CTF

TAJKEYA SULTANA

08/30/2023

—

Cybersecurity Capture the Flag

Level 1

—

Masterschool

TABLE OF CONTENTS

Introduction

Synopsis

Masterschool strives to challenge their students to think out of the box. This CTF challenge was created to test our ability to solve problems, test our technical skills, file management, navigation, prove the skills we acquired during Linux fundamentals and many more that would translate to the field of cybersecurity.

The assignment was straightforward, find the hidden flags using Linux commands or utilizing online resource. Being detailed and thinking outside the box was highly encouraged.

Challenges

There were six challenges:

Linux Basics: demonstrate proficiency with basic Linux commands.

File Flag System: navigate and discover hidden flags through the file system.

Webpage Flags: navigate and discover hidden flags in webpages.

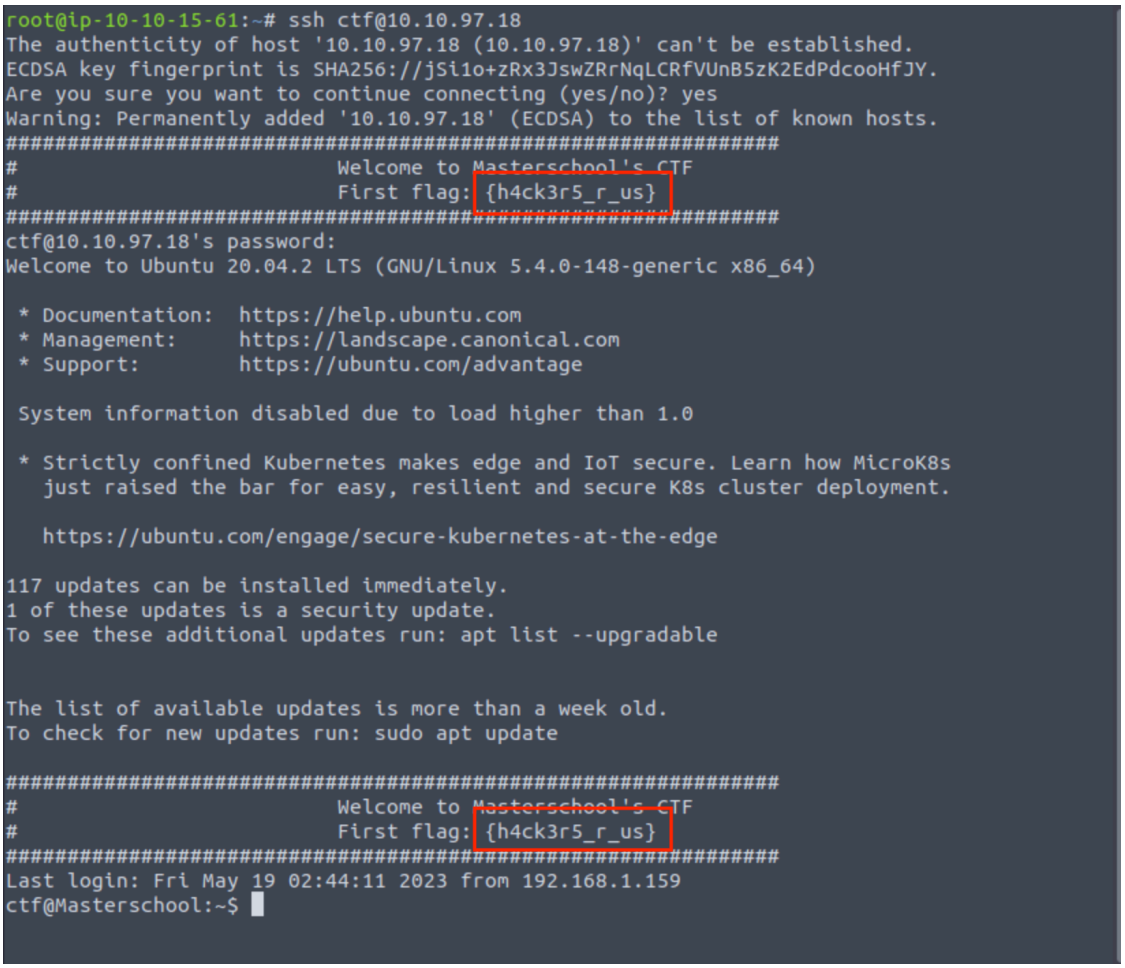
Hidden Flags: objectives hidden within a local system, webpage, or application, to be discovered utilizing learned skills.

