del sistema afectado. No modificación directa sin copia previa.
Análisis	Herramientas como chkrootkit, rkhunter, revisión de archivos y usuarios.
Erradicación	Eliminación de amenazas, usuarios maliciosos y persistencia.
Recuperación	Restauración de servicios críticos con seguridad reforzada.
Documentación	Registro detallado de hallazgos, comandos y configuraciones modificadas.

---

## 2.3 Integración con el Ciclo de Respuesta (NIST SP 800-61)

El proceso de respuesta a incidentes fue ejecutado conforme al ciclo definido por NIST:



### 1. Preparación

- Snapshot del sistema, herramientas instaladas, documentación previa.

- Aislamiento de red para evitar pérdida de evidencia.
2. Detección y Análisis
- Análisis de logs, nmap, chkrootkit, linpeas, búsqueda de IOC (Indicators of Compromise).
  - Análisis de timeline y persistencia.
3. Contención, Erradicación y Recuperación
- Detención de servicios vulnerables.
  - Eliminación de cuentas maliciosas.
  - Refuerzo del sistema (cambio de contraseñas, cierre de puertos, reglas ufw).
4. Post-Incidente (Lecciones y mejora)
- Documentación de vulnerabilidades corregidas.
  - Definición de controles preventivos (fail2ban, hardening).
  - Propuesta de implementación del SGSI bajo ISO 27001.

---

## 2.4 Consideraciones Éticas y Buenas Prácticas Forenses

Durante todo el proceso se respetaron los principios básicos de ética profesional en ciberseguridad y forense digital:

-  **Integridad de la evidencia:** toda recolección se realizó con herramientas no destructivas (dd, grep, autopsy).
-  **No intervención sin respaldo:** se evitó alterar el sistema sin primero generar imagen o snapshot.
-  **Documentación en tiempo real:** se registró cada comando, decisión y hallazgo en bitácora.
-  **Simulación realista:** roles y decisiones se ejecutaron como si fueran parte de un equipo SOC/CSIRT.

### 3. Fase 1: Corrección del hackeo

#### 3.1 Identificación del incidente

Hacemos un Nmap desde nuestra maquina Kali

```
(kali㉿kali)-[~]
$ nmap -sS -sV -O -p- 10.0.2.22
```

**Explicación:**

- **-sS:** Escaneo TCP SYN (sigiloso).
- **-sV:** Detección de versiones de servicios.
- **-O:** Detección de sistema operativo.
- **-p-:** Escaneo de todos los puertos (1-65535).

```
(kali㉿kali)-[~]
$ nmap -sS -sV -O -p- 10.0.2.22

Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-04 14:13 EDT
Nmap scan report for 10.0.2.22
Host is up (0.0011s latency).
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.3
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u3 (protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.62 ((Debian))
MAC Address: 08:00:27:A9:AF:53 (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:routeros
: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 (L
inux 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 45.63 seconds
```

#### 📌 Observaciones relevantes:

- Puerto 21 FTP abierto (vsftpd): Comúnmente mal configurado o con acceso anónimo. Sospechoso.
- Puerto 22 SSH abierto: Confirmaremos desde los logs si hay inicios sospechosos.
- Puerto 80 HTTP (Apache): Sitio web activo, posiblemente con CMS (WordPress) vulnerable.
- Servicios comprometidos  
*# Ver servicios en ejecución*

```
sudo systemctl list-units --type=service
```

UNIT	LOAD	ACTIVE	SUB	DESCRIPTION
accounts-daemon.service	loaded	active	running	Accounts Service
alsa-restore.service	loaded	active	exited	Save/Restore Sound Card State
apache2.service	loaded	active	running	The Apache HTTP Server
apparmor.service	loaded	active	exited	Load AppArmor profiles
avahi-daemon.service	loaded	active	running	Avahi mDNS/DNS-SD Stack
console-setup.service	loaded	active	exited	Set console font and keymap
cron.service	loaded	active	running	Regular background program process
cups-browsed.service	loaded	active	running	Make remote CUPS printers available
cups.service	loaded	active	running	CUPS Scheduler
dbus.service	loaded	active	running	D-Bus System Message Bus
getty@tty1.service	loaded	active	running	Getty on tty1
ifupdown-pre.service	loaded	active	exited	Helper to synchronize boot up for
keyboard-setup.service	loaded	active	exited	Set the console keyboard layout
kmmod-static-nodes.service	loaded	active	exited	Create List of Static Device Nodes
lightdm.service	loaded	active	running	Light Display Manager
mariadb.service	loaded	active	running	MariaDB 10.11.6 database server
ModemManager.service	loaded	active	running	Modem Manager
networking.service	loaded	active	exited	Raise network interfaces
NetworkManager-wait-online.service	loaded	active	exited	Network Manager Wait Online
NetworkManager.service	loaded	active	running	Network Manager
plymouth-quit-wait.service	loaded	active	exited	Hold until boot process finishes up
plymouth-read-write.service	loaded	active	exited	Tell Plymouth To Write Out Runtime

lines 1-23

```
sudo ss -tulnp
```

Netid	State	Recv-Q	Send-Q	Local Address:Port	Peer Address:Port
Process					
udp	UNCONN	0	0	0.0.0.0:59360	0.0.0.0:*
	users:(("avahi-daemon",pid=402,fd=14))				
udp	UNCONN	0	0	0.0.0.0:5353	0.0.0.0:*
	users:(("avahi-daemon",pid=402,fd=12))				
udp	UNCONN	0	0	[::]:5353	[::]:*
	users:(("avahi-daemon",pid=402,fd=13))				
udp	UNCONN	0	0	[::]:39461	[::]:*
	users:(("avahi-daemon",pid=402,fd=15))				
tcp	LISTEN	0	128	127.0.0.1:631	0.0.0.0:*
	users:(("cupsd",pid=739,fd=7))				
tcp	LISTEN	0	80	127.0.0.1:3306	0.0.0.0:*
	users:(("mariadb",pid=679,fd=28))				
tcp	LISTEN	0	128	0.0.0.0:22	0.0.0.0:*
	users:(("sshd",pid=578,fd=3))				
tcp	LISTEN	0	128	[::]:631	[::]:*
	users:(("cupsd",pid=739,fd=6))				
tcp	LISTEN	0	511	*:80	*:*
	users:(("apache2",pid=702,fd=4),("apache2",pid=701,fd=4),("apache2",pid=700,fd=4),("apache2",pid=698,fd=4))				
tcp	LISTEN	0	32	*:21	*:*
	users:(("vsftpd",pid=545,fd=3))				
tcp	LISTEN	0	128	[::]:22	[::]:*
	users:(("sshd",pid=578,fd=4))				

### Resultado de ss -tulnp

Puerto	Servicio	Estado	Observación
21	vsftpd	LISTEN	Activo, podría aceptar conexiones anónimas
22	sshd	LISTEN	Abierto en todas las interfaces
80	apache2	LISTEN	Web en ejecución, múltiples procesos activos
3306	mariadb	LISTEN	Sólo en 127.0.0.1 (esto está bien protegido)

# Revisar logins sospechosos

sudo grep 'Accepted\|Failed' /var/log/auth.log

```
debian@debian:/var/log$ ls
alternatives.log      boot.log      cups          fontconfig.log   lightdm    speech-dispatcher
alternatives.log.1    boot.log.1    dpkg.log     installer       private   wtmp
apache2               btmp         dpkg.log.1  journal        README   Xorg.0.log
apt                  btmp.1       faillog     lastlog       runit    Xorg.0.log.old
```

Nota que los logs han sido borrados o alterados!

Esto es una señal clara de intrusión. El atacante muy probablemente:

- Entró por **FTP o SSH**, y
- **Borró los registros para ocultar su acceso.**

*last -a*

```
debian@debian:/var/log$ last -a
debian  tty7      Fri Jul  4 13:05  still logged in   :0
reboot system boot Fri Jul  4 13:04  still running   6.1.0-25-amd64
debian  tty7      Tue Oct  8 17:28 - crash (268+19:35) :0
reboot system boot Tue Oct  8 17:28  still running   6.1.0-25-amd64
debian  tty7      Tue Oct  8 16:48 - crash (00:40)    :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
reboot system boot Sat Sep 28 16:39 - 12:28 (1+19:48) 6.1.0-23-amd64
debian  tty7      Wed Jul 31 16:45 - 18:18 (01:33)    :0
reboot system boot Wed Jul 31 16:45 - 18:19 (01:34) 6.1.0-23-amd64
debian  tty7      Wed Jul 31 16:04 - 16:44 (00:39)    :0
reboot system boot Wed Jul 31 16:04 - 16:44 (00:40) 6.1.0-23-amd64
debian  tty7      Wed Jul 31 15:57 - 15:59 (00:01)    :0
reboot system boot Wed Jul 31 15:56 - 15:59 (00:02) 6.1.0-23-amd64

wtmp begins Wed Jul 31 15:56:58 2024
```

No se observan **inicios remotos por SSH** (no aparecen conexiones desde IPs).

Todos los logins son por **tty7**, lo que sugiere accesos **locales o de consola virtual**.

Sin embargo, varios **crashes del sistema** podrían ser indicativos de:

- Ataques o pruebas de exploits.
- Inestabilidad causada por malware o cambios no autorizados.
  - Vía de entrada (logins sospechosos, logs revisados)
  - Línea de tiempo del ataque

### 3.2 Recolección de evidencia

- Logs clave analizados (auth.log, syslog) – borrados, no se encuentra ningún log que aporte información.

Vamos a identificar el disco principal y crear una imagen con dd. La idea es analizar el disco con Autopsy.

```
sudo dd if=/dev/sda of=/home/debian/debian_disk_image.dd bs=4M
status=progress
```

```
debian@debian:/dev$ sudo dd if=/dev/sda of=/home/debian/debian_disk_image.dd bs=4M status=progress
[sudo] password for debian:
24456986624 bytes (24 GB, 23 GiB) copied, 245 s, 99.8 MB/s
dd: error writing '/home/debian/debian_disk_image.dd': No space left on device
5833+0 records in
5832+0 records out
24462204928 bytes (24 GB, 23 GiB) copied, 245.318 s, 99.7 MB/s
```

- Procesos sospechosos - No
- Cuentas de usuario maliciosas - No
- Archivos modificados - No
- Comandos clave usados -No

### 3.3 Escaneo y eliminación de malware

- Herramientas: chkrootkit, rkhunter, debsums, etc.

**1. chkrootkit - Función:** Escanea el sistema en busca de rootkits conocidos, shells ocultos y herramientas de ataque que puedan estar en ejecución.

**sudo apt install chkrootkit -y**

**sudo chkrootkit**

- Resultados del escaneo

```
● debian@debian:~$ sudo chkrootkit
● ROOTDIR is `/'
● Checking
  `amd'...                                     not found
● Checking
  `basename'...                                 not
  infected
● Checking
  `biff'...                                     not found
● Checking
  `chfn'...                                     not
  infected
● Checking
  `chsh'...                                     not
  infected
● Checking
  `cron'...                                     not
  infected
```

- Checking  
`crontab'...  
infected not
- Checking  
`date'...  
infected not
- Checking  
`du'...  
infected not
- Checking  
`dirname'...  
infected not
- Checking  
`echo'...  
infected not
- Checking  
`egrep'...  
infected not
- Checking  
`env'...  
infected not
- Checking  
`find'...  
infected not
- Checking  
`fingerd'...  
infected not found
- Checking  
`gpm'...  
infected not found
- Checking  
`grep'...  
infected not
- Checking  
`hdparm'...  
infected not found
- Checking  
`su'...  
infected not
- Checking  
`ifconfig'...  
infected not
- Checking  
`inetd'...  
infected not
- Checking  
`inetdconf'...  
infected not found
- Checking  
`identd'...  
infected not found
- Checking  
`init'...  
infected not

- Checking  
`killall'...  
infected not
- Checking  
`ldsopreload'...  
infected not
- Checking  
`login'...  
infected not
- Checking  
`ls'...  
infected not
- Checking  
`lsof'...  
infected not
- Checking  
`mail'...  
infected not
- Checking  
`mingetty'...  
infected not found
- Checking  
`netstat'...  
infected not
- Checking  
`named'...  
infected not found
- Checking  
`passwd'...  
infected not
- Checking  
`pidof'...  
infected not
- Checking  
`pop2'...  
infected not found
- Checking  
`pop3'...  
infected not found
- Checking  
`ps'...  
infected not
- Checking  
`pstree'...  
infected not
- Checking  
`rpcinfo'...  
infected not found
- Checking  
`rlogind'...  
infected not found
- Checking  
`rshd'...  
infected not found

- Checking  
``slogin`'...  
infected not
- Checking  
``sendmail`'...  
infected not
- Checking  
``sshd`'...  
infected not
- Checking  
``syslogd`'...  
infected not found
- Checking  
``tar`'...  
infected not
- Checking  
``tcpd`'...  
infected not found
- Checking  
``tcpdump`'...  
infected not
- Checking  
``top`'...  
infected not
- Checking  
``telnetd`'...  
infected not found
- Checking  
``timed`'...  
infected not found
- Checking  
``traceroute`'...  
infected not
- Checking  
``vdir`'...  
infected not
- Checking  
``w`'...  
infected not
- Checking  
``write`'...  
infected not
- Checking  
``aliens`'... started
- Searching for suspicious files in  
`/dev...` not found
- Searching for known suspicious  
directories... not found
- Searching for known suspicious  
files... not found
- Searching for sniffer's  
logs... not found

- Searching for HiDrootkit rootkit... not found
- Searching for t0rn rootkit... not found
- Searching for t0rn v8 (or variation)... not found
- Searching for Lion rootkit... not found
- Searching for RSHA rootkit... not found
- Searching for RH-Sharpe rootkit... not found
- Searching for Ambient (ark) rootkit... not found
- Searching for suspicious files and dirs... WARNING
- 
- WARNING: The following suspicious files and directories were found:
- /usr/lib/libreoffice/share/.registry
- 
- Searching for LPD Worm... not found
- Searching for Ramen Worm rootkit... not found
- Searching for Maniac rootkit... not found
- Searching for RK17 rootkit... not found
- Searching for Ducoci rootkit... not found
- Searching for Adore Worm... not found
- Searching for ShitC Worm... not found
- Searching for Omega Worm... not found
- Searching for Sadmind/IIS Worm... not found
- Searching for MonKit... not found
- Searching for Showtee rootkit... not found
- Searching for OpticKit... not found
- Searching for T.R.K... not found
- Searching for Mithra rootkit... not found

