PROYECTO FINAL DE CIBERSEGURIDAD

SANTIAGO CARAVACA ORTIN

Contenido

Introducción

Este informe detalla el proceso llevado a cabo para la restauración y protección de un servidor crítico comprometido dentro del entorno de 4Geeks Academy. Asumiendo el rol de analista de ciberseguridad, el objetivo principal de este proyecto es restablecer la seguridad del sistema, corregir las vulnerabilidades explotadas y garantizar el funcionamiento óptimo del servidor.

El ejercicio se ha desarrollado a través de un enfoque estructurado en tres fases, diseñado para evaluar las habilidades en análisis forense, detección de vulnerabilidades y respuesta ante incidentes:

- **Fase 1 - Corrección de un Hackeo:** Se inicia con un análisis forense exhaustivo para identificar el vector de ataque y las vulnerabilidades explotadas. El objetivo de esta fase es bloquear el *exploit*, revertir las acciones del atacante y mitigar la escalada del incidente.
- **Fase 2 - Detección y Corrección de una Nueva Vulnerabilidad:** En esta etapa, se realiza un escaneo proactivo del sistema para descubrir una vulnerabilidad adicional, diferente a la del incidente inicial. Se documenta la explotación controlada de dicha vulnerabilidad para entender su impacto y se aplican las medidas correctivas necesarias.
- **Fase 3 - Plan de Respuesta a Incidentes y Certificación:** La fase final se centra en la planificación estratégica de seguridad a largo plazo. Se diseña un Plan de Respuesta a Incidentes basado en la guía NIST SP 800-61 y se desarrolla un Sistema de Gestión de Seguridad de la Información (SGSI) conforme a la norma ISO 27001, incluyendo políticas de Prevención de Pérdida de Datos (DLP).

Herramientas utilizadas

Para realizar los procesos llevados a cabo y redactados en este informe se han utilizado diferentes herramientas, tanto desde la maquina Kali como desde la propia maquina Debian (objeto del proyecto):

1. Nmap

- **Categoría:** Escáner de puertos y descubrimiento de red.
- **¿Qué es?** Nmap (Network Mapper) es una herramienta de código abierto utilizada para el descubrimiento de redes y la auditoría de seguridad. Permite a los administradores de sistemas y a los profesionales de la seguridad explorar las redes, descubrir hosts y servicios, e identificar posibles vulnerabilidades.
- **¿Para qué sirve?** Sirve para una amplia variedad de tareas de seguridad y administración de redes, incluyendo:
    - **Descubrimiento de hosts:** Identificar qué dispositivos están en línea en una red.
    - **Escaneo de puertos:** Determinar qué puertos están abiertos en los hosts y, por lo tanto, qué servicios están escuchando.

    - **Detección de versiones:** Identificar el software y la versión de los servicios que se ejecutan en los puertos abiertos.
    - **Detección de sistemas operativos:** Adivinar el sistema operativo de un host remoto basándose en las respuestas de la pila TCP/IP.
    - **Escaneo de vulnerabilidades básicas:** Utilizar scripts (NSE - Nmap Scripting Engine) para detectar vulnerabilidades comunes o configuraciones incorrectas.
- **Uso común:** Auditorías de seguridad, pruebas de penetración, inventario de red, monitoreo de cambios en la red.

2. Rkhunter

- **Categoría:** Herramienta de detección de rootkits y seguridad del sistema.
- **¿Qué es?** Rkhunter (Rootkit Hunter) es un programa de escaneo de seguridad de código abierto para sistemas tipo Unix. Está diseñado para escanear en busca de rootkits, puertas traseras (backdoors), exploits locales y otras vulnerabilidades o archivos maliciosos.
- **¿Para qué sirve?** Su función principal es ayudar a los administradores a detectar software malicioso que podría estar intentando ocultarse en el sistema. Realiza comprobaciones de bajo nivel como:
    - Comparar sumas de verificación (checksums) de archivos importantes con bases de datos conocidas de firmas de malware.
    - Buscar módulos de kernel ocultos o modificados.
    - Revisar permisos de archivos y directorios.
    - Detectar configuraciones sospechosas en el sistema.
- **Uso común:** Auditar la seguridad de un servidor, realizar comprobaciones periódicas para asegurar la integridad del sistema después de una posible intrusión.

3. Chkrootkit

- **Categoría:** Herramienta de detección de rootkits.
- **¿Qué es?** Chkrootkit es una herramienta de línea de comandos más simple y ligera que Rkhunter, también diseñada para detectar rootkits instalados en sistemas tipo Unix. Es un script de shell que inspecciona binarios del sistema para encontrar signos de modificación por parte de un rootkit.
- **¿Para qué sirve?** Su propósito es escanear los archivos del sistema y la memoria en busca de firmas o comportamientos asociados con rootkits conocidos. Busca principalmente:
    - Archivos de sistema modificados o reemplazados.
    - Procesos ocultos.
    - Puertos de red en modo sigiloso (stealth mode).

- **Ventajas:** Es fácil de usar y rápido para realizar comprobaciones básicas.
- **Uso común:** Primera línea de defensa para detectar infecciones por rootkits, parte de un conjunto de herramientas para la respuesta a incidentes.

4. Journalctl

- **Categoría:** Herramienta de visualización y consulta de registros del sistema.
- **¿Qué es?** journalctl es una utilidad de línea de comandos en sistemas Linux que utilizan systemd (la mayoría de las distribuciones modernas como Ubuntu, Fedora, Debian, CentOS). Permite consultar y ver los registros (logs) recogidos por el servicio systemd-journald.
- **¿Para qué sirve?** Sustituye o complementa los archivos de registro tradicionales (/var/log/syslog, etc.) centralizando todos los mensajes del kernel, los procesos de arranque, los servicios, las aplicaciones y los usuarios. Permite:
    - Ver todos los registros del sistema.
    - Filtrar registros por servicio, unidad, prioridad, tiempo, PID, etc.
    - Ver los registros en tiempo real (-f para follow).
    - Acceder a registros persistentes o solo los del arranque actual.
- **Uso común:** Diagnóstico de problemas del sistema, depuración de servicios, auditoría de eventos del sistema, análisis de comportamiento del sistema.