Hash cracking: use appropriate tool to decrypt the hidden flags.

Nmap Scan Report: Run Nmap on the target to explore and discover open ports that can be exploited.

Lets' Get Started

The first step is to access the machine. After your machine has loaded and you can see root@IP, start entering "ssh ctf@your virtual machine IP address. It should look like the image below.



After we are in the system, we get our first flag.

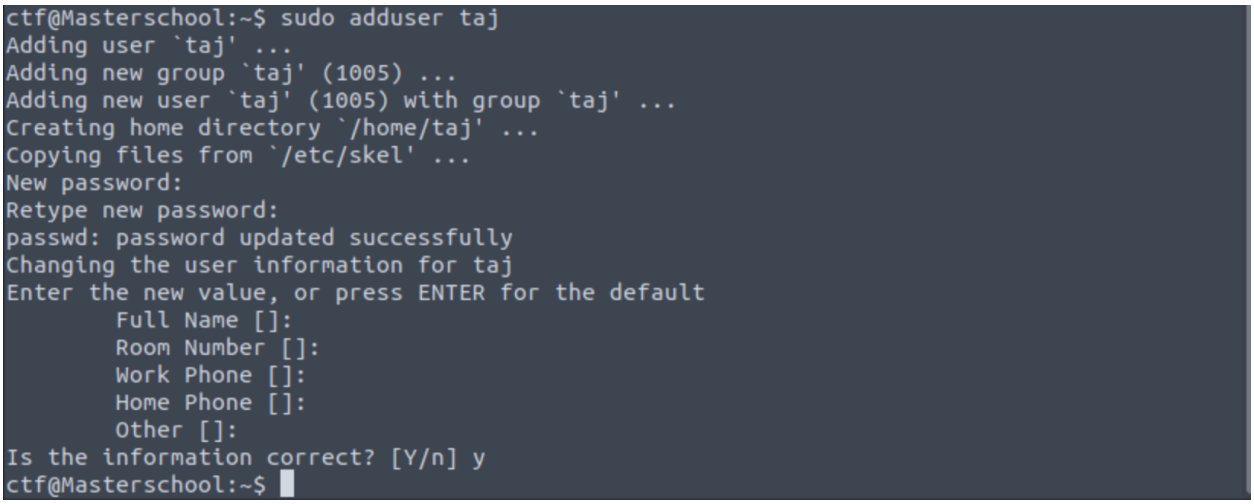
| {h4ck3r5_r_us}

Now that we are in the system, we can start our challenge.

Linux Basics: User and File Management

Our first challenge was to create a new user in the Linux system. To achieve this, we will be using “sudo” command. The sudo command allows us to run as an administrator, allowing us to make changes to users, create and delete users. Let just say it gives us premium privileges.

After we create our user, we have to assign it a password. For the next steps, press enter until you get the “Is this information correct? [Y/n]” type “y”. By pressing entering you are selecting the default option. For this process, we don’t have to enter all these information as shown in the image below.



Next step is to use “su” to start user change process. “su” Allows us to switch between users. You will need to enter your password to access. Type in cd to switch user.

See image below.

```
ctf@Masterschool:~$ su taj
Password:
taj@Masterschool:/home/ctf$ cd
taj@Masterschool:~$
```

Figure 3: Changing users

Now that we are in the user we want to be, its time to create folder and file. To create a folder, we will be using the “mkdir” command, which allows us create directories, a similar name is folder. This command can create multiple directories at once. It also allows the user to set permissions for the directories (folder).

The folder I created was “HiTaj”. I want to write a simple message inside the folder, let’s say “Hello World from Tajkeya”.