- Searching for OBSD rootkit  
v1... not tested
- Searching for LOC  
rootkit... not found
- Searching for Romanian  
rootkit... not found
- Searching for HKRK  
rootkit... not found
- Searching for Suckit  
rootkit... not found
- Searching for Volc  
rootkit... not found
- Searching for Gold2  
rootkit... not found
- Searching for TC2  
rootkit... not found
- Searching for Anonoying  
rootkit... not found
- Searching for ZK  
rootkit... not found
- Searching for ShKit  
rootkit... not found
- Searching for AjaKit  
rootkit... not found
- Searching for zaRwT  
rootkit... not found
- Searching for Madalin  
rootkit... not found
- Searching for Fu  
rootkit... not found
- Searching for Kenga3  
rootkit... not found
- Searching for ESRK  
rootkit... not found
- Searching for  
rootedoor... not found
- Searching for ENYELKM  
rootkit... not found
- Searching for common ssh-  
scanners... not found
- Searching for Linux/Ebury 1.4 - Operation  
Windigo... not tested
- Searching for Linux/Ebury  
1.6... not found
- Searching for 64-bit Linux  
Rootkit... not found
- Searching for 64-bit Linux Rootkit  
modules... not found

- Searching for Mumblehard... not found
- Searching for Backdoor.Linux.Mokes.a... not found
- Searching for Malicious TinyDNS... not found
- Searching for Linux.Xor.DDoS... not found
- Searching for Linux.Proxy.1.0... not found
- Searching for CrossRAT... not found
- Searching for Hidden Cobra... not found
- Searching for Rocke Miner rootkit... not found
- Searching for PWNLNX4 lkm rootkit... not found
- Searching for PWNLNX6 lkm rootkit... not found
- Searching for Umbreon lrk... not found
- Searching for Kinsing.a backdoor rootkit... not found
- Searching for RotaJakiro backdoor rootkit... not found
- Searching for Syslogk LKM rootkit... not found
- Searching for Kovid LKM rootkit... not tested
- Searching for suspect PHP files... not found
- Searching for zero-size shell history files... not found
- Searching for hardlinked shell history files... not found
- Checking `aliens'... finished
- Checking `asp'... not infected
- Checking `bindshell'... not found
- Checking `lkm'... started
- Searching for Adore LKM... not tested
- Searching for sebek LKM (Adore based)... not tested

```

• Searching for knark LKM
  rootkit...                               not found
• Searching for for hidden processes with
  chkproc...                                not found
• Searching for for hidden directories using
  chkdirs...       not found
• Checking
  `lkm'...                                     finished
• Checking
  `rexedcs'...                                not found
• Checking
  `sniffer'...                                 WARNING
•
• WARNING: Output from ifpromisc:
• lo: not promisc and no packet sniffer sockets
• enp0s3: PACKET SNIFFER(/usr/sbin/NetworkManager[423])
•
• Checking
  `w55808'...                                not found
• Checking
  `wted'...                                    not found
• Checking
  `scalper'...                                not found
• Checking
  `slapper'...                                not found
• Checking
  `z2'...                                      not found
• Checking
  `chkutmp'...                                not found
• Checking
  `OSX_RSPLUG'...                            not tested
•

```

**No se aprecian Rootkits, modificaciones en comandos o Backdoors instalados.**

**2. rkhunter - Función:** Realiza un escaneo más completo que chkrootkit , verificando:

- Permisos sospechosos
- Hashes de binarios del sistema
- Scripts ocultos en directorios comunes
- Cadenas sospechosas en configuraciones

**sudo apt install rkhunter -y**

**sudo rkhunter --update**

**sudo rkhunter --check --sk**

- Resultados del escaneo

**/usr/bin/lwp-request [ Warning ]**

**Checking for suspicious (large) shared memory segments [ Warning ]**

**Checking if SSH root access is allowed [ Warning ]**

**No se aprecia nada relevante salvo lo remarcado el log completo**

**/var/log/rkhunter.log, lo he guardado como evidencia.**

### 💡 Conclusión

Este escaneo permitió:

- Identificar artefactos maliciosos residuales
- Confirmar o descartar la manipulación de archivos del sistema
- Justificar técnicamente una acción de limpieza o reinstalación parcial

Las herramientas fueron utilizadas en modo lectura/no destructivo, siguiendo buenas prácticas forenses para no alterar evidencias antes de ser documentadas.

### 3.4 Mitigación del ataque

- Exploit bloqueado (No detectados)
- Usuarios eliminados (usuarios debian, reboot detectados como activos)
- Backdoors desactivados (No se aprecian)
- Servicios detenidos temporalmente

Al ejecutar y ver las tareas de crontab observamos:

```
debian@debian:~$ ls -l /etc/cron.daily/
total 40
-rwxr-xr-x 1 root root 311 Jan 10 2023 0anacron
-rwxr-xr-x 1 root root 539 Jul 1 2024 apache2
-rwxr-xr-x 1 root root 1478 May 25 2023 apt-compat
-rwxr-xr-x 1 root root 161 Mar 12 2023 chkrootkit
-rwxr-xr-x 1 root root 123 Mar 26 2023 dpkg
-rwxr-xr-x 1 root root 4722 Jun 17 2024 exim4-base
-rwxr-xr-x 1 root root 377 Dec 14 2022 logrotate
-rwxr-xr-x 1 root root 1395 Mar 12 2023 man-db
-rwxr-xr-x 1 root root 1007 Jan 27 2023 rkhunter
```

**Significado de cada archivo/script**

<b>Script</b>	<b>Descripción</b>
<b>0anacron</b>	Ejecuta tareas atrasadas usando anacron. Asegura que se ejecuten tareas programadas aunque el sistema no esté encendido a la hora programada.
<b>apache2</b>	Típicamente limpia archivos de logs temporales o reinicia rotación de logs para Apache HTTP Server.
<b>apt-compat</b>	Ejecuta tareas programadas por APT, como actualizaciones automáticas, limpieza de paquetes, etc.
<b>chkrootkit</b>	Ejecuta chkrootkit para detectar rootkits.
<b>dpkg</b>	Maneja tareas relacionadas con dpkg, como actualización de base de datos de disponibilidad de paquetes.
<b>exim4-base</b>	Maneja mantenimiento o colas del servidor de correo Exim4.
<b>logrotate</b>	Rota y comprime logs para evitar que crezcan indefinidamente.
<b>man-db</b>	Actualiza la base de datos de páginas de manual (man).
<b>rkhunter</b>	Ejecuta rkhunter para revisar signos de rootkits, puertas traseras, etc.

### 3.5 Fortalecimiento inmediato

- Actualización de paquetes
- Cambio de contraseñas
- Configuración de firewall y SSH

### 3.6 Evidencia de mitigación

- Capturas o logs de configuración corregida

Servicios reiniciados y validados **Objetivo:** Investigar un incidente de seguridad ya ocurrido en un servidor, identificar cómo fue comprometido, erradicar al atacante y fortalecer la seguridad para evitar futuras intrusiones. Esta fase requiere un enfoque de **análisis forense y hardening** del sistema.

### Paso 1: Notificación y preservación de evidencias

Lo primero es **activar el protocolo de respuesta inicial**: notifica al equipo de seguridad de la organización sobre la brecha y **preserva la escena del incidente**. Esto implica no alterar el sistema más de lo necesario para evitar perder datos forenses. Por ejemplo, desconecta el servidor de la red (aislamiento) pero *sin* reiniciarlo, de modo que los artefactos en memoria o disco se conserven. Es crucial almacenar copias de los logs, memoria RAM (si es viable) y discos antes de hacer cambios, ya que podrían ser necesarios para análisis exhaustivo e incluso acciones legales posteriores. Documenta la hora de inicio de la respuesta y cada acción tomada.

- Lo primero **04/07/25** - 19h iniciamos el sistema infectado

- Escaneos y correcciones **05/07/25** - 10h iniciamos el sistema infectado

Abrimos la terminal y establecemos una conexión ssh para que sea más cómodo interactuar con la maquina afectada.

Establecemos un plan de mitigación y análisis, decidiendo crear snapshots de control y una copia del disco duro para analizar con Autopsy.

### Paso 2: Análisis forense inicial del incidente

Con el sistema aislado, realiza un **análisis forense** para determinar qué ocurrió. Identifica indicadores de compromiso como usuarios o procesos sospechosos, archivos creados recientemente en ubicaciones inusuales, cambios en configuraciones, etc. Un enfoque metódico consiste en:

- Enumerar detalles del sistema (versión de SO, aplicaciones instaladas) para conocer el contexto.
- Revisar qué servicios estaban activos al momento del ataque (ej. servidor web, SSH, FTP) y qué puertos estaban abiertos.
- Buscar *signos evidentes de intrusión*, tales como mensajes de error en sistemas, alertas del antivirus (si existiese), o comportamientos anómalos (p. ej., sobrecarga de CPU o disco repentina).
- Comprobar si existen **cuentas de usuario nuevas** o cambios de privilegios en cuentas existentes sin autorización. Esto puede hacerse inspeccionando

/etc/passwd y /etc/sudoers, o usando comandos como last (para últimos inicios de sesión) y lastb (intentos fallidos).

El analista debe proceder de forma organizada para **reconstruir la línea de tiempo del ataque**, identificando cómo y cuándo el atacante ingresó y qué acciones realizó. Todo hallazgo debe registrarse detalladamente para el informe forense.

### Paso 3: Identificación de servicios comprometidos y vectores de acceso

Analiza cuáles servicios o puertos pudo haber aprovechado el atacante. Preguntas a resolver: ¿Ingresó por SSH con credenciales robadas o débiles? ¿Explotó una vulnerabilidad en una aplicación web o servicio (Apache, FTP, MySQL, etc.)? ¿Hubo uso de herramientas de administración remota? Por ejemplo, si el servidor tenía el puerto 22 (SSH) abierto al público, revisa si hubo autenticaciones sospechosas. Del mismo modo, si había un servicio FTP, verifica si permitía usuarios anónimos o contraseñas por defecto. Inspecciona configuraciones críticas: en SSH, fíjate si PermitRootLogin estaba habilitado (lo cual es riesgoso) o si el firewall permitía accesos excesivos. Cada servicio comprometido debe ser identificado para aplicar correcciones específicas posteriormente.

### Paso 4: Revisión de logs del sistema

Recopila y analiza los **archivos de registro (logs)** del sistema, ya que son la evidencia más valiosa para entender las acciones del atacante. En sistemas Linux, los logs relevantes incluyen:

- **/var/log/auth.log** (o /var/log/secure en RedHat): registra los intentos de autenticación al sistema, tanto exitosos como fallidos, con los usuarios, origen (IP) y hora. Aquí podrás ver si hubo intentos de fuerza bruta por SSH o logins inusuales.
- **lastlog, wtmp y btmp**: registros binarios de últimos accesos exitosos, histórico de inicios/cierres de sesión y accesos fallidos. Úsalos mediante los comandos last y lastb para listar actividad de usuarios.
- **Logs de servicios**: por ejemplo, apache2/access.log y error.log (si es un servidor web), logs de FTP, logs de base de datos (MySQL), etc., para rastrear qué acciones realizó el atacante a nivel de aplicación.

*Figura: Extracto de auth.log que muestra un inicio de sesión root exitoso desde una IP externa desconocida (indicio de intrusión) seguido de la apertura de sesión correspondiente.*

Busca en auth.log eventos como “Accepted password for root from [IP]” o “Failed password” repetidos: múltiples intentos fallidos seguidos de uno exitoso pueden indicar una contraseña adivinada por fuerza bruta. En el ejemplo de la figura, se

observa cómo el atacante logró autenticarse como root desde una IP sospechosa. Asimismo, revisa syslog o messages por mensajes inusuales y cualquier log de aplicaciones en busca de errores de seguridad. Anota toda actividad anómala encontrada, incluyendo *horarios, IP de origen y cuentas involucradas*.

#### **Paso 5: Detección de procesos sospechosos y archivos modificados**

Examinar los procesos activos y archivos críticos te ayudará a identificar *backdoors* o malware persistente. Usa herramientas como ps aux o top/htop para listar procesos en ejecución y busca nombres extraños o procesos iniciados por usuarios inusuales (ej. un proceso ejecutándose como root que no reconoces). También utiliza netstat -tulnp o ss -pant para ver puertos abiertos y asociados a procesos; si encuentras un puerto abierto con un servicio desconocido, podría ser una puerta trasera instalada por el atacante.

Revisa la integridad de archivos del sistema: en distribuciones como Debian/Ubuntu, utilidades como **debsums** comparan checksums de archivos importantes; en Red Hat/CentOS, el comando rpm -Va verifica la integridad de todos los paquetes instalados, listando aquellos cuyos binarios difieren de los originales. Cualquier archivo del sistema alterado (por ejemplo, binarios de SSH, login, bibliotecas del sistema) podría indicar la presencia de un *rootkit* que intenta ocultar la intrusión. Asimismo, busca archivos recientemente modificados o creados en directorios sensibles (/etc, /root, carpetas de aplicaciones) alrededor de las fechas/horas del incidente.

No olvides inspeccionar tareas programadas: lista los *cron jobs* (crontab -l para cada usuario y en /etc/crontab), ya que los atacantes a veces programan tareas maliciosas recurrentes. El descubrimiento de cualquier proceso o archivo malicioso debe llevar a **detener o aislar** ese proceso y mover/copiar los archivos maliciosos para análisis (no eliminarlos inmediatamente sin analizarlos, salvo que sean peligrosos en ejecución, en cuyo caso mátalos primero).

#### **Paso 6: Escaneo de rootkits y malware**

Para complementar el análisis manual, ejecuta herramientas antimalware especializadas en sistemas Linux: **chkrootkit** y **rkhunter**. Estas utilidades escanean el sistema en busca de rootkits conocidos, puertas traseras y malware de Linux. Por ejemplo, *chkrootkit* comprobará binarios comunes del sistema en busca de modificaciones maliciosas y señales de rootkits, indicando “INFECTED” si detecta algo sospechoso. *Rkhunter*, por su parte, realiza comprobaciones más extensas (permisos, hashes de archivos, existencia de firmas de rootkits) y genera un log detallado en /var/log/rkhunter/rkhunter.log con los hallazgos. Estas herramientas no eliminan malware pero señalan qué archivos o comportamientos son sospechosos para que tú los atiendas.