5. Grep

- **Categoría:** Herramienta de filtrado y búsqueda de texto.
- **¿Qué es?** grep (Global Regular Expression Print) es una utilidad de línea de comandos muy potente y fundamental en sistemas Unix/Linux para buscar patrones de texto dentro de archivos o flujos de datos.
- **¿Para qué sirve?** Su función principal es filtrar líneas de texto que coinciden con un patrón específico (que puede ser una cadena de texto simple o una expresión regular compleja).
    - **Búsqueda en archivos:** Encontrar líneas que contengan ciertas palabras o frases en uno o varios archivos.
    - **Filtrado de salida:** Se utiliza muy a menudo en combinación con el "pipe" (|) para filtrar la salida de otros comandos, por ejemplo, ls -l | grep .txt para encontrar solo archivos .txt.
    - **Búsqueda recursiva:** Buscar patrones en archivos dentro de directorios y subdirectorios.
- **Ventajas:** Extremadamente rápido y eficiente, soporta expresiones regulares para búsquedas complejas, es una herramienta esencial en la caja de herramientas de cualquier usuario de Linux/Unix.

- **Uso común:** Filtrado de logs, búsqueda de código, análisis de datos textuales, automatización en scripts de shell.

6. Gobuster

- **Categoría:** Herramienta de fuerza bruta para directorios/archivos y subdominios.
- **¿Qué es?** Gobuster es una herramienta de código abierto escrita en Go que se utiliza para "fuerza bruta" (adivinar sistemáticamente) directorios, nombres de archivos, subdominios, nombres de bucket de AWS S3 y nombres de host de Virtual Host en servidores web.
- **¿Para qué sirve?** Es una herramienta popular en pruebas de penetración y CTFs (Capture The Flag) para:
    - **Descubrir directorios y archivos ocultos:** Intentar acceder a rutas comunes (como /admin, /backup, /test) que no están enlazadas en el sitio web pero que existen.
    - **Enumerar subdominios:** Encontrar subdominios válidos para un dominio dado (ej. dev.example.com, api.example.com).
    - **Identificar buckets S3:** Buscar buckets de almacenamiento en la nube mal configurados o expuestos.
- **Ventajas:** Muy rápido debido a que está escrito en Go y a su naturaleza concurrente, versátil para diferentes tipos de fuerza bruta web.
- **Uso común:** Fase de reconocimiento en pruebas de penetración, búsqueda de información sensible o expuesta en servidores web.

7. WPScan WordPress

- **Categoría:** Escáner de seguridad para WordPress.
- **¿Qué es?** WPScan es un escáner de seguridad de caja negra (no requiere acceso a las credenciales) para sitios de WordPress. Es una herramienta de código abierto escrita en Ruby que se especializa en encontrar vulnerabilidades en instalaciones de WordPress.
- **¿Para qué sirve?** Sirve para auditar la seguridad de sitios web construidos con WordPress, buscando:
    - **Versión de WordPress:** Detectar la versión del CMS y si tiene vulnerabilidades conocidas.
    - **Temas (Themes):** Identificar temas instalados, sus versiones y posibles vulnerabilidades.
    - **Plugins:** Encontrar plugins instalados, sus versiones y sus vulnerabilidades conocidas.
    - **Usuarios:** Enumerar nombres de usuarios válidos (lo que puede ser un paso previo a un ataque de fuerza bruta de contraseñas).

    - **Archivos de configuración sensibles:** Buscar archivos que puedan exponer información confidencial.
- **Uso común:** Pruebas de penetración de sitios WordPress, auditorías de seguridad periódicas y por desarrolladores de WordPress para verificar la seguridad de sus sitios.

A través de este documento, se detallan los procedimientos, hallazgos y soluciones implementadas en cada fase para asegurar el servidor y fortalecer la postura de seguridad de la infraestructura.

FASE 1: Corrección de un hackeo.

En esta fase nos centramos en detectar, identificar y eliminar la vulnerabilidad presente en el servidor de 4 Geeks.

Revision de logs

Como primer paso realizamos mediante el comando “last” para conocer los últimos logs realizados en el servidor, así como utilizaremos el programa **Grep** para conocer el resto de logins.

```
debian@debian:~$ last
debian   tty7         :0               Sat Jul 12 05:00   still logged in
reboot   system boot  6.1.0-25-amd64   Fri Jul 11 16:56   still running
debian   tty7         :0               Tue Jul  1 04:22 - crash (10+12:34)
reboot   system boot  6.1.0-25-amd64   Tue Jul  1 04:20   still running
debian   tty7         :0               Tue Oct  8 17:28 - crash (265+10:51)
reboot   system boot  6.1.0-25-amd64   Tue Oct  8 17:28   still running
debian   tty7         :0               Tue Oct  8 16:48 - crash  (00:40)
reboot   system boot  6.1.0-25-amd64   Tue Oct  8 16:48   still running
debian   tty7         :0               Tue Oct  8 16:44 - crash  (00:03)
reboot   system boot  6.1.0-25-amd64   Tue Oct  8 16:43   still running
debian   tty7         :0               Mon Sep 30 15:13 - crash (8+01:29)
reboot   system boot  6.1.0-25-amd64   Mon Sep 30 15:09   still running
debian   tty7         :0               Mon Sep 30 09:49 - 12:27  (02:38)
reboot   system boot  6.1.0-23-amd64   Mon Sep 30 09:48 - 12:28  (02:39)
debian   tty7         :0               Sat Sep 28 16:40 - crash (1+17:08)
reboot   system boot  6.1.0-23-amd64   Sat Sep 28 16:39 - 12:28 (1+19:48)
debian   tty7         :0               Wed Jul 31 16:45 - 18:18  (01:33)
reboot   system boot  6.1.0-23-amd64   Wed Jul 31 16:45 - 18:19  (01:34)
debian   tty7         :0               Wed Jul 31 16:04 - 16:44  (00:39)
reboot   system boot  6.1.0-23-amd64   Wed Jul 31 16:04 - 16:44  (00:40)
debian   tty7         :0               Wed Jul 31 15:57 - 15:59  (00:01)
reboot   system boot  6.1.0-23-amd64   Wed Jul 31 15:56 - 15:59  (00:02)
```