To create a message inside the directory, we need to use the “echo” command. It allows users to display lines of text or strings.

```
taj@Masterschool:~$ mkdir HiTaj
taj@Masterschool:~$ ls
HiTaj
taj@Masterschool:~$ cd HiTaj
taj@Masterschool:~/HiTaj$ echo 'Hello World from Tajkeya'
Hello World from Tajkeya
taj@Masterschool:~/HiTaj$ echo 'Hello World From Tajkeya' > HiTaj.txt
taj@Masterschool:~/HiTaj$ ls
HiTaj.txt
taj@Masterschool:~/HiTaj$ su ctf
Password:
ctf@Masterschool:/home/taj/HiTaj$ ls
HiTaj.txt
ctf@Masterschool:/home/taj/HiTaj$ cd
ctf@Masterschool:~$
```

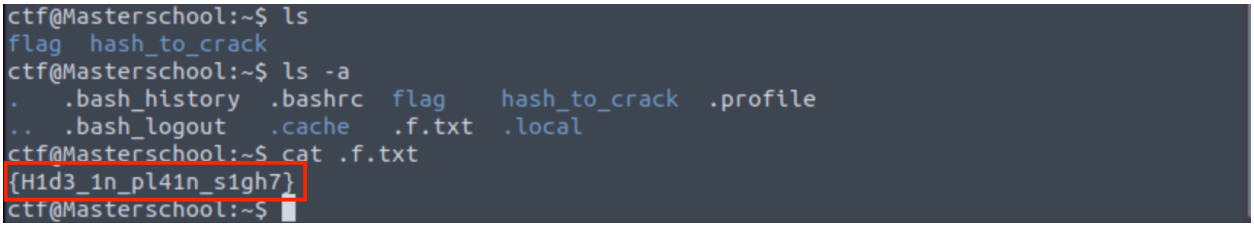
We created the text we want, now let’s save it using “>” and “HiTaj.txt” and change back to our default user which is CTF for this practice.

First challenge done!!

File System Flag

This challenge had us navigating the file system to find the flags. The command “ls” came in handy as it gave us hints on where to start looking.

The first thing I did was type ls. The ls command lists all the files and directories. Typing in ls gave me “flag hash_to_crack”, so I knew this was one of the hidden flags. I than used “-a”. the command “ls -a” will list all hidden files beginning with a dot.
As you can see in the figure below.



I used the cat command to display the flag directory. Which gave us our second hidden flag.

| {H1d3_1n_pl41n_s1gh7}

To find our next flag, we need to look for any files that end in “.txt”. We will be using the wildcard globing which represented by a asterisk, “*”. This lists all the file types in the current directories. Therefore, using “***.txt**” will display all the current directories that end in **.txt**.

The command we will execute is “*find . -name “*.txt*”. This command gave us another flag directory which is boxed in white.

```
ctf@Masterschool:~$ find . -name "*.txt"
./hash_to_crack/hash1.txt
./hash_to_crack/hash2.txt
./hash_to_crack/hash3.txt
./hash_to_crack/hash4.txt
./hash_to_crack/hash5.txt
./hash_to_crack/wordlist.txt
./flag/story.txt
./flag/6/m/a/s/t/e/r/s/c/h/o/o/l/f_l_a_g.txt
./.f.txt
```

Figure 6: includes 2 flag directory commands.