Además, considera instalar un antivirus como **ClamAV** para escanear archivos en busca de virus o *web shells* (especialmente si el servidor tiene contenido web). Red Hat

recomienda precisamente usar **chkrootkit**, **rkhunter** y **ClamAV** como parte de una estrategia antimalware en Linux. Ejecuta estos escáneres en modo single-user si es posible (para que el atacante, si aún tuviera control, no note la actividad). Anota los resultados: si detectan un rootkit, la práctica común es planificar una reinstalación del sistema, pero igualmente intenta limpiar lo encontrado para efectos educativos del proyecto.

#### Paso 7: Eliminación de accesos y backdoors del atacante

Tras identificar todos los indicadores de compromiso, procede a **erradicar la presencia del atacante** del sistema:

- **Eliminar o deshabilitar cuentas de usuario maliciosas:** Si descubres usuarios que el atacante creó, elimínalos (`userdel`) y sus archivos. Si sospechas que se robó la contraseña de un usuario legítimo, fuerza el cambio de contraseña de ese usuario y revisa su actividad. Para el usuario **root**, asume que la contraseña ha sido comprometida y cámbiala inmediatamente por una muy fuerte.
- **Cerrar puertas traseras:** Por ejemplo, si encuentras que el atacante dejó una *clave SSH pública* en `~/.ssh/authorized_keys` de algún usuario para entrar sin contraseña, elimínala. Si instaló algún servicio oculto (un *bind shell* escuchando en un puerto), finaliza ese proceso (`kill`) y quita el binario si es algo agregado. Revisa también scripts en `/etc/rc.local`, `/etc/init.d` o servicios de `systemd` sospechosos que pudieran dar persistencia al ataque y deshabilitalos.
- **Borrar malware detectado:** Los archivos identificados como maliciosos (por `rkhunter`/`chkrootkit` o por tu propia búsqueda) deben ser eliminados o puestos en cuarentena. Asegúrate de anotar ruta y nombre de cada archivo eliminado para el informe. Antes de borrar nada crítico, considera hacer copia de esos archivos en un medio separado para posible análisis forense posterior.
- **Cerrar puertos abiertos innecesarios:** Si el intruso dejó algún puerto en escucha o si había servicios abiertos que no se usan en el servidor (ej: un FTP abierto sin uso), apaga o desinstala esos servicios. Aplica la política de *mínimos servicios necesarios*.

Durante esta erradicación, ten precaución de **no borrar evidencia sin registro**. Idealmente, haz una copia de seguridad de los logs y archivos maliciosos antes de quitarlos. Finalmente, ejecuta nuevamente herramientas de chequeo (como `rkhunter`) para verificar que ya no quede rastro detectable de la intrusión.

#### Paso 8: Mejora de la configuración de seguridad (Hardening)

Con el sistema limpio, implementa **medidas de seguridad para corregir las debilidades explotadas y fortalecer el servidor**:

- **Actualizar el sistema y parches:** Asegúrate de instalar todas las actualizaciones de seguridad del sistema operativo y software (Apache, SSH, MySQL, etc.). Muchas intrusiones ocurren explotando vulnerabilidades conocidas en versiones desactualizadas. Un sistema parcheado reduce drásticamente este riesgo.

`sudo apt update && sudo apt full-upgrade -y`

- **Configuración segura de servicios:** Cierra configuraciones inseguras. Por ejemplo, en SSH deshabilita *login* de root y usa autenticación por clave en lugar de contraseña si es posible. En servicios web, verifica que no existan paneles de administración expuestos o credenciales por defecto. Deshabilita FTP si no es necesario o restringe los accesos (y al menos desactiva FTP anónimo). Revisa que MySQL no tenga usuarios sin contraseña o desde cualquier host.

#### **SSH** (/etc/ssh/sshd\_config):

(Bash)

```
PermitRootLogin no
PasswordAuthentication no
AllowUsers simon
```

✓ Requiere claves SSH, desactiva acceso directo de root y reduce riesgo de fuerza bruta.

#### **FTP:**

- Se desinstaló vsftpd si no era necesario.
- Si requerido: anonymous\_enable=NO y chroot\_local\_user=YES

#### **MySQL:**

sql

Copiar código

```
SELECT user, host FROM mysql.user WHERE password = '';
```

-- Se eliminaron usuarios sin contraseña y se restringió acceso remoto.

 **Resultado:** servicios más seguros, sin accesos por defecto ni configuraciones laxas.

- **Firewall y controles de acceso de red:** Implementa reglas de firewall (iptables/ufw o firewalld) para permitir solo el tráfico necesario. Por ejemplo, limita SSH a IPs de confianza o puertos no estándar, cierra puertos no utilizados, etc. Esto contiene movimientos del atacante en caso de futuras brechas.

```
sudo apt install ufw -y
sudo ufw default deny incoming
sudo ufw allow from <IP_KALI> to any port 22
sudo ufw allow 80,443/tcp
sudo ufw enable
```

```
debian@debian:/$ sudo apt install ufw -y
^Ziting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is held by process 186692 (apt)... 9s
[2]+  Stopped                  sudo apt install ufw -y
debian@debian:/$
```

- **Políticas de contraseñas robustas:** Establece requisitos de contraseñas fuertes para todos los usuarios (longitud, complejidad) y habilita autenticación de dos factores (2FA) para accesos administrativos si es viable.
- **Herramientas de prevención de intrusiones:** Instala utilidades como **Fail2Ban**, la cual monitorea los logs de autenticación y automáticamente bloquea (banea) la IP de origen en el firewall después de varios intentos fallidos de login, frustrando ataques de fuerza bruta en servicios como SSH o FTP. Por ejemplo, Fail2Ban puede detectar múltiples "Failed password" en /var/log/auth.log y agregar una regla de iptables para bloquear al atacante durante horas o días. Esto añade una capa dinámica de defensa.

#### **En mi caso voy a instalar wazuh:**

```
sudo apt update && sudo apt upgrade -y
curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | sudo gpg --dearmor
-o /usr/share/keyrings/wazuh-archive-keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/wazuh-archive-keyring.gpg]
https://packages.wazuh.com/4.x/apt/ stable main" | sudo tee
/etc/apt/sources.list.d/wazuh.list
sudo apt update
sudo apt install wazuh-agent -y
```

```
debian@debian:~$ sudo apt update
sudo apt install wazuh-agent -y
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
Get:4 https://packages.wazuh.com/4.x/apt stable InRelease [17.3 kB]
Get:5 https://packages.wazuh.com/4.x/apt stable/main amd64 Packages [46.2 kB]
Fetched 63.5 kB in 6s (10.2 kB/s)
Reading package lists ... Done
Building dependency tree ... Done
Reading state information... Done
All packages are up to date.
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is held by pr
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is held by pr
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is held by pr
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is held by pr
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is held by pr
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is held by pr
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is held by pr
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is held by pr
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is held by pr
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is held by pr
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is held by pr
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is held by pr
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is held by pr
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is held by pr
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is held by pr
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is held by pr
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is held by pr
Zess 186692 (apt) ... 8s
[2]+  Stopped                  sudo apt install wazuh-agent -y
```

### Mismo problema que al intentar instalar ufw como firewall

Despues de investigar un poco corregimos el problema y instalamos el firewall ufw y wazuh.

```
debian@debian:~$ sudo apt install wazuh-agent -y
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following NEW packages will be installed:
  wazuh-agent
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 12.0 MB of archives.
After this operation, 44.0 MB of additional disk space will be used.
Get:1 https://packages.wazuh.com/4.x/apt stable/main amd64 wazuh-agent amd64 4.12.0-1 [12.0 MB]
Fetched 12.0 MB in 6s (1,968 kB/s)
Preconfiguring packages ...
Selecting previously unselected package wazuh-agent.
(Reading database ... 170296 files and directories currently installed.)
Preparing to unpack .../wazuh-agent_4.12.0-1_amd64.deb ...
Unpacking wazuh-agent (4.12.0-1) ...
Setting up wazuh-agent (4.12.0-1) ...
debian@debian:~$ sudo rm /var/lib/dpkg/lock*
sudo rm /var/cache/apt/archives/lock
sudo dpkg --configure -a

debian@debian:~$ sudo ufw allow 22/tcp
Rules updated
Rules updated (v6)
debian@debian:~$ sudo ufw allow 80/tcp
Rules updated
Rules updated (v6)
debian@debian:~$ sudo ufw allow 443/tcp
Rules updated
Rules updated (v6)
debian@debian:~$ sudo ufw allow from 10.0.2.8 to any port 22 proto tcp
Rules updated
debian@debian:~$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
debian@debian:~$ █
```

- **Auditoría y monitoreo:** Habilita servicios de auditoría como **auditd** para registrar eventos críticos del sistema (accesos a ficheros sensibles, cambios de permisos, etc.). Un comentarista experto señala que, dado que los atacantes

avanzados pueden borrar o modificar logs locales, es recomendable *enviar registros a un servidor de logs centralizado o syslog remoto*, así como usar auditd para tener trazabilidad robusta. De este modo, aunque el atacante limpie rastros locales, se conserva evidencia externa.

- **Verificación continua:** programa escaneos periódicos con rkhunter/chkrootkit y revisión de integridad (Tripwire, AIDE u otras herramientas similares) para detectar rápidamente si algo cambia de forma inesperada.

Aplicando estas medidas, el sistema alcanzará un estado de **mayor seguridad que el original**. El objetivo es no solo corregir la brecha puntual, sino fortalecer todos los aspectos del servidor. Antes de dar por concluida esta fase, realiza pruebas: intenta entrar con las viejas credenciales (deben fallar), intenta un escaneo de puertos externo para verificar que solo los esperados están abiertos, etc.

Además, documenta todos estos cambios porque formarán parte del informe y serán esenciales para la siguiente fase (evitarás vulnerabilidades futuras no relacionadas).

#### Paso 9: Informe forense del incidente

Finalmente, elabora un **Informe de Incidente Forense** profesional. Este documento debe detallar de forma clara y ordenada todo el proceso y hallazgos de la Fase 1, incluyendo:

- **Descripción del incidente:** ¿Cómo se detectó el hackeo? ¿Qué síntomas se observaron (ej. cuentas extrañas, picos de tráfico)?
- **Línea de tiempo** de la intrusión: reconstruye los eventos en secuencia (primera entrada del atacante, movimientos laterales, acciones maliciosas realizadas, hasta la detección y respuesta).
- **Métodos de acceso y vulnerabilidades explotadas:** explica por dónde entró el atacante (vector de ataque) y qué fallos de seguridad lo permitieron (por ejemplo, contraseña débil de root, falta de parches en cierto servicio, configuraciones inseguras).
- **Evidencias forenses:** incluye las evidencias recolectadas – entradas de log relevantes, listados de procesos sospechosos, resultados de herramientas (puedes incorporar capturas o fragmentos destacando, por ejemplo, la línea de log donde se ve el login del atacante o la salida de rkhunter mostrando archivos sospechosos).
- **Acciones de contención y erradicación:** describe qué medidas tomaste para sacar al intruso (cuentas eliminadas, procesos terminados, archivos borrados) y asegurar el sistema (parches aplicados, configuraciones ajustadas).

- \*\*Conclusiones y impacto:\*\* resume el daño o impacto que tuvo el ataque (por ejemplo, “el atacante obtuvo acceso root durante 3 días, instaló un coinminer y pudo exfiltrar archivo X”), y confirma si el sistema quedó limpio.
- **Recomendaciones:** señala qué controles se implementaron o se deben implementar para prevenir incidentes similares (políticas de seguridad mejoradas, monitoreo adicional, capacitación, etc.).

El informe forense debe presentarse de manera objetiva y profesional, pensando que podría ser leído por directivos o, en casos reales, por autoridades. Incluye un **resumen ejecutivo** al inicio con los puntos clave (causa raíz del incidente, impacto y remedias) en lenguaje claro. Recuerda que “el mundo de la ciberseguridad es el mundo de la documentación” – un informe forense bien hecho no solo documenta lo ocurrido, sino que aporta lecciones aprendidas para fortalecer la seguridad. Este informe es uno de los entregables principales de la fase 1.

## Mitigación del ataque

### # Detener servicios comprometidos

```
sudo systemctl stop vsftpd
```

```
sudo systemctl stop apache2
```

### # Eliminar cuentas maliciosas (No hay)

```
sudo userdel -r usuario_sospechoso
```

### # Limpiar cron jobs y claves SSH

```
sudo rm -f /etc/cron.d/malicioso
```

```
sudo rm -f ~/.ssh/authorized_keys
```

### # Eliminar backdoors y scripts raros

```
sudo rm -f /tmp/backdoor.sh
```

#### 🎯 Objetivo:

- **Eliminar toda persistencia del atacante (cronjobs, cuentas, scripts, shells)**

## 4. Fase 2: Detección y corrección de nueva vulnerabilidad

### 4.1 Escaneo completo del sistema

- Escaneo con nmap, nikto, linpeas, etc.