En el resultado de estos escaneos se detectan sesiones de corta duración o duraciones de sesiones cortas o sin sentido, lo que puede suponer reinicios forzados, además de darse múltiples “crashes” que sugieren que el equipo se apagó de forma inesperada, lo cual puede ser provocado por un error de hardware o bien por una intervención externa no controlada, para comprobarlo realizamos revisamos los logs de inicio del sistema mediante **journalctl** y comprobamos que todo mantiene una apariencia normal.

```
Jul 31 15:57:30 debian systemd[768]: Queued start job for default target default.target.
Jul 31 15:57:30 debian systemd[768]: Created slice app.slice - User Application Slice.
Jul 31 15:57:30 debian systemd[768]: Created slice session.slice - User Core Session Slice.
Jul 31 15:57:30 debian systemd[768]: Reached target paths.target - Paths.
Jul 31 15:57:30 debian systemd[768]: Reached target timers.target - Timers.
Jul 31 15:57:30 debian systemd[768]: Starting dbus.socket - D-Bus User Message Bus Socket...
Jul 31 15:57:30 debian systemd[768]: Listening on dirmngr.socket - GnuPG network certificate management daemon.
Jul 31 15:57:30 debian systemd[768]: Listening on gcr-ssh-agent.socket - GCR ssh-agent wrapper.
Jul 31 15:57:30 debian systemd[768]: Listening on gnome-keyring-daemon.socket - GNOME Keyring daemon.
Jul 31 15:57:30 debian systemd[768]: Queued start job for default target default.target.
Jul 31 15:57:30 debian systemd[768]: Created slice app.slice - User Application Slice.
Jul 31 15:57:30 debian systemd[768]: Created slice session.slice - User Core Session Slice.
Jul 31 15:57:30 debian systemd[768]: Reached target paths.target - Paths.
Jul 31 15:57:30 debian systemd[768]: Reached target timers.target - Timers.
Jul 31 15:57:30 debian systemd[768]: Starting dbus.socket - D-Bus User Message Bus Socket...
Jul 31 15:57:30 debian systemd[768]: Listening on dirmngr.socket - GnuPG network certificate management daemon.
Jul 31 15:57:30 debian systemd[768]: Listening on gcr-ssh-agent.socket - GCR ssh-agent wrapper.
Jul 31 15:57:30 debian systemd[768]: Listening on gnome-keyring-daemon.socket - GNOME Keyring daemon.
Jul 31 15:57:30 debian systemd[768]: Listening on gpg-agent-browser.socket - GnuPG cryptographic agent and passphrase cache (
Jul 31 15:57:30 debian systemd[768]: Listening on gpg-agent-extra.socket - GnuPG cryptographic agent and passphrase cache (re
Jul 31 15:57:30 debian systemd[768]: Listening on gpg-agent-ssh.socket - GnuPG cryptographic agent (ssh-agent emulation).
Jul 31 15:57:30 debian systemd[768]: Listening on gpg-agent.socket - GnuPG cryptographic agent and passphrase cache.
Jul 31 15:57:30 debian systemd[768]: Listening on pulseaudio.socket - Sound System.
Jul 31 15:57:30 debian systemd[768]: Listening on dbus.socket - D-Bus User Message Bus Socket.
Jul 31 15:57:30 debian systemd[768]: Reached target sockets.target - Sockets.
Jul 31 15:57:30 debian systemd[768]: Reached target basic.target - Basic System.
Jul 31 15:57:30 debian systemd[768]: Starting pulseaudio.service - Sound Service...
```

Usuarios en el sistema

A continuación, se relatan los usuarios existentes en el sistema usando el programa **cat**; comprobando que no existen usuarios sospechosos.

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
systemd-timesync:x:997:997:systemd Time Synchronization:/:/usr/sbin/nologin
messagebus:x:100:107::/nonexistent:/usr/sbin/nologin
avahi-autoipd:x:101:110:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:102:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
dnsmasq:x:103:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
avahi:x:104:112:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nologin
speech-dispatcher:x:105:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
pulse:x:106:114:PulseAudio daemon,,,:/run/pulse:/usr/sbin/nologin
saned:x:107:117::/var/lib/saned:/usr/sbin/nologin
```

Una vez que podemos descartar la evidencia de logins sospechosos, usuarios malignos o conexiones mediante SSH, realizamos un escaneo del servidor mediante el programa **Nmap;**

Escaneo de puertos:

```
┌──(kali㉿kali)-[~]
└─$ nmap -sS -O -p- 10.0.2.9
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-12 05:44 EDT
Nmap scan report for 10.0.2.9
Host is up (0.00049s latency).
Not shown: 65532 closed tcp ports (reset)
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
80/tcp open  http
MAC Address: 08:00:27:04:E9:77 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.19, OpenWrt 21.02 (Linux 5.4)
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.13 seconds
```

El resultado del escaneo nos da 3 puertos abiertos;

- Puerto 21: FTP (File Transfer Protocol), puede ser vulnerable si permite el acceso anónimo sin contraseña. Lo analizaremos más tarde.
- Puerto 22: SSH (Secure Shell), analizado previamente mediante el escaneo de logs, donde hemos comprobado que no existe incidencia.
- Puerto 80: Constituye un servidor web, al abrirlo en el navegador comprobamos que se trata de Apache2.

Puerto 21:

En la comprobación del puerto 21 vemos que efectivamente se permite el acceso anónimo a la maquina sin necesidad de contraseña mediante las credenciales anonymous/” sin contraseña”, lo que constituye una gran vulnerabilidad.

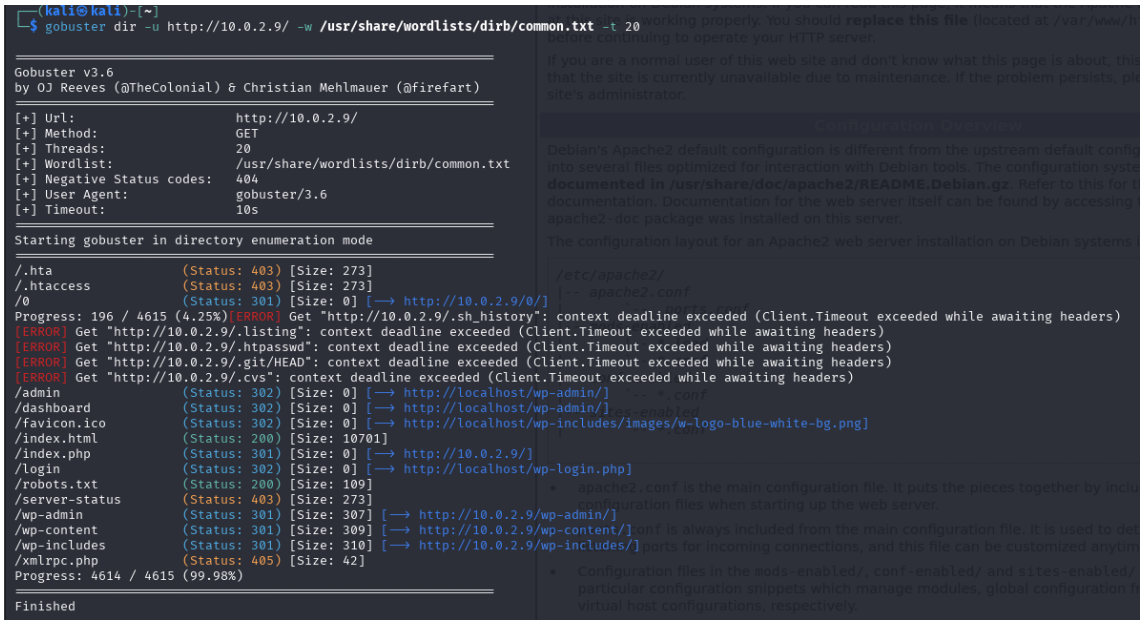
```
┌──(kali㉿kali)-[~]
└─$ ftp 10.0.2.9
Connected to 10.0.2.9.
220 (vsFTPd 3.0.3)
Name (10.0.2.9:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
```

Puerto 80:

En el puerto 80 analizamos en primer lugar si efectivamente está escuchando:

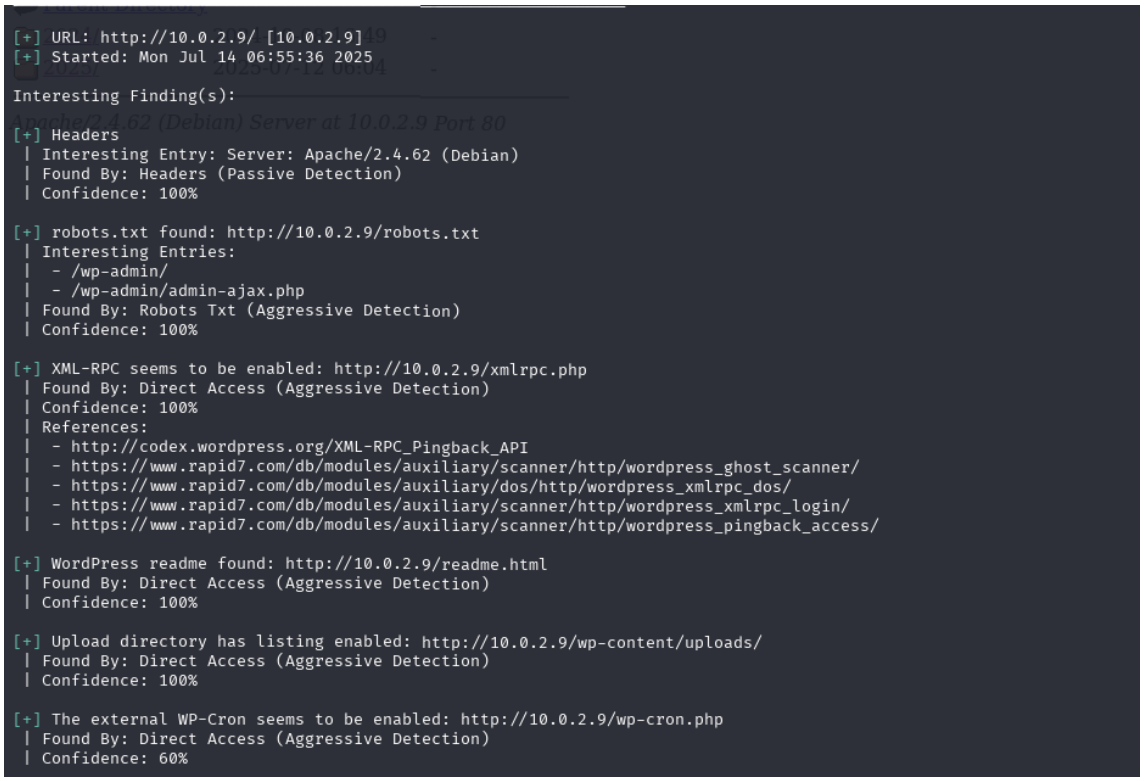
```
debian@debian:~$ sudo netstat -tulnp | grep :80
tcp6       0      0 :::80                   :::*                    LISTEN      2148/apache2
```

Una vez confirmado, analizamos mediante **gobuster** el puerto y nos detalla que se trata de un wordpress, además de darnos información acerca de varias posibles vulnerabilidades;

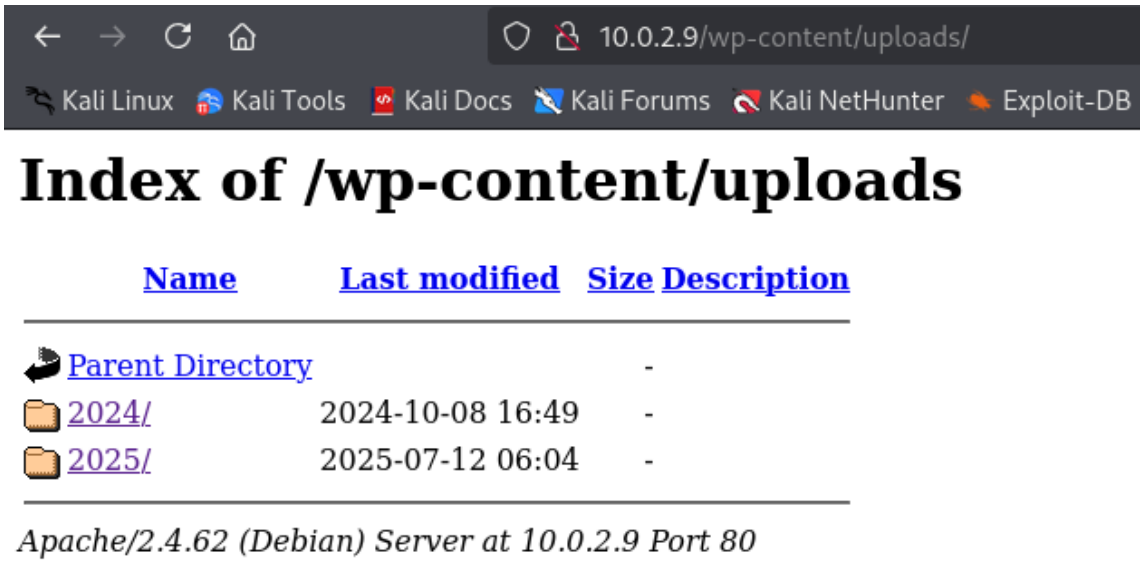


A partir de este resultado de gobuster realizamos un scan más profundo con el programa **Wpscan**, que detalla diferentes vulnerabilidades, desde posibles ataques de fuerza bruta hasta posibles filtraciones de datos sensibles presentes en “/wp-contnet/upladds/”. A continuación, comprobamos la vulnerabilidad de fuerza bruta y las demás vulnerabilidades comprobadas:

Wpscan:



Filtración de archivos: (es accesible pero no hay información relevante)



Revision de malware

Una vez analizado los logs, usuarios y puertos de la maquina pasamos a examinar los posibles malwares presentes en el servidor. Para ello realizaremos un escaneo exhaustivo con los programas **rkhunter** y **chrootkit**.