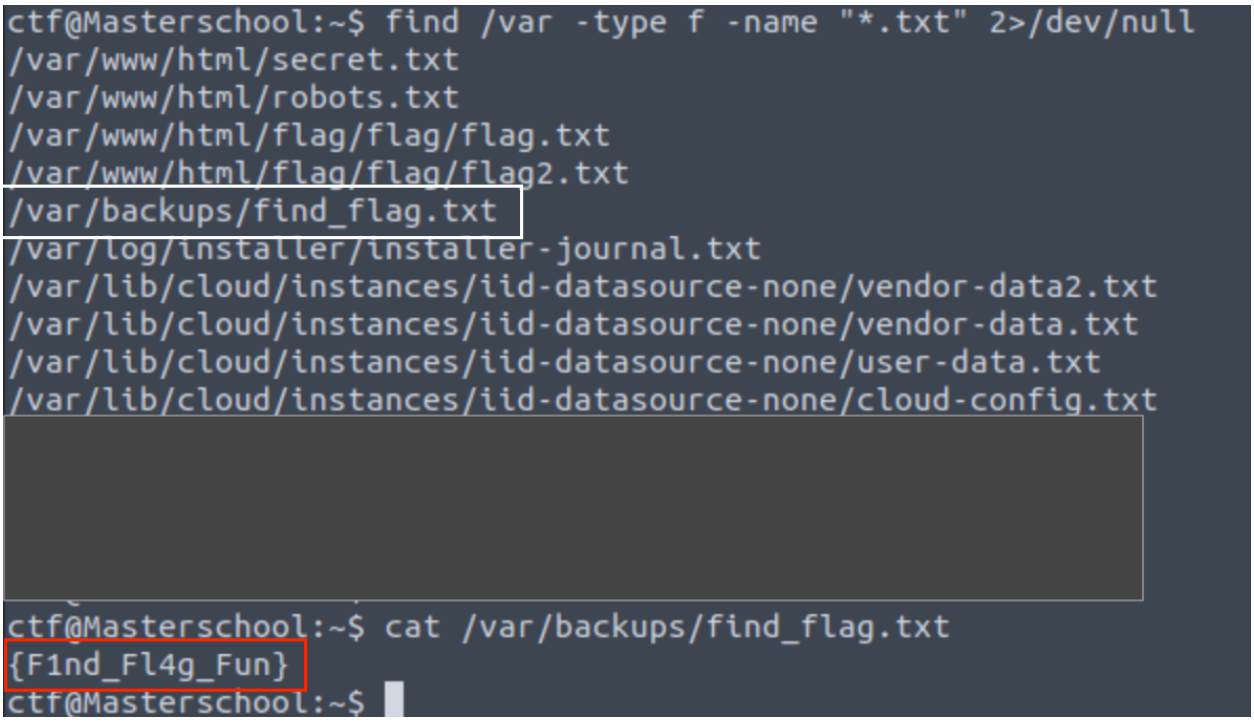
```
ctf@Masterschool:~$ cat flag/story.txt
Ch4pt3r 1: Cyb3rs3curity 3xp3rts S4v3 th3 D4y

1n 4 w0rld wh3r3 d1g1t4l thr34ts l00m l4rg3, 1t's 0ur cyb3rs3curity h3r03s wh0 st4nd t4ll 4nd d3f3nd th3 f0rt. Th3s3 4
r3 n0t y0ur typ1c4l sup3rh3r03s, but th3y p0ss3ss sk1lls f4r b3y0nd th3 0rd1n4ry, r34dy t0 w4rd 0ff 4ny m4l1c10us 1nv4
s10n. 💻🛡️

0n3 d4y, 4n 3m3rg3ncy s1tu4t10n 4r0s3. 4 l4rg3 c0rp0r4t10n f4c3d 4 s3v3r3 n3tw0rk br34ch, thr34t3n1ng th3 s3cur1ty 0f
th31r cl13nts' d4t4. Th3 s1tu4t10n w4s d1r3. ⚠️

4 t34m 0f cyb3rs3curity 3xp3rts w3r3 summ0n3d t0 s4v3 th3 d4y. Th3y spr4ng 1nt0 4ct10n, m4n0uv3r1ng thr0ugh th3 n3tw0r
k w1th pr3c1s10n, tr4c1ng th3 1ntrus10n's 0r1g1n. 🧬

Th3 3xp3rts w3r3 w3ll-3qu1pp3d w1th th3 l4t3st t00ls, 4nd th31r sk1lls w3r3 und3n14bl3. Th3y d3pl0y3d th31r f1r3w4lls,
 1mpl3m3nt3d 1ntrus10n d3t3ct10n syst3ms, 4nd 3ng4g3d th31r 3ncrypt10n m3th0ds. Th3 n3tw0rk w4s s3cur3d, 4nd th3 1ntru
d3rs w3r3 1s0l4t3d. 💻🔐

Th3 n3xt st3p w4s tr4ck1ng d0wn th3 p3r3tr4t0rs. Us1ng th31r f0r3ns1cs sk1lls, th3y m4n4g3d t0 p1n d0wn th3 1p 4ddr3ss
3s 0f th3 4tt4ck3rs. Th3 1ntrud3rs h4d n0 1d34 th4t th31r d4ys w3r3 numb3r3d. 🕵️‍♀️

1n th3 4ft3rm4th 0f th3 br34ch, 0ur cyb3rs3curity h3r03s d1dn't r3st 0n th31r l4ur3ls. 1t w4s t1m3 f0r th3 c0unt3r-4tt
4ck. 💪

Th3y tr4ck3d th3 1ntrud3rs' 1P 4ddr3ss3s, p1nn1ng th31r l0c4t10n. W1th pr3c1s10n 4nd d3t3rm1n4t10n, th3y s3t 0ut t0 br
1ng th3 p3r3tr4t0rs t0 just1c3. 🎯

Th3 3xp3rts 4ssembl3d 4 t34m 0f d1g1t4l f0r3ns1cs 4n4lysts, n3tw0rk 3ng1n33rs, 4nd l3g4l 4uth0r1t13s. T0g3th3r, th3y l
4unch3d 4 c0mpreh3ns1v3 1nv3st1g4t10n. 🔍

Us1ng th3 3v1d3nc3 th3y h4d g4th3r3d, th3y w3r3 {St0ry_Fl4g} 4bl3 t0 1d3nt1fy th3 1ntrud3rs 4nd th31r m0dus 0p3r4nd1.
Th3y r3v34l3d th3 1ntrud3rs' 1nt3nt10ns, th31r m3th0ds, 4nd, m0st 1mp0rt4ntly, th31r 1d3nt1t13s. 🧐

W1th th3s3 1nf0rmat10n 1n h4nd, th3y w3r3 4bl3 t0 4l3rt l4w 3nf0rc3m3nt 4g3nc13s. Th3 1ntrud3rs w3r3 s00n c0rn3r3d, th
31r pl4ns f01l3d, 4nd th31r 4tt4ck n3utr4l1z3d. 💼

But th3 cyb3rs3curity h3r03s' j0b w4s n0t y3t d0n3. Th3y w0rk3d t1r3l3ssly t0 r3p41r th3 d4m4g3, str3ngth3n1ng th3 c0r
p0r4t10n's n3tw0rk, 1nst4ll1ng str0ng3r d3f3ns3s, 4nd 3nsur1ng s1m1l4r 4tt4cks w0uld b3 pr3v3nt3d 1n th3 futur3. 🛡️

1n th3 3nd, 1t w4s
ctf@Masterschool:~$
```