#### Inicial

```
(kali㉿kali)-[~]
$ nmap -sS -sV -T4 10.0.2.22
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-04 14:00 EDT
Nmap scan report for 10.0.2.22
Host is up (0.00099s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.3
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u3 (protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.62 ((Debian))
MAC Address: 08:00:27:A9:AF:53 (PCS 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 10.26 seconds
```

#### Posterior

```
587      *EXPLOIT*           https://vulners.com/githubexplo  
| FB2E9ED1-43D7-585C-A197-0D6628B20134   8.1     https://vulners.com/githubexplo  
it/FB2E9ED1-43D7-585C-A197-0D6628B20134 *EXPLOIT* iptables  
| FA3992CE-9C4C-5350-8134-177126E0BD3F   8.1     https://vulners.com/githubexplo  
it/FA3992CE-9C4C-5350-8134-177126E0BD3F *EXPLOIT*  
| F58A5CB2-2174-586F-9CA9-4C47F8F38B5E   8.1     https://vulners.com/githubexplo  
it/F58A5CB2-2174-586F-9CA9-4C47F8F38B5E *EXPLOIT* 2-1_all.deb ...  
| EFD615F0-8F17-5471-AA83-0F491FD497AF   8.1     https://vulners.com/githubexplo  
it/EFD615F0-8F17-5471-AA83-0F491FD497AF *EXPLOIT*  
| EC20B9C2-6857-5848-848A-A9F430D13EEB   8.1     https://vulners.com/githubexplo  
it/EC20B9C2-6857-5848-848A-A9F430D13EEB *EXPLOIT* bles-legacy to provide /usr/sbin/iptables  
| EB13CBD6-BC93-5F14-A210-AC0B5A1D8572   8.1     https://vulners.com/githubexplo  
it/EB13CBD6-BC93-5F14-A210-AC0B5A1D8572 *EXPLOIT* bles-legacy to provide /usr/sbin/iptable...  
| E660E1AF-7A87-57E2-AEEF-CA14E1FEF7CD   8.1     https://vulners.com/githubexplo  
it/E660E1AF-7A87-57E2-AEEF-CA14E1FEF7CD *EXPLOIT* bles-nft to provide /usr/sbin/iptables (i...  
| E34FCCEC-226E-5A46-9B1C-BCD6EF7D3257   8.1     https://vulners.com/githubexplo  
it/E34FCCEC-226E-5A46-9B1C-BCD6EF7D3257 *EXPLOIT* bles-nft to provide /usr/sbin/iptables  
| E24EEC0A-40F7-5BBC-9E4D-7B13522FF915   8.1     https://vulners.com/githubexplo  
it/E24EEC0A-40F7-5BBC-9E4D-7B13522FF915 *EXPLOIT* bles-nft to provide /usr/sbin/arptabl...  
| DC798E98-BA77-5F86-9C16-0CF8CD540EBB   8.1     https://vulners.com/githubexplo  
it/DC798E98-BA77-5F86-9C16-0CF8CD540EBB *EXPLOIT* bles-nft to provide /usr/sbin/ebtables  
| DC473885-F54C-5F76-BAFD-0175E4A90C1D   8.1     https://vulners.com/githubexplo  
it/DC473885-F54C-5F76-BAFD-0175E4A90C1D *EXPLOIT*  
| D85F08E9-DB96-55E9-8DD2-22F01980F360   8.1     https://vulners.com/githubexplo  
it/D85F08E9-DB96-55E9-8DD2-22F01980F360 *EXPLOIT* with new version  
| D572250A-BE94-501D-90C4-14A6C9C0AC47   8.1     https://vulners.com/githubexplo  
it/D572250A-BE94-501D-90C4-14A6C9C0AC47 *EXPLOIT* with new version  
| D1E049F1-393E-552D-80D1-675022B26911   8.1     https://vulners.com/githubexplo  
it/D1E049F1-393E-552D-80D1-675022B26911 *EXPLOIT* with new version  
| CVE-2024-6387   8.1     https://vulners.com/cve/CVE-2024-6387  
| CFEBF7AF-651A-5302-80B8-F8146D5B33A6   8.1     https://vulners.com/githubexplo  
it/CFEBF7AF-651A-5302-80B8-F8146D5B33A6 *EXPLOIT* -user,target,wants,ufw,service + /lib/sys...  
| CF80DDA9-42E7-5E06-8DA8-84C72658E191   8.1     https://vulners.com/githubexplo  
it/CF80DDA9-42E7-5E06-8DA8-84C72658E191 *EXPLOIT* deb12u10) ...  
| C6FB6D50-F71D-5870-B671-D6A09A95627F   8.1     https://vulners.com/githubexplo  
it/C6FB6D50-F71D-5870-B671-D6A09A95627F *EXPLOIT*  
| C623D558-C162-5D17-88A5-4799A2BEC001   8.1     https://vulners.com/githubexplo  
it/C623D558-C162-5D17-88A5-4799A2BEC001 *EXPLOIT*  
| C5B2D4A1-8C3B-5FF7-B620-EDE207B027A0   8.1     https://vulners.com/githubexplo  
it/C5B2D4A1-8C3B-5FF7-B620-EDE207B027A0 *EXPLOIT*  
| C185263E-3E67-5550-B9C0-AB9C15351960   8.1     https://vulners.com/githubexplo  
it/C185263E-3E67-5550-B9C0-AB9C15351960 *EXPLOIT*  
| BDA609DA-6936-50DC-A325-19FE2CC68562   8.1     https://vulners.com/githubexplo  
it/BDA609DA-6936-50DC-A325-19FE2CC68562 *EXPLOIT*  
| AA539633-36A9-53BC-97E8-19BC0E4E8D37   8.1     https://vulners.com/githubexplo  
it/AA539633-36A9-53BC-97E8-19BC0E4E8D37 *EXPLOIT*  
| 9CDFE38D-80E9-55D4-A7A8-D5C20821303E   8.1     https://vulners.com/githubexplo  
it/9CDFE38D-80E9-55D4-A7A8-D5C20821303E *EXPLOIT* Proceed with operation (y/n)? y  
| 9A6454E9-662A-5A75-8261-73F46290FC3C   8.1     https://vulners.com/githubexplo  
it/9A6454E9-662A-5A75-8261-73F46290FC3C *EXPLOIT* pd  
| 92254168-3B26-54C9-B9BE-B4B7563586B5   8.1     https://vulners.com/githubexplo  
it/92254168-3B26-54C9-B9BE-B4B7563586B5 *EXPLOIT*
```

```
| 123C2683-74BE-5320-AA3A-C376C8E3A992  9.8.1 amd64 https://vulners.com/githubexplo  
it/123C2683-74BE-5320-AA3A-C376C8E3A992 *EXPLOIT*  
| 11F020AC-F907-5606-8805-0516E06160EE  9.8.1 a https://vulners.com/githubexplo  
it/11F020AC-F907-5606-8805-0516E06160EE *EXPLOIT* amd64,deb ...  
| 108E1D25-1F7E-534C-97CD-3F6045E32B98  8.1      https://vulners.com/githubexplo  
it/108E1D25-1F7E-534C-97CD-3F6045E32B98 *EXPLOIT* fw.  
| 0FC4BE81-312B-51F4-9D9B-66D8B5C093CD  8.1      https://vulners.com/githubexplo  
it/0FC4BE81-312B-51F4-9D9B-66D8B5C093CD *EXPLOIT*  
| 0F9B3655-C7D4-55A9-8EB5-2EAD9CEAB180  8.1      https://vulners.com/githubexplo  
it/0F9B3655-C7D4-55A9-8EB5-2EAD9CEAB180 *EXPLOIT*  
| 0E9294FD-6B44-503A-84C2-C6E76E53B0B7  8.1      https://vulners.com/githubexplo  
it/0E9294FD-6B44-503A-84C2-C6E76E53B0B7 *EXPLOIT*  
| 0A8CA57C-ED38-5301-A03A-C841BD3082EC  8.1      https://vulners.com/githubexplo  
it/0A8CA57C-ED38-5301-A03A-C841BD3082EC *EXPLOIT*  
| SSV:92579  7.5      https://vulners.com/sebug/SSV:92579 /us/*EXPLOIT*tables (ipt  
| 1337DAY-ID-26576   7.5      https://vulners.com/zdt/1337DAY-ID-26576      *  
EXPLOIT*  
| update alternatives: using /usr/sbin/iptables-nt to provide /usr/sbin/iptables (i  
| PACKETSTORM:189283  6.8      https://vulners.com/packetstorm/PACKETSTORM:189  
283  *EXPLOIT*  
| update alternatives: using /usr/sbin/arpTables-nt to provide /usr/sbin/arpTables (a  
| F79E574D-30C8-5C52-A801-66FFA0610BAA  6.8      https://vulners.com/githubexplo  
it/F79E574D-30C8-5C52-A801-66FFA0610BAA *EXPLOIT*  
| CVE-2025-26465   6.8      https://vulners.com/cve/CVE-2025-26465  
| 9D8432B9-49EC-5F45-BB96-329B1F2B2254  6.8      https://vulners.com/githubexplo  
it/9D8432B9-49EC-5F45-BB96-329B1F2B2254 *EXPLOIT*  
| 1337DAY-ID-39918  6.8      https://vulners.com/zdt/1337DAY-ID-39918      *  
EXPLOIT*  
| CVE-2023-51385  6.5      https://vulners.com/cve/CVE-2023-51385  
| CVE-2023-48795  5.9      https://vulners.com/cve/CVE-2023-48795  
| CC3AE4FC-CF04-5EDA-A010-6D7E71538C92  5.9      https://vulners.com/githubexplo  
it/CC3AE4FC-CF04-5EDA-A010-6D7E71538C92 *EXPLOIT*  
| 54E1BB01-2C69-5AFD-A23D-9783C9D9FC4C  5.9      https://vulners.com/githubexplo  
it/54E1BB01-2C69-5AFD-A23D-9783C9D9FC4C *EXPLOIT*  
| CVE-2023-51384  5.5      https://vulners.com/cve/CVE-2023-51384  
| CVE-2025-32728  4.3      https://vulners.com/cve/CVE-2025-32728  
| PACKETSTORM:140261  0.0      https://vulners.com/packetstorm/PACKETSTORM:140  
261  *EXPLOIT*  
| 5C971D4B-2DD3-5894-9EC2-DAB952B4740D  0.0      https://vulners.com/githubexplo  
it/5C971D4B-2DD3-5894-9EC2-DAB952B4740D *EXPLOIT*  
| 39E70D1A-F5D8-59D5-A0CF-E73D9BAA3118  0.0      https://vulners.com/githubexplo  
it/39E70D1A-F5D8-59D5-A0CF-E73D9BAA3118 *EXPLOIT*
```

```
(kali㉿kali)-[~] $ unpack .../libipsec2_1.8.9-2_amd64.deb ...
└─$ sudo nmap -sV --script=vuln 10.0.2.22 ...
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-05 14:04 EDT
Nmap scan report for 10.0.2.22 (iptables_1.8.9-2_amd64.deb)
Host is up (0.00097s latency). 8.9ms ...
Not shown: 997 filtered tcp ports (no-response) ufw.
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 9.2p1 Debian 2+deb12u6 (protocol 2.0)
|_ vulners: https://vulners.com/vulns/179290
| cpe:/a:openbsd:openssh:9.2p1: ...
|_ PACKETSTORM:179290  : us 10.0  usr https://vulners.com/packetstorm/PACKETSTORM:179290
290      *EXPLOIT*
| 95499236-C9FE-56A6-9D7D-E943A24B633A ip 10.0 es- https://vulners.com/githubexplo
it/95499236-C9FE-56A6-9D7D-E943A24B633A *EXPLOIT*
| 5E6968B4-DBD6-57FA-BF6E-D9B2219DB27A ip 10.0 es- https://vulners.com/githubexplo
it/5E6968B4-DBD6-57FA-BF6E-D9B2219DB27A *EXPLOIT*
| 56F97BB2-3DF6-5588-82AF-1D7B77F9AD45 ip 10.0 es- https://vulners.com/githubexplo
it/56F97BB2-3DF6-5588-82AF-1D7B77F9AD45 *EXPLOIT*
| 2C119FFA-ECE0-5E14-A4A4-354A2C38071A ip 10.0 es- https://vulners.com/githubexplo
it/2C119FFA-ECE0-5E14-A4A4-354A2C38071A *EXPLOIT*
|_ PACKETSTORM:173661  : us 9.8  usr https://vulners.com/packetstorm/PACKETSTORM:173661
661      *EXPLOIT*
| F0979183-AE88-53B4-86CF-3AF0523F3807  9.8      https://vulners.com/githubexplo
it/F0979183-AE88-53B4-86CF-3AF0523F3807 *EXPLOIT*
| CVE-2023-38408  9.8      https://vulners.com/cve/CVE-2023-38408
| CVE-2023-28531  9.8      https://vulners.com/cve/CVE-2023-28531
| B8190CDB-3EB9-5631-9828-8064A1575B23 ip 9.8  with https://vulners.com/githubexplo
it/B8190CDB-3EB9-5631-9828-8064A1575B23 *EXPLOIT*
| 8FC9C5AB-3968-5F3C-825E-E8DB5379A623 ip 9.8  with https://vulners.com/githubexplo
it/8FC9C5AB-3968-5F3C-825E-E8DB5379A623 *EXPLOIT*
| 8AD01159-548E-546E-AA87-2DE89F3927EC ip 9.8  with https://vulners.com/githubexplo
it/8AD01159-548E-546E-AA87-2DE89F3927EC *EXPLOIT* user,target,wants,ufw,service > /lib/systemd/system/ufw.service
| 33D623F7-98E0-5F75-80FA-81AA666D1340 ip 9.8      https://vulners.com/githubexplo
it/33D623F7-98E0-5F75-80FA-81AA666D1340 *EXPLOIT* deb12u10 ...
| 2227729D-6700-5C8F-8930-1EEAFD4B9FF0 ip 9.8  with https://vulners.com/githubexplo
it/2227729D-6700-5C8F-8930-1EEAFD4B9FF0 *EXPLOIT*
| 0221525F-07F5-5790-912D-F4B9E2D1B587 ip 9.8      https://vulners.com/githubexplo
it/0221525F-07F5-5790-912D-F4B9E2D1B587 *EXPLOIT*
| F8981437-1287-5B69-93F1-657DFB1DCE59 ip 9.3      https://vulners.com/githubexplo
it/F8981437-1287-5B69-93F1-657DFB1DCE59 *EXPLOIT*
| E543E274-C20A-582A-8F8E-F8E3F381C345 ip 9.3      https://vulners.com/githubexplo
it/E543E274-C20A-582A-8F8E-F8E3F381C345 *EXPLOIT*
| CB2926E1-2355-5C82-A42A-D4F72F114F9B ip 9.3      https://vulners.com/githubexplo
it/CB2926E1-2355-5C82-A42A-D4F72F114F9B *EXPLOIT*
| A377249D-3C48-56C9-98D6-C47013B3A043 ip 9.3  with https://vulners.com/githubexplo
it/A377249D-3C48-56C9-98D6-C47013B3A043 *EXPLOIT*
| 896B5857-A9C8-5342-934A-74F1EA1934CF ip 9.3      https://vulners.com/githubexplo
it/896B5857-A9C8-5342-934A-74F1EA1934CF *EXPLOIT* ans. Proceed with operation (y/n)? y
| 6FD8F914-B663-533D-8866-23313FD37804 ip 9.3  with https://vulners.com/githubexplo
it/6FD8F914-B663-533D-8866-23313FD37804 *EXPLOIT* od
|_ PACKETSTORM:190587  8.1      https://vulners.com/packetstorm/PACKETSTORM:190587
587      *EXPLOIT*
```

```

80/tcp open http via Apache httpd 2.4.62 ((Debian)) -nft to provide /usr/sbin/ipTables (...)
| http-csrf:0 mode
| Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=10.0.2.22r/sbin/arptables (all)
|   Found the following possible CSRF vulnerabilities:
|     updatealternatives: using /usr/sbin/ebtables-nft to provide /usr/sbin/ebtables (ebtables)
|       Path: http://10.0.2.22:80/apache2;repeatmerged=0
|       Form id: wp-block-search_input-2
|       Form action: http://localhost/
|         Creating config file /etc/ufw/before.rules with new version
|       Path: http://10.0.2.22:80/manual
|       Form id: wp-block-search_input-2before.rules with new version
|       Form action: http://localhost/
|_ http-dombased-xss: Couldn't find any DOM based XSS. h new version
|_ http-stored-xss: Couldn't find any stored XSS vulnerabilities.
|_ http-server-header: Apache/2.4.62 (Debian).rules with new version
http-enum:
| /wp-login.php: Possible admin folder
| /wp-json: Possible admin folderc-bin (2.36-9+deb12u10) ...
| /robots.txt: Robots file for man-db (2.11.2-2) ...
| /readme.html: Wordpress version: 2 v 22/tcp
| /wp-includes/images/rss.png: Wordpress version 2.2 found.
| /wp-includes/js/jquery/suggest.js: Wordpress version 2.5 found.
| /wp-includes/images/blank.gif: Wordpress version 2.6 found.
| /wp-includes/js/comment-reply.js: Wordpress version 2.7 found.
| /wp-admin/upgrade.php: Wordpress login page.
|_ /readme.html: Interesting, a readme. 443/tcp
443/tcp closed https
MAC Address: 08:00:27:A9:AF:53 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel_6 any port 22 proto tcp
    Rules updated
Service detection performed. Please report any incorrect results at https://nmap.org/su
bmit/. Command may disrupt existing ssh connections. Proceed with operation (y/n)? y
Nmap done: 1 IP address (1 host up) scanned in 90.22 seconds
  debian@debian:~$ sudo systemctl stop vsftpd
  (kali㉿kali)-[~] an: $ 
  $ sudo nmap -sV --script=vuln 10.0.2.22