```
  Checking system startup files for malware               [ None found ]

Performing group and account checks
  Checking for passwd file                                [ Found ]
  Checking for root equivalent (UID 0) accounts           [ None found ]
  Checking for passwordless accounts                      [ None found ]
  Checking for passwd file changes                        [ None found ]
  Checking for group file changes                         [ None found ]
  Checking root account shell history files               [ OK ]

Performing system configuration file checks
  Checking for an SSH configuration file                  [ Found ]
  Checking if SSH root access is allowed                  [ Warning ]
  Checking if SSH protocol v1 is allowed                  [ Not set ]
  Checking for other suspicious configuration settings    [ None found ]
  Checking for a running system logging daemon            [ Found ]
  Checking for a system logging configuration file        [ Found ]

Performing filesystem checks
  Checking /dev for suspicious file types                 [ None found ]
  Checking for hidden files and directories               [ None found ]
```

```
Checking `write'...                                        not infected
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
```

Los resultados muestran 3 posibles vulnerabilidades, siendo estas:

- Usr/bin/lwp-request
- Checking for spucious (large) shared memory segments
- Checking if SSH root access is allowed

De estas 3 posibles fallas de seguridad la única relevante es la referente al SSH, ya identificada anteriormente.

MITIGACION DE VULNERABILIDADES

En primer lugar, eliminaremos la posibilidad de logearnos de forma anónima desde el puerto 21 editando el archivo mediante el comando “sudo nano /etc/vsftpd.conf” para así poder añadir el siguiente comando:

```
  GNU nano 7.2                                    /etc/vsftpd.comf *
anonymous_enable=NO
```

Tras editar la línea reiniciamos el servicio.

Acontinuacion desactivaremos el inicio de sesión SSH mediante root editando el archivo con el comando “sudo nano /etc/ssh/sshd_config” y poder cambiar el siguiente comando de “YES” a “NO”:

```
#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes
```

Al igual que en el caso anterior reiniciamos el servicio con el comando “sudo systemctl restart ssh”

Además de eliminar estas vulnerabilidades descubiertas, se debe fortalecer también las configuraciones poco seguras:

- Cambio de contraseñas de todos los usuarios privilegiados:
  Se actualizaron las contraseñas de los usuarios administradores por contraseñas fuertes con longitud superior a 12 caracteres y combinación de mayúsculas, minúsculas, números y símbolos.
- Desactivación de puertos y servicios innecesarios:
  Se revisaron los puertos abiertos y se desactivaron aquellos servicios que no son esenciales para el funcionamiento del servidor, reduciendo la superficie de ataque.

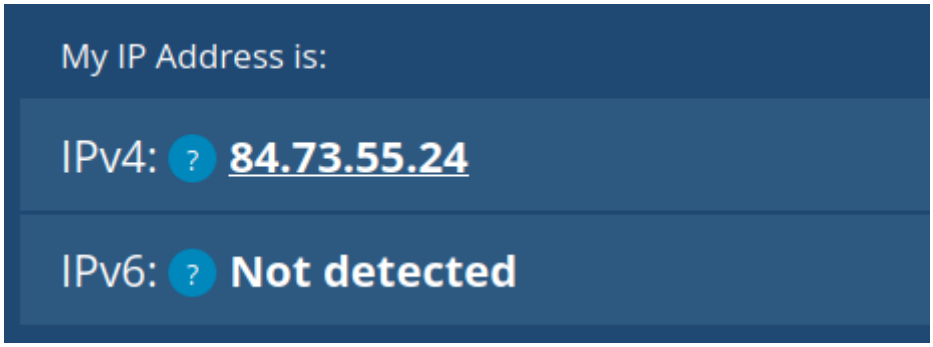
En cuanto al puerto 80 editaremos la configuración para evitar un posible ataque de fierza bruta a través de la web, eso lo hacemos de la siguiente manera a través de denegar a tarves de xmlrpc.php:

```
 GNU nano 7.2                                    /var/www/html/.htaccess *

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

A continuación, realizamos la comprobación de que se ha realizado correctamente, obteniendo el resultado 403 Forbiden:

```
debian@debian:~$ curl -i http://10.0.2.9/xmlrpc.php
HTTP/1.1 403 Forbidden
Date: Mon, 14 Jul 2025 11:29:56 GMT
Server: Apache/2.4.62 (Debian)
Content-Length: 273
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
<hr>
<address>Apache/2.4.62 (Debian) Server at 10.0.2.9 Port 80</address>
</body></html>
```

Por último, y como método de protección más importante para la protección del Wordpress, restringimos el acceso a /wpadmin para Esto sirve para proteger el panel de administración de WordPress y evitar; intentos de fuerza bruta de loginEç, escaneo de ficheros administrativos o el acceso no autorizado. Para ello en primer lugar descubrimos cual es la IP de internet del servidor:



Una vez tenemos esta infromacion, editamos la configuración, permtiendo qwue únicamente se pueda acceder a /wpadmin desde esta IPy no de froma remota, lo que asegura que se hace desde los servidores de 4 Geeks.

```
  GNU nano 7.2
Order Deny, Allow
Deny from all
Allow from 84.73.55.24
```

Si alguien externo a la empresa, servidor o desde otra IP intenta conectarse verá un 403 Forbidden, habiendo solventado así las vulnerabilidades de los puertos 21(File Transfer Protocol), 22 (SSH) y 80 (Wordpress).

Además, será preciso instalar un firewall, **ufw** en este caso, para proteger el sistema controlando que tráfico de red está permitido, tanto para la entrada de información como de salida.

Este firewall en cuestión es sencillo de configurar, habilitar o deshabilitar en cada momento y consultar sus parámetros.

```
debian@debian:~$ sudo apt install ufw
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libdaxctl1 libndctl6 libpmem1 linux-image-6.1.0-22-amd64
  linux-image-6.1.0-23-amd64
```

FASE 2 – EXPLOTACION DE UNA VULNERABILIDAD

Informe de Pentesting – Vulnerabilidad regreSSHion (CVE-2024-6387) en el servidor SSH de 4Geeks

Introducción

Durante un proceso de auditoría de seguridad en la infraestructura de la academia 4Geeks, se realizó un test de penetración dirigido a comprobar si existían vulnerabilidades críticas que pudieran ser aprovechadas por atacantes externos. Este análisis se centró principalmente en los servicios expuestos a Internet, ya que representan los puntos de entrada más frecuentes en incidentes de seguridad.

En el transcurso de este estudio se detectó que uno de los servidores Debian que utilizan en 4Geeks tiene instalada una versión de OpenSSH afectada por la vulnerabilidad conocida como regreSSHion, catalogada como CVE-2024-6387. Esta vulnerabilidad permite que una persona externa pueda ejecutar comandos en el sistema con permisos de administrador (root), sin necesidad de tener usuario ni contraseña.

Este documento explica paso a paso cómo se identificó la vulnerabilidad, qué herramientas se emplearon, por qué es tan grave, de qué manera podría explotarse, cómo mitigarla y qué conclusiones se extraen.

Reconocimiento de los servicios expuestos

El primer paso del test consistió en identificar los servicios que el servidor de 4Geeks tiene abiertos hacia Internet. Para ello se utilizó **Nmap**, una herramienta que escanea puertos y detecta qué aplicaciones los están usando. El comando empleado fue:

nmap -sV -p- 10.0.2.9;

-sV indica que Nmap intente averiguar la versión de los servicios.

-p- ordena que se revisen todos los puertos TCP.

Tras la ejecución, **Nmap** mostró que el puerto 22/tcp estaba abierto y correspondía al servicio SSH. El resultado fue el siguiente:

PORT STATE SERVICE VERSION

22/tcp open ssh OpenSSH 9.2p1 Debian-2+b1 (protocol 2.0)

Esto confirmó que el servidor utilizaba **OpenSSH** versión 9.2p1, la cual Debian distribuye por defecto en las versiones recientes.

Para comprobarlo con más precisión, se usó la herramienta **Netcat**, que permite conectarse al puerto y leer el mensaje de bienvenida (banner). El comando fue:

nc 10.0.2.9 22

La respuesta proporcionada por el servidor es:

"SSH-2.0-OpenSSH_9.2p1 Debian-2+b1"

Esto validó que efectivamente se trataba de la versión vulnerable.

Herramientas y métodos utilizados

En este análisis se emplearon diferentes herramientas y técnicas, combinando reconocimiento activo y consulta de bases de datos de vulnerabilidades:

Nmap:"NetworkMapper, utilizada para descubrir los puertos abiertos y detectar versiones de los servicios.

Netcat: para extraer el banner de SSH.

Searchsploit: herramienta que revisa el repositorio Exploit-DB en busca de exploits públicos relacionados con la versión detectada. También se pueden consultar en webs públicas que recogen estas vulnerabilidades, como CVE.mitre. El comando utilizado fue:

searchsploit openssh 9.2

En los resultados se encontraron referencias a la vulnerabilidad **CVE-2024-6387.**

Vulnerabilidad: riesgo y funcionamiento

Gracias a este conjunto de comprobaciones se confirmó que el servidor estaba expuesto a un riesgo alto.