Our third flag was hidden in the story. I must say, I have never read a story like this before. It was quite enjoyable and a security story.

| {Story_Fl4g}

To find our next flag, he /var command which contains our variable data files came in handy. I had to seek aid from ChatGPT to find the correct command. We will still be using the "*.txt", but adding a few more syntax.
After a few attempts, this is the command I decided to execute. "*find /var -type f -name "*.txt" 2>/dev/null*". The "*2>/dev/null*" gets rid of the error messages providing us with only the necessary information we need.

After executing the command, we found our next flag directory which unveiled our fourth flag.



Fourth flag found.
| {F1nd_Fl4g_Fun}

For the last file flag, I had to cd to flag to unveil the hidden flag. If you recall figure 6, this flag directory was also there along with the ./flag/story.txt. This is another way of finding the flag directory.

```
ctf@Masterschool:~$ cd flag
ctf@Masterschool:~/flag$ find -type f
./story.txt
./6/m/a/s/t/e/r/s/c/h/o/o/l/f_l_a_g.txt
ctf@Masterschool:~/flag$ cat ./6/m/a/s/t/e/r/s/c/h/o/o/l/f_l_a_g.txt
(Y0u_G0T_1t}
ctf@Masterschool:~/flag$
```

Fifth flag found.

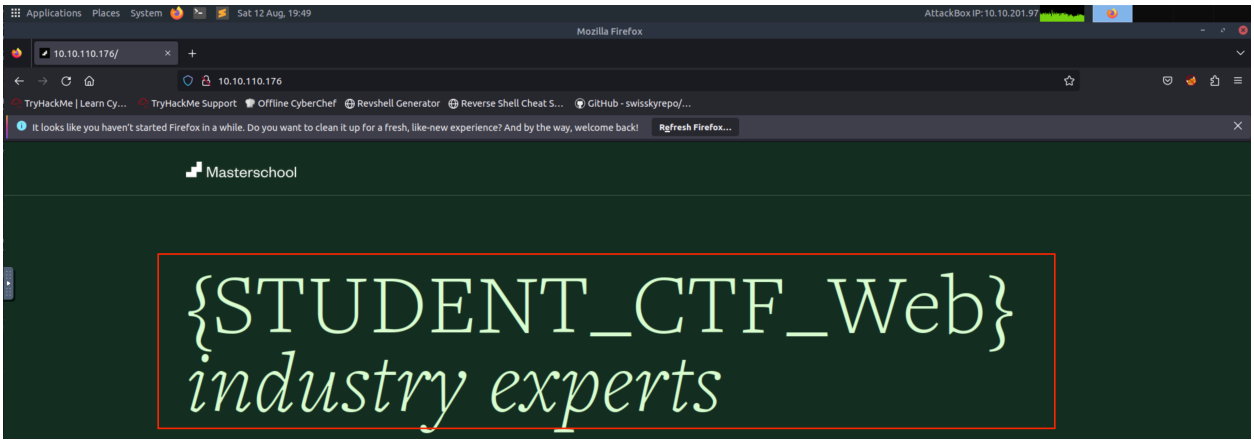
| {You_Got_1t}

Second challenge done!!

Webpage Flags & Hidden Flags

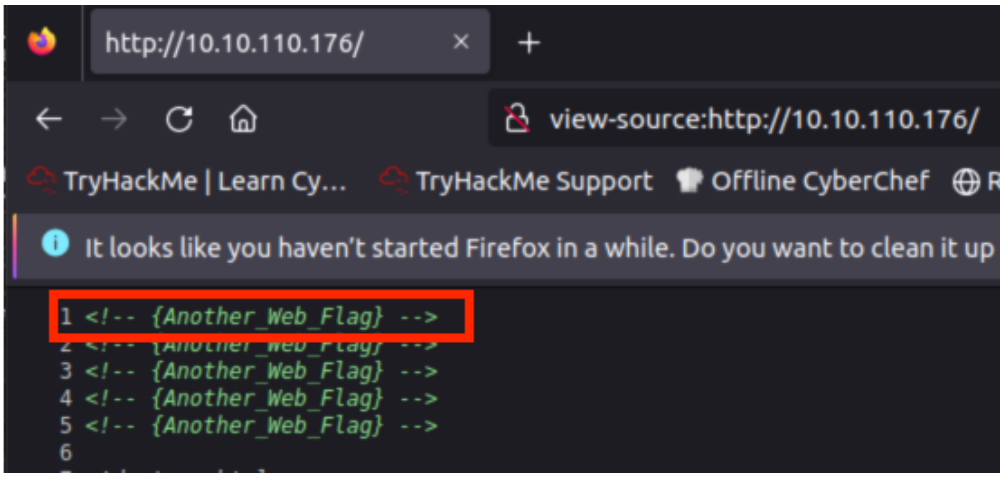
The webpage flags could have been found using Linux, but I wanted to try a different way. Let's just say thinking outside the box. I utilized the web to find the next few flags.

The first thing I did was open the web and put in the machine IP address. Which displayed the sixth flag.



| {STUDENT_CTF_WEB}

For the next flag, I utilized http protocol and entered the machine IP address which took me to this website.



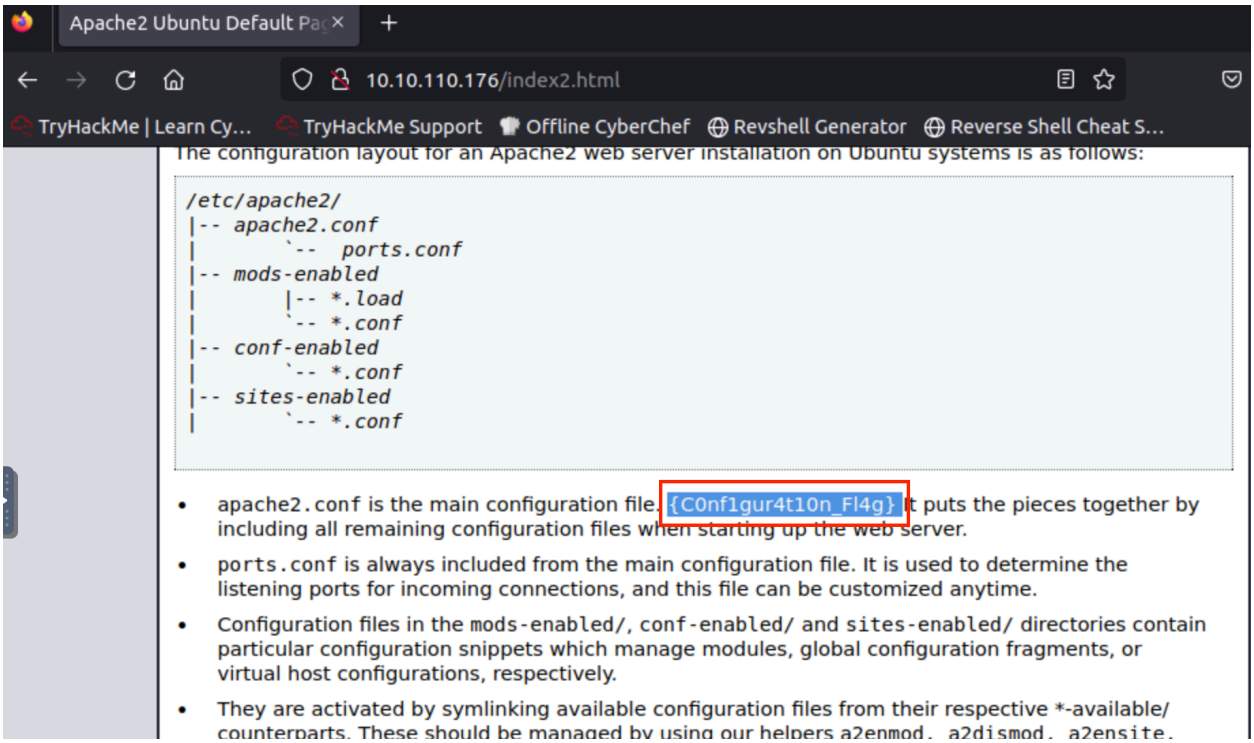
The first thing shown was the seventh flag.