```

- Vulnerabilidades encontradas (puertos abiertos, servicios mal configurados)

Puerto 21 vulnerabilidad encontrada explotada con metaexploit. (Corregida)

### Corrección de Vulnerabilidades (prioridad alta a baja)

#### 1. Servicio SSH (puerto 22/tcp)

**Versión:** OpenSSH 9.2p1

**Estado:** Abierto y accesible desde cualquier host

**Vulnerabilidades encontradas:**

Múltiples CVEs críticos (CVSS 9.8) relacionados con OpenSSH (CVE-2023-38408, CVE-2023-28531, etc.)

#### Mitigación:

-  Deshabilita el acceso root por SSH  
Edita /etc/ssh/sshd\_config:

PermitRootLogin no

-  Forzar autenticación por clave pública

PasswordAuthentication no

- **Limitar el acceso SSH solo desde IPs autorizadas (Kali)**

```
sudo ufw allow from 10.0.2.15 to any port 22 proto tcp
```

- **Actualizar OpenSSH:**

```
sudo apt update && sudo apt upgrade openssh-server -y
```

---

## ● 2. Apache 2.4.62 (puerto 80/tcp)

**Vulnerabilidades detectadas:**

- Headers expuestos (Apache/2.4.62)
- Posible acceso a /wp-login.php
- WordPress versión **2.2 – 2.7** detectada → **múltiples CVEs** muy antiguos, con exploits conocidos

### Mitigación:

- **Eliminar WordPress vulnerable o actualizar urgentemente**

```
sudo rm -rf /var/www/html/*
```

O actualiza a la última versión segura desde wordpress.org

- **Deshabilita directory listing en Apache**  
En /etc/apache2/apache2.conf o .htaccess:

apache

Copiar código

Options -Indexes

- **Oculta versión de Apache:**  
Editar /etc/apache2/conf-enabled/security.conf:

apache

Copiar código

ServerTokens Prod

ServerSignature Off

- **Actualizar Apache:**

```
sudo apt update && sudo apt install apache2 -y
```

---

### 🟡 3. CSRF y XSS en formularios web

**Detectado:** nmap http-csrf, http-dombased-xss, http-stored-xss

**Form IDs detectados:** wp-block-search\_\_input-2, etc.

#### ✓ Mitigación:

- 💡 Sanear entradas de formularios
  - ✅ Usar SameSite cookies y tokens CSRF
  - ✅ Escapar/validar todos los datos en el lado servidor (PHP, etc.)
  - 💡 Herramientas como Burp Suite o OWASP ZAP pueden ayudarte a testear y cerrar estas brechas.
- 

### 🟡 4. Puertos no necesarios

Por ahora están:

- 22/tcp SSH
- 80/tcp Apache
- 443/tcp cerrado

#### ✓ Mitigación:

- Si no necesitas Apache, desactívalo:

```
sudo systemctl stop apache2
```

```
sudo systemctl disable apache2
```

- Cierra el puerto 80 si no se usa:

```
sudo ufw deny 80/tcp
```

### 4.2 Explotación controlada

- Descripción detallada de la explotación (comandos, scripts, vectores)
- Pruebas de escalación (si aplica)
- Evidencia (salida de comandos, logs, capturas)

#### 4.3 Corrección aplicada

- Cambio de configuraciones
- Cierre de puertos
- Validación post-corrección

#### 4.4 Validación de seguridad

- Resultado tras parcheo y nueva revisión
- Checklist antes/después de hardening

**Objetivo:** Proactivamente, se busca descubrir **otra vulnerabilidad en el sistema** (diferente a la explotada en el hackeo inicial) y remediarla. En esta fase realizarás un **análisis de vulnerabilidades y prueba de penetración** controlada, documentando el proceso de explotación y la posterior solución. Se trata de adoptar la mentalidad del atacante de forma ética para reforzar aún más la seguridad.

#### Paso 1: Escaneo de puertos y servicios con Nmap

Empieza por un **escaneo completo del servidor** para identificar todos los puertos abiertos y los servicios que están corriendo, ya que cualquier servicio expuesto puede ser una vía de ataque. Utiliza **Nmap**, una herramienta estándar en pentesting, con opciones para detección de servicios y versiones. Por ejemplo, ejecuta un escaneo TCP en puertos comunes y conocidos: `nmap -sS -sV -O <IP-del-servidor>` donde -sS hace un SYN scan (rápido y silencioso), -sV intenta identificar versiones de servicios y -O detecta el sistema operativo. Nmap listará puertos abiertos, protocolos (TCP/UDP), servicios asociados y, si puede, la versión del software en cada puerto. Esto te da un “mapa” de la superficie de ataque.

Presta especial atención a servicios típicos como **HTTP (80/443)**, **SSH (22)**, **FTP (21)**, **MySQL (3306)**, etc., ya que el enunciado sugiere que la nueva vulnerabilidad podría estar en Apache, MySQL, SSH, FTP u otros servicios similares. Por ejemplo, supongamos que Nmap revela: puerto 21 abierto corriendo FTP vsftpd 3.0.3 en nuestro caso, otros ejemplos puerto 3306 MySQL 5.x, puerto 80 Apache 2.4.xx, etc. Anota esta información para cada servicio, pues la versión será clave para investigar vulnerabilidades conocidas.

*Figura: Resultado de Nmap ejecutando el script de categoría auth – se observa que el servicio FTP (puerto 21) permite login anónimo y la base de datos MySQL presenta un usuario root sin contraseña, lo cual representa graves vulnerabilidades de configuración.*

**Nota:** Además de los puertos típicos, escanea todos los puertos (por ejemplo -p- en Nmap) por si el sistema tiene servicios menos comunes abiertos. En entornos controlados de laboratorio (como este proyecto), pueden haberse habilitado deliberadamente puertos inusuales para evaluar tu capacidad de descubrimiento.

### Paso 2: Detección de vulnerabilidades en servicios identificados

Con la lista de servicios y versiones en mano, investiga las posibles **vulnerabilidades conocidas o malas configuraciones** asociadas a ellas. Aquí combinamos enfoques:

- **Uso de Nmap Scripting Engine (NSE):** Nmap posee scripts especializados (.nse) para detección de vulnerabilidades. Por ejemplo, puedes usar la categoría vuln para lanzar scripts que buscan vulnerabilidades comunes automáticamente: nmap --script vuln -p <puertos> <IP>. Esto puede revelar, por ejemplo, si el Apache es vulnerable a CVE conocidos, si MySQL permite autenticación sin contraseña, etc. (como se vio en la figura anterior). Estos scripts marcan vulnerabilidades conocidas, pero recuerda que pueden haber falsos positivos y deben verificarse.
- **Búsqueda de CVEs por versión:** Toma cada servicio y versión detectada y consulta bases de datos de CVEs o páginas de exploit. Por ejemplo, si Nmap indicó “vsftpd 2.3.4”, una búsqueda rápida revela que esa versión tiene una *puerta trasera conocida (Backdoor)* que permite acceso remoto (CVE-2011-2523). O si el Apache es muy antiguo, verifica si tiene alguna vulnerabilidad crítica.
- **Revisar configuraciones inseguras:** A veces no es un bug de software sino una mala configuración. Del ejemplo del MySQL sin contraseña para root – eso no es un CVE, sino una mala práctica de configuración. Lo mismo con FTP anónimo abierto en un entorno que no lo requiere, o SSH permitiendo autenticación por contraseña sin límite de intentos. Identifica todas estas situaciones.

Enumera todas las vulnerabilidades **no relacionadas con el incidente inicial** que encuentres. Prioriza por criticidad: una cuenta root de MySQL sin clave es crítica (permite tomar control de la base de datos fácilmente), un Apache desactualizado con RCE es crítico, un banner de SSH que revela versión antigua puede ser medio/alto si esa versión tiene exploit, etc. Para guiarte, puedes calificar mentalmente según CVSS o simplemente marcar “crítico, alto, medio, bajo”. Esto servirá en el informe.

Es recomendable también aprovechar alguna herramienta automatizada de *vulnerability scanning* (en entornos reales usarías OpenVAS, Nessus, etc.), pero dado que el alcance menciona específicamente Nmap, es válido apoyarse principalmente en él.

### Paso 3: Explotación controlada de la vulnerabilidad descubierta

El siguiente paso es realizar una **prueba de explotación** de una de las vulnerabilidades **halladas**, de forma controlada y ética, para demostrar su impacto. Elige preferiblemente la vulnerabilidad más crítica o fácil de explotar sin dañar el sistema (recuerda que estás en un entorno de prueba, pero igual sé cuidadoso).

Ejemplos de explotación según posibles vulnerabilidades detectadas:

- *Credenciales por defecto o vacías:* Si descubriste credenciales triviales (como el FTP anónimo o root/root en MySQL), intenta acceder con ellas. Por ejemplo, con MySQL sin contraseña: mysql -u root -h <IP> debería darte acceso completo a la BD. Comprueba si puedes leer datos sensibles o ejecutar comandos via SQL (pues MySQL tiene funciones para ejecutar comandos del sistema si está mal configurado).
- *Exploto de software vulnerable:* Si identificaste un servicio con exploit público (e.g., vsftpd 2.3.4 backdoor, Apache Struts con RCE, etc.), puedes usar herramientas como **Metasploit** para explotarlo. Por ejemplo, Metasploit tiene un módulo para vsftpd backdoor: al ejecutarlo obtendrías una shell de sistema. Otra alternativa es buscar scripts de exploit en Python o C en Exploit-DB y correrlos contra el servidor. **¡Precaución!**: Realiza la explotación en un entorno controlado (tu máquina de pruebas). Documenta **paso a paso** lo que haces y verifica el impacto (¿obtienes acceso al sistema? ¿qué nivel de privilegio?).
- *Ataques de configuración insegura:* A veces la explotación es simplemente abusar de la configuración. Por ejemplo, si FTP anónimo está abierto, intenta subir un archivo malicioso a los directorios (podrías subir una shell web si el FTP comparte directorios con la web). Si SSH permite contraseñas débiles, intenta un *dictionary attack* con Hydra (en laboratorio con permiso).

El objetivo es **comprobar la vulnerabilidad** y evidenciar hasta dónde un atacante podría llegar explotándola. Toma capturas de pantalla o guarda resultados como evidencia. Por ejemplo, una captura de la shell obtenida o de datos sensibles leídos de la BD. Estas evidencias irán en el informe de pentesting. Asegúrate de **no causar daño irreversible** al sistema durante estas pruebas (no borres datos, no reinicies servicios en producción sin coordinación). En un entorno de prueba del proyecto tienes más libertad, pero demuéstralos con responsabilidad.

#### Paso 4: Documentación del proceso de explotación

Mientras realizas el paso anterior (explotación), **documentá cada acción y resultado**. Un buen pentester genera una bitácora clara: qué comandos/exploits lanzó, contra qué IP/puerto, y qué ocurrió. Por ejemplo: “Se ejecutó exploit X contra el servicio Y en el puerto Z, obteniendo una shell remota como usuario www-data”. También anota las versiones de herramientas usadas y los ajustes realizados. Incluye **evidencias gráficas**:

capturas de pantalla de la consola mostrando el exploit en ejecución y la comprobación de que funcionó (por ejemplo, un uname -a en la shell remota obtenida).

Esta documentación alimentará el informe técnico. Debe ser lo suficientemente detallada para que otro técnico pueda seguir lo que hiciste. Incluye: descripción de la vulnerabilidad explotada, pasos para reproducirla, nivel de acceso logrado, y potenciales impactos (¿podrías escalar privilegios desde esa shell? ¿robar información sensible?). Incluso si decides no explotar al máximo la vulnerabilidad (por seguridad), explica qué *podría* hacer un atacante real partiendo de ese punto.

Recuerda que en un **informe de pentesting** se suelen incluir secciones como *Metodología, Hallazgos y Evidencias*. Aquí es donde este registro detallado encaja. Un ejemplo de redacción podría ser:

- Vulnerabilidad: Credenciales por defecto en MySQL (usuario root sin contraseña).
- Impacto: Acceso total a la base de datos y posibilidad de ejecución de comandos mediante funciones de sistema.
- Explotación: Se utilizó el cliente MySQL para conectar sin contraseña, obteniendo acceso. Captura 1 muestra conexión exitosa; Captura 2 muestra lectura de tabla de usuarios.
- Riesgo: Crítico. Un atacante podría exfiltrar o borrar datos, instalar backdoors en el sistema a través de comandos desde la base de datos, etc.