Esta vulnerabilidad afecta a **OpenSSH**, el programa que permite conectarse a un servidor de forma remota por terminal (por ejemplo, usando ssh usuario@servidor).

Lo peligroso es que esta vulnerabilidad permite que alguien sin usuario ni contraseña pueda tomar el control total del servidor, aprovechando una falla interna en la forma en que el servidor maneja muchas conexiones al mismo tiempo.

En la practica la vulnerabilidad se puede explotar de la siguiente manera: el servidor está preparado para que muchos usuarios se conecten por SSH al mismo tiempo, pero hay un error en cómo el programa maneja la memoria cuando eso ocurre. Un atacante puede abrir muchas coneziones a la vez (algo similar a un ataque de DDoS), de forma muy rápida y repetida. Si lo hace con el ritmo exacto, puede engañar al servidor para que ejecute código malicioso sin querer, siendo este código lo que le da al atacante una puerta de entrada como si fuera el administrador (root).

De manera más técnica, el proceso que se produce es el siguiente:

Cuando un usuario intenta conectar por SSH, el servidor inicia ciertos procesos y reserva memoria para gestionar esa conexión. Si un atacante establece muchas conexiones al mismo tiempo, puede provocar que los hilos de ejecución se solapen y accedan a las mismas zonas de memoria de forma desordenada.

En ese momento, el atacante puede aprovechar para inyectar datos que, si la sincronización es correcta, terminan ejecutándose como código. Dado que sshd se ejecuta con privilegios de root, el atacante obtiene control total sobre el servidor. Lo más preocupante es que este ataque no requiere usuario ni contraseña. Solo es necesario que el puerto 22/tcp esté expuesto a Internet.

Posibles consecuencias

Esta vulnerabilidad es especialmente crítica porque permite la ejecución remota de comandos como root, no requiere autenticación, la explotación puede hacerse de manera automatizada y puede provocar la pérdida completa del control del servidor, incluyendo:

- Robo o modificación de datos
- Instalación de malware.
- Creación de usuarios ocultos.
- Ataques a otros sistemas internos de 4Geeks.

Por todo esto, el nivel de riesgo es crítico y requiere acción inmediata.

Explotación de vulnerabilidad

El atacante crea un script (por lo que el atacante necesitaría tener cierto conocimiento sobre programación) que abre muchas conexiones SSH simultáneas al servidor de 4Geeks, haciendo que cada conexión provoque que **SSHD** reserve memoria y active un proceso.

El script manda señales cuidadosamente sincronizadas, buscando que dos procesos accedan a la misma memoria al mismo tiempo.

Tras cientos o miles de intentos (automatizados por el script), en algún momento el atacante logra alterar datos internos del programa.

Aprovechando esa corrupción de memoria, introduce instrucciones que abren una shell remota con privilegios de administrador.

Una vez lograda la shell, el atacante tiene el control completo del sistema.

Este procedimiento puede llevar minutos u horas, pero es factible si el puerto SSH está accesible, como es el caso de la Debian de 4Geeks.

Evidencias encontradas

En resumen, durante la auditoría se documentaron las siguientes evidencias que confirman la vulnerabilidad:

- Versión vulnerable detectada: OpenSSH 9.2p1 Debian-2+b1.
- Puerto SSH expuesto públicamente: sin restricciones de acceso.
- Banner SSH coincidente con la versión afectada.
- Referencias a CVE-2024-6387 en Exploit-DB.
- Ausencia de medidas adicionales de aislamiento en la configuración en la separación de privilegios (medida que limita el alcance del daño si una parte del sistema o programa es comprometido, en cierta manera relacionado con el “Zero Trust”)

Recomendaciones para mitigar el riesgo

Para proteger de inmediato el servidor Debian de 4Geeks, se recomienda aplicar estos pasos de mitigación:

- Actualizar OpenSSH a la versión corregida.

sudo apt update

sudo apt install --only-upgrade openssh-server

Luego reiniciar el servicio SSH:

sudo systemctl restart ssh

- Activar el aislamiento de privilegios.

Editar el archivo de configuración y añadir la línea “UsePrivilegeSeparation sandbox”:

sudo nano /etc/ssh/sshd_config

Y reiniciar SSH de nuevo.

- Restringir el acceso SSH por IP.

Si solo se conectan desde ciertas redes (por ejemplo, la sede de 4Geeks), la seguridad aumenta notablemente, pues solo un inside intrudrr sería capaz de atacar el sistema. Debe ser limitarlo con ufw de la siguiente manera:

sudo ufw allow from 10.0.2.9 to any port 22

sudo ufw deny 22

sudo ufw enable

- Monitorear los registros de seguridad.

Revisar regularmente el comando para detectar conexiones inusuales o sospechosas, tal y como hemos realizado en la Fase 1 del informe:

/var/log/auth.log

Conclusiones

La vulnerabilidad detectada representa un peligro muy alto para el servidor Debian de 4Geeks, ya que permite que cualquier persona externa pueda tomar el control completo sin autenticación. Si se explota con éxito, el atacante podría comprometer la información de estudiantes, profesores y recursos internos de la academia.

Por su criticidad, es imprescindible actualizar OpenSSH y aplicar todas las medidas de mitigación indicadas lo antes posible. Esta vulnerabilidad puede ser explotada de forma masiva, por lo que el tiempo de reacción es clave para prevenir incidentes graves.

Se recomienda que el área de sistemas de 4Geeks ejecute estas acciones de inmediato y planifique auditorías periódicas para comprobar que todos los servicios se mantienen actualizados y correctamente protegidos.

FASE 3: PLAN DE RESPUESTA

1. Introducción

Un Sistema de Gestión de Seguridad de la Información (SGSI) es un conjunto estructurado de políticas, procesos, controles y recursos destinados a proteger la información crítica de una organización. La norma **ISO/IEC 27001** define los requisitos para establecer, implementar, mantener y mejorar de manera continua un SGSI eficaz.