| {Another_Web_Flag}

To find the next few webpage flags, I needed to utilize the Linux machine. I executed a simple command to switch user, "*cd /var/www/html*", and typed in ls to list all the webpage files.

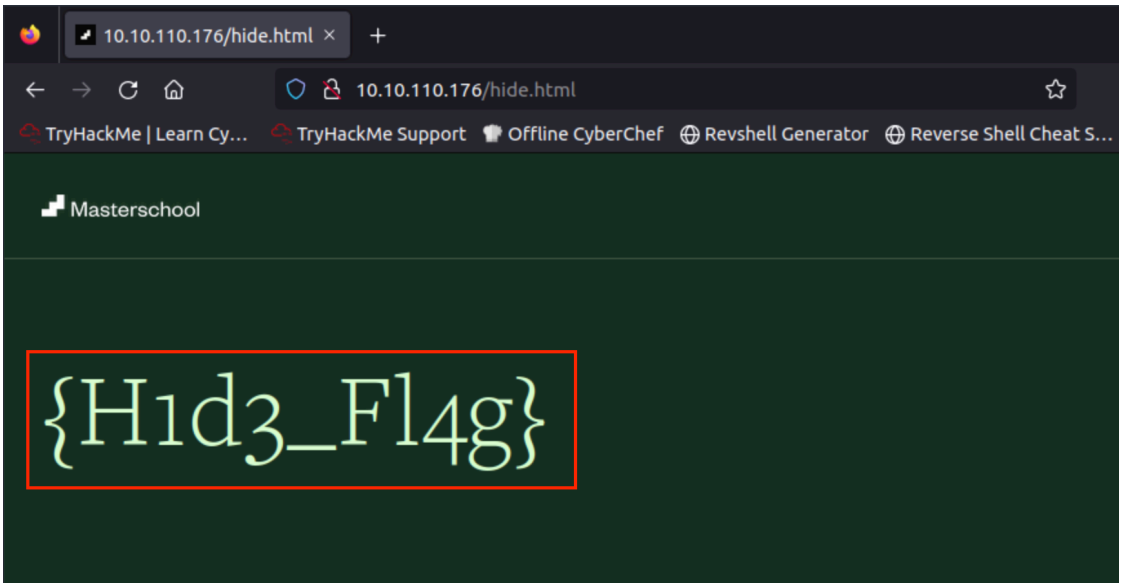
```
ctf@Masterschool:~$ cd /var/www/html
ctf@Masterschool:/var/www/html$ ls
flag  hide.html  index2.html  index.html  robots.txt  secret.txt
ctf@Masterschool:/var/www/html$
```

The next steps were self-explanatory, entering these hints with the IP address.