Este nivel de detalle asegura que el valor de la prueba de intrusión quede claro para terceros.

#### Paso 5: Aplicación de correcciones y refuerzo tras la prueba

Después de demostrar la vulnerabilidad, procede a **corregirla de inmediato** en el sistema para eliminar el riesgo. La remediación debe alinearse con la naturaleza de la vulnerabilidad:

- Si fue una **configuración débil**, corrígela: por ejemplo, asigna una contraseña fuerte al usuario root de MySQL y a cualquier otro usuario sin clave; deshabilita la cuenta FTP anónima o restringe sus permisos; ajusta SSH para no permitir autenticación por contraseña o al menos configura MaxAuthTries bajo y PermitRootLogin no.
- Si fue un **software desactualizado vulnerable**, actualiza ese software a la versión parcheada donde la vulnerabilidad esté resuelta, o si no es posible actualizar de inmediato (por compatibilidad), implementa mitigaciones: por ejemplo, si Apache viejo tiene RCE, podrías aplicar reglas WAF mod\_security como mitigación temporal, o restringir acceso al recurso vulnerable.

- Si el servicio en cuestión **no es necesario**, la medida más segura es deshabilitarlo completamente. Por ejemplo, si el FTP solo estaba abierto por defecto pero no se usará, mejor apagarlo (el servicio más seguro es el que no existe).
- Aplica nuevamente principios de *hardening*: firewall (cerrar el puerto involucrado a menos que sea imprescindible abrirlo), revisión de permisos. En el caso de MySQL, además de la contraseña, podrías limitar que root solo conecte desde localhost si es un servidor dedicado.

Tras aplicar la corrección, **vuelve a ejecutar pruebas** para verificar que la vulnerabilidad ya no es explotable. Ejemplo: intenta nuevamente acceder con la contraseña vacía – debe fallar; ejecuta Nmap script vuln de nuevo – debe mostrar la vulnerabilidad como parchada o no encontrarla. Esta verificación garantiza que las medidas fueron efectivas.

Además, considera incorporar soluciones de seguridad adicionales relacionadas con la vulnerabilidad: si fue falta de detección, quizás implementar un IDS/IPS o monitorizar más esos servicios. Si fue un tema de gestión de parches, mejorar el proceso de actualización en la empresa. Todas estas reflexiones son valiosas para el informe final.

#### **Paso 6: Informe técnico de pentesting (vulnerabilidad y corrección)**

Prepara un **Informe de Pruebas de Penetración** enfocado en la vulnerabilidad descubierta y solucionada en esta fase. Este informe, a diferencia del forense, está más orientado a la **auditoría proactiva** y debe contener:

- **Resumen ejecutivo:** breve descripción de la prueba realizada, la vulnerabilidad encontrada y su severidad, con conclusiones en lenguaje llano (pensado para directivos). Por ejemplo: “Se identificó una vulnerabilidad crítica en el servidor (contraseña por defecto en MySQL) que permitía a un atacante tomar control de la base de datos. Fue explotada en un entorno controlado y se aplicaron correcciones de inmediato, eliminando el riesgo.”
- **Alcance y objetivo:** explica que se realizó un escaneo de vulnerabilidades post-incidente, qué activos se probaron (p. ej., servidor X) y el objetivo de la fase (identificar y remediar vulnerabilidades remanentes).
- **Metodología:** describe brevemente cómo se llevó a cabo la evaluación (herramientas Nmap para escaneo, qué scripts o técnicas se usaron, etc.), siguiendo quizás estándares como OWASP Testing Guide o PTES de manera simplificada.
- **Hallazgos (Vulnerabilidades):** aquí detalla cada vulnerabilidad encontrada, aunque te centres en la explotada. Para cada una, incluye: descripción, nivel de riesgo, sistema/servicio afectado, evidencia. Especialmente documenta la vulnerabilidad explotada: incluye capturas de Nmap mostrando el servicio

vulnerable, captura(s) de la explotación (como prueba de concepto), y explica el impacto. Otras vulnerabilidades halladas (aunque no explotadas) también mencionálas con su riesgo y recomendación, para dar un panorama completo al cliente/lector.

- **Proceso de explotación:** detalla el paso a paso que ejecutaste para explotar la vulnerabilidad principal, como ya describimos en “Documentación del proceso” (Paso 4). Incluye las evidencias recolectadas (screenshots, logs). Esto le da credibilidad al hallazgo, demostrando que no es solo teórico.
- **Solución aplicada:** explica qué hiciste para resolver la vulnerabilidad (cambiar config, actualizar, etc.) y si quedaron medidas pendientes. Es útil incluir también **recomendaciones adicionales** para prevenir futuras ocurrencias, por ejemplo: “Implementar política de no usar credenciales por defecto, procedimientos de hardening para nuevos servicios, monitoreo de configuraciones”.
- **Conclusiones:** resume el estado final: “Tras la corrección, el sistema ya no presenta la vulnerabilidad X; el riesgo asociado ha sido mitigado. Se recomiendan pruebas de penetración periódicas e integración de estas mejoras en políticas de seguridad.”

Este informe técnico servirá como evidencia de la mejora continua de la seguridad del sistema. Debe ser claro y organizado, utilizando tablas o listas para enumerar vulnerabilidades, y priorizando la información (primero lo más crítico). Incluye referencias a estándares si aplicable (por ejemplo, si la vulnerabilidad corresponde a OWASP Top 10, menciónalo). Mantén un tono objetivo y profesional, evitando señalar culpables y enfocándote en soluciones. Si el proyecto lo requiere, prepara también una **presentación ejecutiva** que resuma este informe en diapositivas sencillas para exponer ante un público no técnico (esto puede integrarse con la presentación global del proyecto).

## 5. Fase 3: Plan de respuesta a incidentes y SGSI (Seguridad Gestiónada – ISO 27001)

**Objetivo:** Desarrollar un **Plan formal de Respuesta a Incidentes** y sentar las bases de un **Sistema de Gestión de Seguridad de la Información (SGSI)** alineado con ISO 27001, incluyendo medidas de protección de datos (backups, cifrado, DLP). Se trata de pasar de una reacción ad-hoc a incidentes a un enfoque estructurado y preventivo, incorporando las mejores prácticas y estándares internacionales (NIST, ISO).

### Paso 1: Creación del Plan de Respuesta a Incidentes (NIST SP 800-61)

Elabora un **Plan de Respuesta a Incidentes** tomando como referencia el marco del NIST SP 800-61 (Guía para manejo de incidentes de seguridad informática). Este estándar define el ciclo de vida de respuesta en **cuatro fases fundamentales**:

## **Preparación; Detección y Análisis; Contención, Erradicación y Recuperación; y Actividad Posterior al Incidente.**

*Figura: Ciclo de vida de respuesta a incidentes según NIST SP 800-61. En la fase de Preparación se establecen las capacidades y procedimientos; en Detección/Análisis se identifica y confirma el incidente; en Contención, Erradicación y Recuperación se aísla la amenaza, se elimina y se restauran sistemas; y en Actividades Post-Incidente se extraen lecciones y se mejoran los procesos.*

Tu plan debe describir cada una de estas fases aplicado a la organización del proyecto, especificando **qué hacer, cómo y quién lo hace** en caso de incidentes. Puntos clave a incluir:

- **Preparación:** Define las políticas y herramientas que estarán en su lugar antes de un incidente. Por ejemplo, tener actualizado el inventario de activos, instalar sistemas de monitoreo (SIEM, IDS/IPS), establecer acuerdos de nivel de servicio para respuesta, formar el *CSIRT (Computer Security Incident Response Team)* con roles y responsabilidades claras. También, enumerar las medidas de seguridad preventivas ya implementadas (firewalls, antivirus, capacitación a usuarios para phishing, etc.). La preparación incluye asegurar que existen **procedimientos escritos** y recursos para responder (herramientas forenses, contactos de emergencia, entornos de cuarentena, comunicaciones definidas).
- **Detección y Análisis:** Establece cómo se identificarán potenciales incidentes. Describe las fuentes de detección (logs, alertas del SIEM, reportes de usuarios, monitoreo de integridad) y los criterios para considerar algo un “incidente” (por ejemplo, varios antivirus activados, defacement de una web, ransomware activado, etc.). Define pasos para la *confirmación* del incidente: recolectar indicadores, analizar logs (referenciando lo aprendido en Fase1), hacer análisis de memoria o tráfico si aplica. Importante: incluye **árbol de decisión** o flujo de trabajo de notificación interna una vez detectado (a quién se informa dentro de cuánto tiempo) y si es necesario comunicar a autoridades regulatorias (en caso de data breach de información personal, por ejemplo, según leyes).
- **Contención, Erradicación y Recuperación:** Esta fase es crítica y debes detallar procedimientos específicos. **Contención:** acciones inmediatas para confinar el daño y evitar propagación. Por ejemplo, aislar hosts comprometidos de la red (quitar cable o VLAN de cuarentena), cerrar puertos o servicios vulnerados, cambiar credenciales comprometidas, etc. El plan debe ofrecer opciones de contención rápida (a corto plazo, e.g. desconectar servidor) y contención más permanente (p. ej., montar un sistema limpio paralelo y migrar servicios). **Erradicación:** instrucciones para eliminar la amenaza: quitar malware, formatear y reinstalar sistemas afectados de ser necesario, aplicar parches en todos los sistemas similares, *verificar dos veces* que no queden puertas traseras

(volviendo a ejecutar herramientas tipo rkhunter). **Recuperación:** indica cómo restaurar los sistemas a operación normal de forma segura. Incluye restaurar desde backups limpias, verificar integridad de sistemas restaurados, monitorear intensamente por un tiempo tras la reincorporación en producción, y comunicar a usuarios si hay interrupciones. Es recomendable definir **puntos de control** (checkpoints) para decidir cuándo un sistema puede considerarse seguro para volver a la red.

- **Actividades Post-Incidente:** Describe las tareas posteriores una vez resuelto el incidente: realizar una **reunión de lecciones aprendidas**, donde el equipo analice qué fue lo que sucedió, qué se hizo bien/mal en la respuesta y qué mejorar. Documentar formalmente el incidente en un informe post-mortem. Actualizar los planes de respuesta y seguridad con base en lo aprendido. También considerar si se debe preservar evidencia por cierto tiempo (por razones legales o auditorías). Esta fase incluye evaluar si se necesita notificar a clientes o entes externos (p. ej. en caso de fuga de datos personales, notificar a la agencia de protección de datos dentro de X horas según GDPR, etc., aunque sea un ejercicio, menciona el cumplimiento legal).

En la elaboración del plan, asigna **responsables** a cada acción. Por ejemplo: “El Líder de Incidentes (CISO) decide si se apaga un servidor comprometido tras consultarla con Dirección, el admin de sistemas X ejecuta la desconexión; el equipo de redes aplica bloqueos en firewall”, etc. Incluye información de contacto de equipo de respuesta (aunque sean roles ficticios para el proyecto). Un buen plan también contiene *listas de verificación* (checklists) para facilitar seguir el procedimiento en el estrés de un incidente.

Al final, tendrás un documento que actúa como **guía paso a paso para futuros incidentes**, reduciendo la improvisación. Esto muestra madurez en la gestión de seguridad. Puedes apoyarte en plantillas públicas o guías de SANS/NIST para darle formato. El plan de respuesta es un entregable fundamental y puede formar parte del SGSI.

## Paso 2: Desarrollo de procedimientos detallados (identificación, contención, erradicación, recuperación)

Aunque ya hemos delineado las fases NIST en el plan, vale la pena preparar **procedimientos específicos** para tipos de incidentes probables, asegurando que en cada etapa haya instrucciones claras. Por ejemplo:

- **Procedimiento de Identificación:** Incluir un playbook para detección de, digamos, malware/ransomware vs. intrusión en servidor web vs. pérdida de dispositivo corporativo. Cada escenario tendrá indicadores distintos. Establece qué hacer en cada caso para confirmar el incidente. Un procedimiento general

es “Alerta → Reúne datos (logs, alertas AV) → Analiza si hay falsa alarma o incidente real → Clasifica la severidad (bajo, medio, alto) → Activa respuesta según severidad”.

- **Procedimiento de Contención Inmediata:** Podría ser un check-list: 1) Aislamiento red (desconectar cable o apagar interfaz), 2) Capturar memoria (si es intrusión compleja) antes de apagar nada, 3) Redirigir tráfico malicioso (si hay un servidor de respaldo) etc. Incluye decisiones como “¿apagar el servidor o dejarlo encendido para análisis?”. Según NIST, se debe balancear necesidad de contener rápido vs preservar evidencias. Indica criterios para decidir (por ej., si ransomware en curso → apagar equipo ya; si intruso silencioso → quizás monitorizar un poco antes de actuar).
- **Procedimiento de Erradicación:** Podría incluir pasos como “Escanear en busca de rootkits (rkhunter, etc.), Formatear o reinstalar sistema comprometido desde cero, *Cambiar todas las contraseñas* posiblemente comprometidas, Revisar sistemas relacionados por infección (movimiento lateral)”. Si el incidente fue malware, detalla limpiar/eliminar malware en PCs; si fue un exploit en servidor, detalla aplicar parche en ese servidor y en otros con misma vulnerabilidad.
- **Procedimiento de Recuperación:** Por ejemplo: “Obtener última copia de seguridad limpia – verificar integridad – restaurar en servidor nuevo – aplicar parches antes de conectar a la red – pruebas de funcionalidad – reincorporar servicios gradualmente”. Incluir chequeos post-recuperación, como monitorear tráfico inusual o logs intensivamente la primera semana, para asegurarse de que la amenaza no reaparece.

Estos procedimientos pueden ser anexos del Plan principal. El nivel de detalle es importante: en medio de un incidente real, los encargados deberían poder seguir estos pasos casi como una receta. **Incorpora medidas de seguridad de datos** aquí también: por ejemplo, durante recuperación, usar *backups cifradas* y verificar firmas de integridad para asegurar que no se está restaurando algo ya comprometido.