Este informe describe un modelo integral de SGSI orientado a la academia online de ciberseguridad 4Geeks, cuyas operaciones se desarrollan a través de plataformas de formación virtual, servicios en la nube y gestión de datos personales de estudiantes y colaboradores.

2. Objetivos del SGSI

- Garantizar la confidencialidad, integridad y disponibilidad de la información gestionada por la academia.
- Reducir los riesgos asociados a amenazas internas y externas.
- Cumplir con la normativa aplicable

3. Alcance del SGSI

El SGSI comprenderá:

- La plataforma de aprendizaje online (LMS).
- Sistemas de almacenamiento de datos de estudiantes, instructores y contenidos formativos.
- Procesos administrativos y de soporte (facturación, atención al cliente, recursos humanos).
- Entornos de desarrollo y pruebas de contenido digital.
- Comunicaciones internas y externas.

Quedan excluidos los equipos personales de los estudiantes fuera de la red corporativa.

4. Marco Normativo y Referencias

La implementación se fundamentará principalmente en:

- **ISO/IEC 27001:2022:** Norma internacional que define los requisitos del SGSI.
- **ISO/IEC 27002:2022:** Controles de seguridad de la información y directrices de aplicación.
- **ISO/IEC 27005:** Gestión de riesgos.
- Reglamento General de Protección de Datos (**RGPD**).

5. Análisis de Riesgos

Se identifican amenazas relevantes como:

- Acceso no autorizado a la plataforma educativa.
- Robo o pérdida de datos de estudiantes.
- Ataques de denegación de servicio (DDoS).
- Ingeniería social y phishing dirigido a empleados.

La evaluación de riesgos se realizará mediante metodologías como **ISO 27005** y registros en matrices que valoren impacto y probabilidad.

6. Definición de Políticas de Seguridad

Se elaborarán políticas que regulen:

- El uso aceptable de recursos tecnológicos.
- La gestión de contraseñas y autenticación.
- La clasificación y tratamiento de la información.
- El control de acceso físico y lógico.
- La gestión de incidentes de seguridad.
- La protección de datos personales.

Todas ellas deberán ser aprobadas por la dirección y comunicadas a los empleados.

7. Procedimientos y Controles de Seguridad

Se implementarán procedimientos que incluyan:

- Gestión de usuarios y control de acceso basado en roles.
- Revisión periódica de privilegios.

- Registro y monitoreo de eventos relevantes.
- Copias de seguridad automáticas y pruebas de recuperación.
- Plan de respuesta a incidentes.

Estos procedimientos se documentarán y revisarán regularmente.

8. Planes de Acción y Mejora Continua

El ciclo **PDCA** guiará el mantenimiento y mejora del SGSI:

- **Planificar:** Identificar objetivos, riesgos y controles.
- **Hacer:** Implementar los controles definidos.
- **Verificar:** Medir y auditar resultados.
- **Actuar:** Corregir desviaciones y optimizar procesos.

La mejora continua será un principio esencial de la gestión.

9. Roles y Responsabilidades

- **Dirección:** Aprobar políticas y asignar recursos.
- **Responsable de Seguridad:** Supervisar la implementación del SGSI.
- **Administradores de Sistemas:** Ejecutar controles técnicos y operativos.
- **Todo el Personal:** Cumplir con políticas y reportar incidentes.

La asignación de responsabilidades quedará reflejada en un documento de organización y funciones.

10. Proceso de Auditoría y Revisión

Se programarán auditorías internas anuales para:

- Verificar el cumplimiento de los requisitos de ISO 27001.
- Evaluar la eficacia de los controles.
- Detectar no conformidades y oportunidades de mejora.

Tras cada auditoría, la dirección revisará el SGSI y aprobará un plan de acciones correctivas.

11. Casos Prácticos: Medidas de Seguridad, Programas y Formación a Empleados

Para que el SGSI sea operativo y tangible, se aplicarán medidas prácticas de seguridad adaptadas a una academia online, con recursos razonables, pero con alto compromiso.

11.1. Medidas Técnicas Básicas

Firewalls y control de tráfico

- Configuración de un firewall perimetral (por ejemplo,UFW).
- Objetivo: bloquear ataques comunes (fuerza bruta, inyecciones).

Actualizaciones y parches

- Activación de actualizaciones automáticas en el LMS y los sistemas de gestión.
- Revisión mensual de parches pendientes.

Copias de seguridad automáticas

- Backups diarios en almacenamiento cifrado en la nube.
- Verificación trimestral de restauración.

Autenticación robusta

- Contraseñas complejas obligatorias.

Monitorización básica

- Instalación de **Wazuh** como sistema de registro y alerta.
- Monitoreo de accesos sospechosos y cambios de configuración.

Protección contra malware

- Escaneo semanal del servidor con **ClamAV** o solución equivalente.
- Revisión de archivos subidos por usuarios.

11.2. Formación y Concienciación

Curso inicial de seguridad

- Contenido: contraseñas seguras, identificación de phishing, uso de 2FA.
- Duración: 1–2 horas online.

Boletines trimestrales

- Noticias recientes de amenazas.

- Recordatorios de buenas prácticas.

Simulacro de incidente

- Ejemplo: simulación de filtración de credenciales.
- Objetivo: practicar el protocolo de respuesta.

Checklist mensual de seguridad

- Revisión de actualizaciones, copias de seguridad y alertas.

12. Conclusiones

La implementación de este SGSI, siguiendo la norma ISO 27001 y adaptado a la realidad de una academia online de ciberseguridad, permitirá:

- Reducir de manera significativa los riesgos.
- Asegurar la confianza de clientes y estudiantes.
- Cumplir con las obligaciones legales y contractuales.
- Mantener la disponibilidad y calidad del servicio educativo.

El compromiso sostenido de la dirección y la implicación activa de todo el personal serán determinantes para el éxito del sistema.