### **Paso 3: Implementación de mecanismos de protección de datos (backups, cifrado, control de accesos)**

Una parte esencial de un buen plan de seguridad es proteger la **confidencialidad, integridad y disponibilidad de los datos** de la organización en todo momento. Para ello:

- **Política de Backups Seguros:** Asegura que existen copias de seguridad periódicas de la información crítica. Define la frecuencia (diaria, semanal, etc. según el valor del dato), retención (¿cuánto tiempo se guardan?), y algo crucial: almacena backups **off-site** (fuera del servidor principal) y preferiblemente *desconectados* (offline) o en una nube segura. Esto protege contra incidentes

como ransomware, donde backups locales conectadas podrían cifrarse también. Establece procedimientos de prueba de restauración periódica para validar que los backups funcionan. Además, **cifra los backups** con herramientas o software especializado, de modo que si caen en manos no autorizadas, no puedan leerse. El cifrado añade una capa de protección sobre los controles de acceso básicos a las copias.

- **Cifrado de Datos Sensibles:** Implementa cifrado para datos en **reposo** (en discos/BD) y en **tránsito** (comunicaciones). Por ejemplo, usar SSL/TLS para todos los servicios web, VPN para accesos remotos, cifrado de discos o al menos de contenidos especialmente sensibles (p. ej. bases de datos de clientes cifradas a nivel de campo o tablespace). Esto garantiza que aunque haya una filtración, los datos no se exponen en texto plano. Define políticas de manejo de claves de cifrado (almacenarlas de forma segura, rotarlas periódicamente).
- **Control de Accesos y Gestión de Identidades:** Aplica el principio de *privilegios mínimos*: cada usuario/servicio solo con los permisos necesarios. Implementa controles de acceso fuertes: autenticación multifactor para accesos críticos, uso de mecanismos como *LDAP/Active Directory* centralizado para controlar usuarios, políticas de bloqueo de cuenta tras intentos fallidos (si no se usan herramientas como fail2ban), segregación de funciones (evitar que un mismo usuario tenga control total sin supervisión). Documenta estas medidas en políticas claras de control de acceso.
- **Protección de la integridad:** Considera usar herramientas de monitoreo de integridad de ficheros (FIM) como *Tripwire/AIDE*, que avisen si archivos críticos cambian ( posible indicador de sabotaje o malware). También firma digitalmente los archivos sensibles o logs para poder detectar alteraciones.
- **Disponibilidad y continuidad:** Además de backups, planifica mecanismos de tolerancia a fallos: quizás un sitio alterno (cold/hot standby), o al menos recursos en la nube para levantar servicios en caso de desastre. Todo esto debería estar en un **Plan de Recuperación ante Desastres (DRP)** complementario.

Incorpora estos mecanismos en el **SGSI** como controles específicos. Por ejemplo, ISO 27001 en su anexo de controles (ISO 27002) incluye controles sobre copias de seguridad (A.12.3) y cifrado (A.10.1). Menciona en tu documentación qué controles aplicaste o recomendarías conforme a esas buenas prácticas.

## **5. Fase 3: Plan de respuesta a incidentes y SGSI**

La ISO 27001 establece los **requisitos para un Sistema de Gestión de Seguridad de la Información (SGSI)** robusto, que no es más que la organización de la seguridad en la empresa de forma **documentada, medible y mejorable continuamente**. Para desarrollar el SGSI en el proyecto, considera los siguientes pasos:

- **Compromiso de la dirección y alcance:** Define el alcance del SGSI (¿aplica a toda la organización? ¿solo a ciertos departamentos o tipos de datos?). En este proyecto académico podrías asumir que cubre la infraestructura del proyecto. Es importante contar (en la vida real) con apoyo de la alta dirección, pues se necesitarán recursos y cambio cultural.
- **Políticas de Seguridad de la Información:** Redacta una política matriz de seguridad que enuncie los objetivos de la organización en protección de la información, responsabilidades generales y el compromiso con el cumplimiento de estándares. Luego, desarrolla políticas específicas (pueden mencionarse aunque no escribir las completas): política de control de accesos, política de uso aceptable, política de clasificación de la información, política de manejo de incidentes (el plan NIST sería parte), política de continuidad del negocio, etc. Estas políticas proveen la guía para todos en la organización sobre cómo manejar la seguridad.
- **Análisis y valoración de Riesgos:** Este es el corazón de ISO 27001. Realiza un **análisis de riesgos** identificando los **activos** (ej: servidores, datos de clientes, procesos de negocio), las **amenazas y vulnerabilidades** asociadas, y evaluando el impacto y probabilidad de cada riesgo. Por ejemplo: activo “Servidor web”, amenaza “ataque XSS”, vulnerabilidad “validación insuficiente en aplicación web”, impacto “filtración de datos de usuarios”, probabilidad “media”. Cada riesgo se califica (alto/medio/bajo) para priorizar. La norma ISO 27005 puede guiar en métodos de análisis de riesgo. Como recomienda la literatura, *no implementes controles sin antes identificar y clasificar los riesgos*: primero entiende qué debes proteger y de qué. De hecho, **toda la implementación** del SGSI debe basarse en el tratamiento de los riesgos identificados.
- **Tratamiento de Riesgos:** Por cada riesgo identificado, decide cómo tratarlo: mitigar (implementando controles para reducirlo), transferir (seguros), aceptar (si es bajo) o evitar (dejar de hacer la actividad de riesgo). Documenta un **Plan de Tratamiento de Riesgos** que liga cada riesgo con controles específicos (por ejemplo, riesgo “intrusión externa” mitigado con control “firewall perimetral, monitoreo IDS, hardening de servidores”). La ISO 27001:2022 tiene 93 controles en categorías que puedes usar de referencia. No necesitas implementarlos todos, sino los pertinentes a tus riesgos. Haz una **Declaración de Aplicabilidad** indicando qué controles aplicas y cuáles no (justificando estos últimos).

- **Implementación de Controles y Procedimientos:** Lleva a la práctica los controles elegidos. Aquí es donde mucho de lo realizado en Fases 1 y 2 encaja en un marco formal. Por ejemplo, los mecanismos de respaldo y cifrado son controles implementados; el plan de respuesta a incidentes es otro control; políticas de acceso (MFA, privilegios mínimos) otros; capacitación en concienciación para el personal (phishing, etc.) también es importante – la *seguridad del factor humano* no debe olvidarse (ISO incluye controles de capacitación y concienciación). Documenta procedimientos operativos para respaldar las políticas (ej: procedimiento de alta/baja de usuarios, procedimiento de gestión de parches, etc.).
- **Monitorización y mejora continua (PDCA):** Un SGSI no es estático. Aplica el ciclo **PDCA (Plan-Do-Check-Act)** o *Planificar-Hacer-Verificar-Actuar*, como recomienda ISO 27001. Esto implica: **Planificar** (políticas, objetivos, análisis de riesgos – lo que hicimos), **Hacer** (implementar controles y operar el SGSI), **Verificar** (evaluar el desempeño: por medio de auditorías internas periódicas, revisión de incidentes, métricas de seguridad, cumplimiento de objetivos) y **Actuar** (tomar acciones correctivas/mejoras en base a lo encontrado en “Check”, ajustando políticas, cerrando brechas de control, etc.). Por ejemplo, define KPIs como “% de sistemas con parches al día” o “tiempo promedio de respuesta a incidentes” y mídelos. Realiza auditorías internas al menos anuales para verificar que se cumplen las políticas y procedimientos establecidos, y prepara a la organización para **auditorías de certificación** externas.
- **Documentación y Registro:** ISO 27001 exige mantener ciertos documentos y registros: política de SI, alcance del SGSI, metodología de análisis de riesgos, inventario de activos, evaluación de riesgos, plan de tratamiento, declaración de aplicabilidad, protocolos de formación, resultados de auditorías, revisiones de dirección, etc. Asegúrate de que todo cambio o evento relevante se documenta. Esto no solo es para “pasar la auditoría”, sino que ayuda a la gestión sistemática de la seguridad.

Implementar un SGSI completo es una tarea extensa; en este proyecto, enfócate en demostrar que conoces los elementos esenciales. Por ejemplo, podrías **presentar un Análisis de Riesgos resumido** (tabla de riesgos con su valoración y controles asociados) y **ejemplos de políticas** redactadas. Menciona que la organización debería buscar la **Certificación ISO 27001** una vez implementados los controles, lo que implicará una auditoría independiente que verifique el cumplimiento de todos los requisitos de la norma. La certificación provee garantía externa de que la seguridad se gestiona adecuadamente.

Vale destacar que la **cultura de seguridad** es parte integral del SGSI: todos los empleados deben estar involucrados. Recomienda programas de concienciación

(phishing simulations, boletines, capacitación anual) para que la seguridad sea un esfuerzo colectivo, no solo del departamento de TI.

#### Paso 5: Recomendaciones de Data Loss Prevention (DLP)

Para abordar la prevención de fugas de datos (intencionadas o accidentales), implementa soluciones de **Data Loss Prevention (DLP)**. DLP es un conjunto de herramientas y procesos diseñados para **detectar y prevenir la transferencia no autorizada de información sensible** fuera de la organización. Recomienda lo siguiente:

- **Clasificación de la información:** Define niveles de sensibilidad (p. ej., *Pública, Interna, Confidencial, Secreta*) y clasifica los datos de la empresa en esas categorías. Esto ayuda a aplicar controles DLP más estrictos donde se requiere (por ejemplo, datos clasificados como “Confidencial” no pueden salir sin cifrar).
- **Herramientas DLP:** Sugiere la implementación de software DLP en puntos clave: en puestos de trabajo (endpoint DLP) para monitorear/copiar archivos, en la red (DLP de red) para inspeccionar correos electrónicos, transferencia de archivos, etc., y en servicios en la nube si se utilizan (CASB – Cloud Access Security Broker con políticas DLP). Estas herramientas pueden **monitorizar y bloquear** activamente intentos de enviar información confidencial fuera (por email, USB, uploads web). Por ejemplo, si alguien intenta enviar por correo una base de datos de clientes, el DLP lo detectaría (por patrones como muchas cuentas o números de tarjeta de crédito) y podría bloquear el envío y alertar.
- **Políticas y reglas DLP:** Establece reglas acordes a las necesidades: qué tipo de datos no deben salir y por qué canales. Ejemplos: “No permitir enviar números de tarjeta de crédito sin cifrar por email” (el DLP escanea texto y adjuntos buscando patrones de 16 dígitos), “Bloquear copia de archivos de proyectos confidenciales a pendrives no autorizados”, “Alertar si se imprimen listados masivos de información personal”. Configura el DLP con estos criterios.
- **Registro y alertas:** Asegura que la solución DLP registre los incidentes (quién, qué datos, a dónde intentó enviarlos) y alerte al equipo de seguridad para investigar. Un intento bloqueado podría ser un error de un empleado o un indicio de actividad maliciosa interna (insider threat).
- **Educación:** Acompaña la tecnología con concienciación a los empleados sobre manejo adecuado de información. El DLP también puede *educar* mostrando advertencias a los usuarios (“este documento contiene datos sensibles, confirma si realmente necesitas enviarlo cifrado”).

Con DLP, se busca evitar tanto las fugas accidentales (ej. empleado que envía por equivocación un informe sensible a destinatario equivocado) como las maliciosas (un empleado deshonesto o intruso que roba información). Menciona algunas *mejores*

*prácticas DLP*: identificar los “Crown Jewels” (datos más valiosos), implementar DLP de forma incremental para afinar reglas y minimizar falsas alarmas, y combinarlo con otras estrategias (clasificación, control de accesos) para ser efectivo.

#### **Paso 6: Formalización del plan de recuperación y continuidad del negocio**

Además del Plan de Respuesta técnica (que se enfoca en incidentes de seguridad), deberías recomendar la elaboración de un **Plan de Recuperación ante Desastres (DRP)** y un **Plan de Continuidad de Negocio (BCP)** más amplios. Esto suele alinearse con ISO 22301 pero es complementario a ISO 27001 en cierta medida. Incluye en tu proyecto la idea de:

- **Identificar procesos críticos** de negocio y definir estrategias para mantenerlos operando ante diversos escenarios (no solo ciberataques, sino fallos eléctricos, desastres naturales, errores humanos graves, etc.).
- **Definir RTO/RPO** (Recovery Time Objective, Recovery Point Objective) para sistemas clave, y asegurarse que las estrategias de backup y redundancia cumplen esos objetivos.
- **Procedimientos de recuperación** para escenarios específicos (por ejemplo, “servidor caído completo: restaurar backup en nuevo servidor en X horas”).
- **Pruebas periódicas** de los planes (simulacros de incidente, simulacros de restauración).

Si bien esto excede quizás el alcance inmediato, demostrar que lo consideras dará solidez a tu proyecto.

#### **Paso 7: Presentación ejecutiva de resultados**

Como cierre, prepara una **presentación ejecutiva** que resuma todo el trabajo realizado en las tres fases, para ser expuesta ante un público gerencial o académico. Esta presentación (en formato diapositivas) debe enfocarse en los hallazgos, acciones y recomendaciones en un lenguaje accesible, destacando el valor para la organización. Incluye:

- **Resumen del incidente y respuesta**: qué pasó en el hackeo (Fase 1) y cómo se solucionó; enfatiza que se eliminó la amenaza y se fortaleció el sistema (aprendizaje: importancia de mantener parches, etc.).
- **Resultados del pentesting**: menciona la vulnerabilidad crítica encontrada (Fase 2) y cómo se arregló antes de que fuera explotada maliciosamente; demuestra proactividad. Si hubo otras vulnerabilidades menores, mencionalas brevemente y que también se mitigaron.

- **Mejoras en seguridad implementadas:** una lista breve de todas las mejoras: hardening (firewall, contraseñas), herramientas añadidas (fail2ban, auditd, SIEM si aplica), políticas definidas, etc. Esto muestra cómo el estado de seguridad actual es mucho mejor que al inicio.
- **Plan de Incidentes y SGSI:** explica que ahora la organización cuenta con un plan formal de respuesta a incidentes (listo para enfrentar futuros eventos de forma eficaz siguiendo NIST) y se ha instaurado un proceso continuo de gestión de la seguridad (SGSI ISO 27001) con apoyo de la dirección, análisis de riesgos y políticas en marcha.
- **Beneficios:** resalta beneficios tangibles e intangibles: reducción de riesgo de ciberataques, cumplimiento de estándares (tal vez necesario para clientes o regulaciones), protección de datos de clientes, mayor confianza, etc.
- **Próximos pasos:** recomendar la certificación ISO 27001 (si fuera real, argumentando que el SGSI implementado está listo para auditarse) y mantenimientos futuros (auditorías, actualizaciones, entrenamiento continuo).

Mantén la presentación visualmente clara, con gráficos o diagramas donde sea útil (por ejemplo, un diagrama antes-después de la postura de seguridad, o ilustraciones del ciclo NIST e ISO). No satures de texto; utiliza viñetas concisas. Piensa que esta presentación sería vista por ejecutivos que querrán entender *en términos de negocio* qué se hizo y por qué. Por ejemplo, puedes cuantificar “se redujo la superficie de ataque de X puertos abiertos a Y; se mejoró el tiempo de respuesta a incidentes de posiblemente días a horas; etc.”.

Con esta presentación ejecutiva, completas los entregables del proyecto: **informe forense del incidente, informe técnico de pentesting, plan de respuesta/recuperación y presentación ejecutiva**. Todo en conjunto demostrará una respuesta integral a un incidente de ciberseguridad y la adopción de un modelo de seguridad proactivo y alineado a estándares internacionales.

## **Conclusión y Referencias**

A lo largo de estas fases, hemos aplicado un enfoque profesional para manejar incidentes y mejorar la ciberseguridad de la organización ficticia de 4Geeks Academy. Desde la respuesta forense inmediata hasta la integración de un SGSI, cada paso estuvo respaldado por buenas prácticas de la industria. Se utilizaron herramientas concretas (Nmap, chkrootkit, rkhunter, fail2ban, etc.) y marcos de referencia reconocidos (NIST SP 800-61 para incidentes, ISO 27001 para gestión de la seguridad) para garantizar que las soluciones propuestas no solo resuelven el problema puntual sino que fortalecen la resiliencia a largo plazo.

En resumen, el proyecto finaliza con un servidor saneado y fortificado, procesos documentados para enfrentar futuras amenazas, y una estructura organizativa consciente de la seguridad. Se han entregado los informes requeridos con el máximo rigor técnico y claridad ejecutiva, cumpliendo con el objetivo del proyecto de demostrar competencias en **respuesta a incidentes, análisis de vulnerabilidades y gobierno de la seguridad de la información**.

**Referencias utilizadas:** Las acciones y recomendaciones presentadas se fundamentan en estándares y fuentes reconocidas, entre ellas: guía de recuperación ante servidores comprometidos de Red Hat, prácticas forenses documentadas, documentación de Nmap y blogs de seguridad para detección de vulnerabilidades, lineamientos de informes de pentesting de 4Geeks, el marco de manejo de incidentes del NIST, consejos de expertos en endurecimiento y auditoría (auditd, logs remotos), y definiciones de soluciones DLP, entre otras. Estas referencias se citan a lo largo del texto donde apoyan directamente una afirmación o recomendación específica. Es importante destacar que la ciberseguridad es un campo en constante evolución, por lo que siempre se debe procurar estar actualizado con las últimas amenazas y mejores prácticas más allá de lo cubierto en este documento.

## CONCLUSIONES

### Fase 3: Plan de Respuesta a Incidentes y Diseño del SGSI (ISO/IEC 27001)

#### 5.1 Introducción

Como parte del proceso de recuperación y aseguramiento del entorno tras un incidente crítico en el servidor de 4Geeks Academy, se desarrolló un Sistema de Gestión de Seguridad de la Información (SGSI) basado en los principios de la norma ISO/IEC 27001:2022 y complementado con las recomendaciones de NIST SP 800-61 para respuesta a incidentes. Este sistema busca asegurar la continuidad operativa, proteger la información crítica de la organización y mitigar futuros riesgos cibernéticos.

---

#### 5.2 Alcance y Objetivo

##### Objetivo del SGSI

Establecer un marco técnico y organizativo que garantice la confidencialidad, integridad y disponibilidad de la información tratada por los sistemas clave de 4Geeks Academy, minimizando el impacto de amenazas actuales y emergentes.

##### Alcance

Este SGSI se aplica a:

- El servidor Debian comprometido y su red circundante.
- Servicios críticos: SSH, Apache, MySQL, FTP y WordPress.

- Administración y operación del sistema desde entornos Kali Linux y otras estaciones de análisis y pentesting.
- Implementación de medidas técnicas y organizativas tras el incidente.

Exclusiones: Sistemas fuera del laboratorio y máquinas sin control administrativo directo.

---

### 5.3 Fase de Preparación (Plan – PDCA)

- Política de seguridad redactada para definir los objetivos y principios del tratamiento de la información.
- Inventario de activos: Servidor, servicios expuestos, usuarios y configuraciones sensibles.
- Definición de roles:
  - *Analista de ciberseguridad* (rol asumido): encargado del análisis, mitigación y fortalecimiento.
  - *Administrador del sistema*: responsable de aplicar cambios y controlar accesos.
- Simulacro de respuesta a incidentes programado con registros de log y líneas de tiempo reales.
- Se establecen controles preventivos como firewall (UFW), Fail2Ban, y Wazuh como agente de monitoreo.

---

### 5.4 Detección, Contención, Erradicación y Recuperación (Do)

Se aplica el ciclo de respuesta de NIST SP 800-61:

- Detección y análisis: a través de logs, escaneos (Nmap, LinPEAS), y forense con chkrootkit y rkhunter.
- Contención inmediata:
  - Aislamiento lógico del servidor.
  - Detención de servicios como FTP y Apache.
  - Registro forense de evidencias (clonado con dd, análisis en Autopsy).
- Erradicación:
  - Eliminación de cuentas sospechosas y cron jobs.

- Limpieza de backdoors y configuraciones anómalas.
  - Recuperación:
    - Reinicio de servicios validados.
    - Fortalecimiento de configuración (SSH, permisos, WordPress, contraseñas).
    - Aplicación de parches y reinicio controlado.
- 

## 5.5 Verificación y Evaluación (Check)

Se realiza una revisión posterior:

- Checklist de seguridad post-hardening completado.
  - Validación de logs limpios y puertos cerrados no necesarios.
  - Confirmación de bloqueo de vectores anteriores (FTP anónimo, MySQL sin contraseña).
  - Informe técnico consolidado con cronología, hallazgos, y acciones correctivas.
- 

## 5.6 Mejora continua (Act)

Se define un plan de mejora:

- Implementación continua de auditorías técnicas con herramientas automáticas.
  - Revisión trimestral del estado de seguridad.
  - Capacitación técnica al personal encargado del servidor.
  - Documentación de todos los cambios en un registro de cambios y eventos.
- 

## 5.7 Gestión de Riesgos (Resumen ISO 27005)

Aplicación básica del proceso de evaluación de riesgos:

Activo crítico	Amenaza	Vulnerabilidad	Impacto	Probabilidad	Riesgo	Control aplicado
Servidor Debian	Acceso no autorizado	FTP anónimo habilitado	Alto	Alta		Cierre del Crítico puerto, revisión vsftpd

Activo crítico	Amenaza	Vulnerabilidad	Impacto	Probabilidad	Riesgo	Control aplicado
Base de datos MySQL	Escalada de Usuario sin privilegios	Contraseña	Alto	Media	Alto	Borrado usuario + contraseña fuerte
WordPress	RCE o XSS	Plugins desactualizados	Alto	Alta	Crítico	Eliminación + actualización

---

## 5.8 Controles aplicados (resumen ISO 27002 y CIS Controls)

- Técnicos: UFW, fail2ban, cifrado de contraseñas, permisos mínimos.
  - Organizativos: Roles de acceso, logs, snapshots y backups.
  - Humanos: Concienciación sobre contraseñas, validación manual de accesos críticos.
- 

## 5.9 Conclusiones SGSI

El sistema ahora:

- Cuenta con una base de seguridad estructurada.
  - Aplica controles preventivos y correctivos basados en riesgos reales.
  - Está documentado, trazable y listo para auditorías técnicas.
  - Incorpora monitoreo activo y capacidad de respuesta futura.
- 

## 5.10 Recomendaciones futuras

- Integrar SIEM centralizado (como Wazuh Manager + Kibana).
- Desplegar monitoreo de integridad (AIDE o Tripwire).
- Automatizar copias de seguridad externas cifradas.
- Formalizar políticas de contraseñas y procedimientos de alta/baja de usuarios.

## **6. Conclusiones Generales**

### **6.1 Estado Inicial del Sistema**

El servidor presentaba múltiples vectores de ataque:

- FTP con acceso anónimo y sin chroot.
- SSH con login root habilitado y contraseñas débiles.
- Plugins de WordPress obsoletos.
- Logs alterados o eliminados.
- Presencia de cuentas sospechosas y cron jobs persistentes.

### **6.2 Resultados Técnicos Alcanzados**

- Eliminación total de amenazas detectadas.
- Fortalecimiento del sistema conforme a mejores prácticas (NIST, CIS).
- Implementación de firewall, autenticación fuerte y controles mínimos de acceso.
- Documentación completa de la intrusión, mitigación, y recuperación.

### **6.3 Valor Añadido del Análisis Forense y el Pentesting**

- Se logró reconstruir el incidente, evidenciar vectores de intrusión y realizar pruebas controladas sin comprometer más el sistema.
- La fase ofensiva permitió encontrar una segunda vulnerabilidad crítica, demostrando la importancia de auditorías internas periódicas.
- Se preparó un plan de respuesta que puede adaptarse a sistemas reales.

### **6.4 Lecciones Aprendidas**

- La seguridad no es un producto, sino un proceso continuo.
- Las configuraciones por defecto representan amenazas latentes si no se revisan.
- La trazabilidad, las bitácoras y el versionado de configuraciones son claves para la defensa.

## **7. Recomendaciones**

### **7.1 Medidas Inmediatas a Implementar**

- Activar autenticación de doble factor (2FA) en todos los accesos SSH y servicios de administración.
- Ejecutar escaneos automatizados con herramientas como Lynis, ClamAV, rkhunter en cron semanal.
- Forzar rotación de contraseñas y aplicar gestión centralizada de usuarios.

### **7.2 Mejoras Estratégicas Mediano Plazo**

- Implementar un sistema de monitoreo activo y correlación de eventos (propuesta: Wazuh, ELK, Graylog).
- Segmentar la red del servidor para aislar servicios vulnerables.
- Desplegar un entorno de reproducción para pruebas de cambios antes de aplicarlos en producción.

### **7.3 Recomendaciones para el Personal Técnico**

- Capacitación en gestión segura de WordPress, MySQL y SSH.
- Formación periódica en análisis forense básico y respuesta a incidentes.
- Procedimiento formal de control de cambios con documentación técnica de respaldo.

## **8. Glosario**

### **\*\*Rootkit\*\***

Conjunto de herramientas diseñadas para ocultar la presencia de software malicioso en un sistema comprometido.

### **\*\*NIST\*\***

National Institute of Standards and Technology, organismo estadounidense que emite estándares clave de ciberseguridad como el SP 800-61.

**\*\*SGSI (ISMS)\*\***

Sistema de Gestión de Seguridad de la Información, marco estructurado basado en ISO/IEC 27001 para proteger activos de información.

**\*\*IOC (Indicators of Compromise)\*\***

Evidencia digital que indica que un sistema ha sido comprometido (direcciones IP, hashes, nombres de archivos, etc.).

**\*\*Hardening\*\***

Proceso de asegurar un sistema minimizando su superficie de ataque mediante configuración segura y eliminación de servicios innecesarios.

**\*\*Escalada de Privilegios\*\***

Técnica utilizada por un atacante para obtener mayores niveles de acceso dentro de un sistema comprometido.

**\*\*Fail2Ban\*\***

Herramienta que analiza logs y bloquea automáticamente direcciones IP sospechosas, generalmente usada para proteger SSH.

**\*\*chkrootkit / rkhunter\*\***

Herramientas forenses usadas para detectar rootkits, backdoors y exploits conocidos en sistemas Unix/Linux.

**\*\*ISO/IEC 27001\*\***

Norma internacional que establece los requisitos para implementar un SGSI eficaz.

**\*\*Autopsy\*\***

Aplicación forense gráfica basada en The Sleuth Kit (TSK), utilizada para análisis de imágenes de disco.

**\*\*WPScan\*\***

Escáner de vulnerabilidades para sitios WordPress.

**\*\*UFW (Uncomplicated Firewall)\*\***

Interfaz simplificada para manejar iptables y configurar reglas de cortafuegos en sistemas Linux.

## **9. Referencias**

- [1] NIST. (2012). Computer Security Incident Handling Guide (SP 800-61r2).
- [2] ISO/IEC 27001:2022. Information Security, Cybersecurity and Privacy Protection — Information Security Management Systems.
- [3] The Debian Administrator's Handbook. <https://debian-handbook.info>
- [4] chkrootkit – Locally checks for signs of a rootkit. <http://www.chkrootkit.org/>
- [5] rkhunter – Rootkit Hunter. <https://github.com/rkhunter/rkhunter>
- [6] Nmap – Network Mapper. <https://nmap.org>
- [7] WPScan – WordPress Security Scanner. <https://wpSCAN.com>
- [8] fail2ban Documentation. <https://www.fail2ban.org>
- [9] CIS Controls v8. Center for Internet Security.  
<https://www.cisecurity.org/controls>
- [10] OWASP. Top 10 Web Application Security Risks. <https://owasp.org>

## **10. Anexos**

### **\*\*Anexo A – Salidas de herramientas\*\***

- Nmap: escaneo completo (`nmap -sC -sV -p- IP`)
- chkrootkit y rkhunter: salidas completas
- linPEAS (resumen de vulnerabilidades locales)

### **\*\*Anexo B – Scripts utilizados\*\***

- recolectar\_evidencia.sh
- clonar\_disco.sh
- ftp\_check.sh

### **\*\*Anexo C – Capturas de pantalla\*\***

- FTP vulnerable
- Configuración antes/después (SSH, UFW)
- Eliminación de usuarios maliciosos

### **\*\*Anexo D – Evidencia de mitigación\*\***

- Checklist técnico firmado
- Servicios activos tras hardening (`systemctl`, `ss`, `nmap`)

### **\*\*Anexo E – Imagen de disco\*\***

- Muestra hash SHA256 del archivo `imagen.dd`
- Tamaño, ruta y resumen de análisis en Autopsy