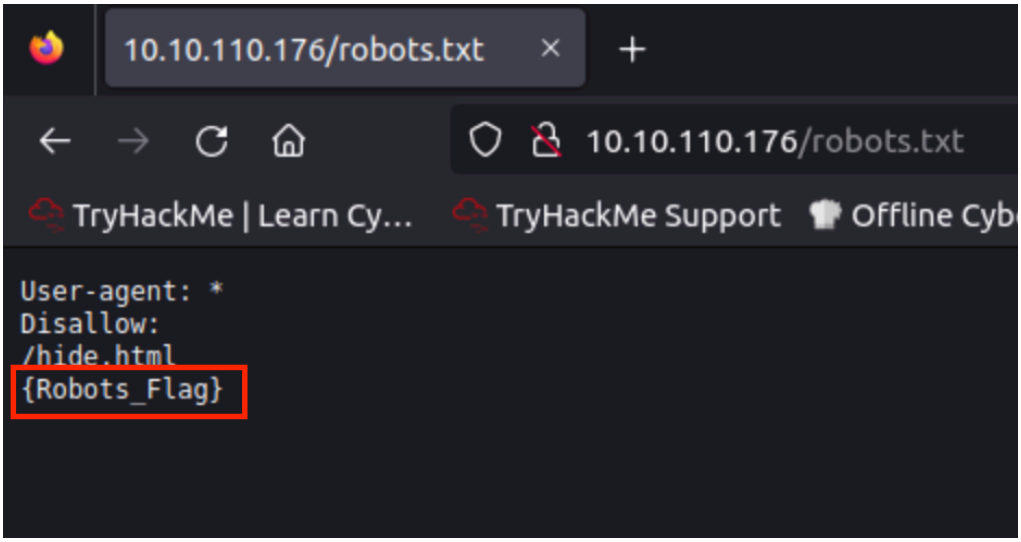
Eighth flag found.

| {C0nf1gur4t10n_Fl4g}



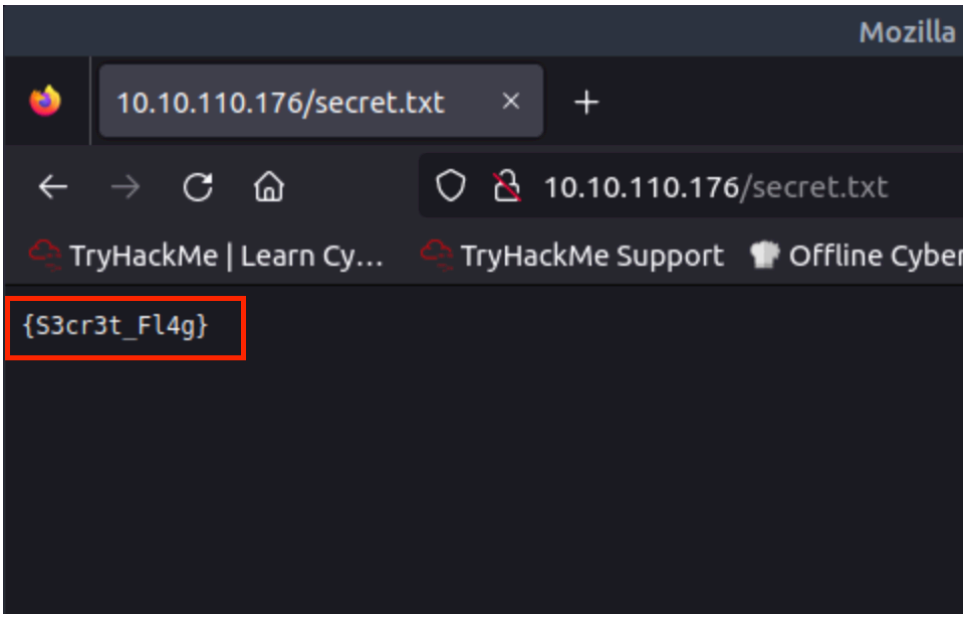
The ninth flag found

|{H1d3_Fl4g}



The tenth flag found

|{Robots_Flag}



The eleventh flag found.

|{S3cr3t_Fl4g}

I utilized Linux to find the last two hidden flags. I executed the same command I used in file system flags: "*find /var -type f -name "*.txt" 2>/dev/null*".
The hidden flags could be located anywhere, files, network, webpage etc. After executing the command, I found two directories I haven't used yet which are boxed in white.

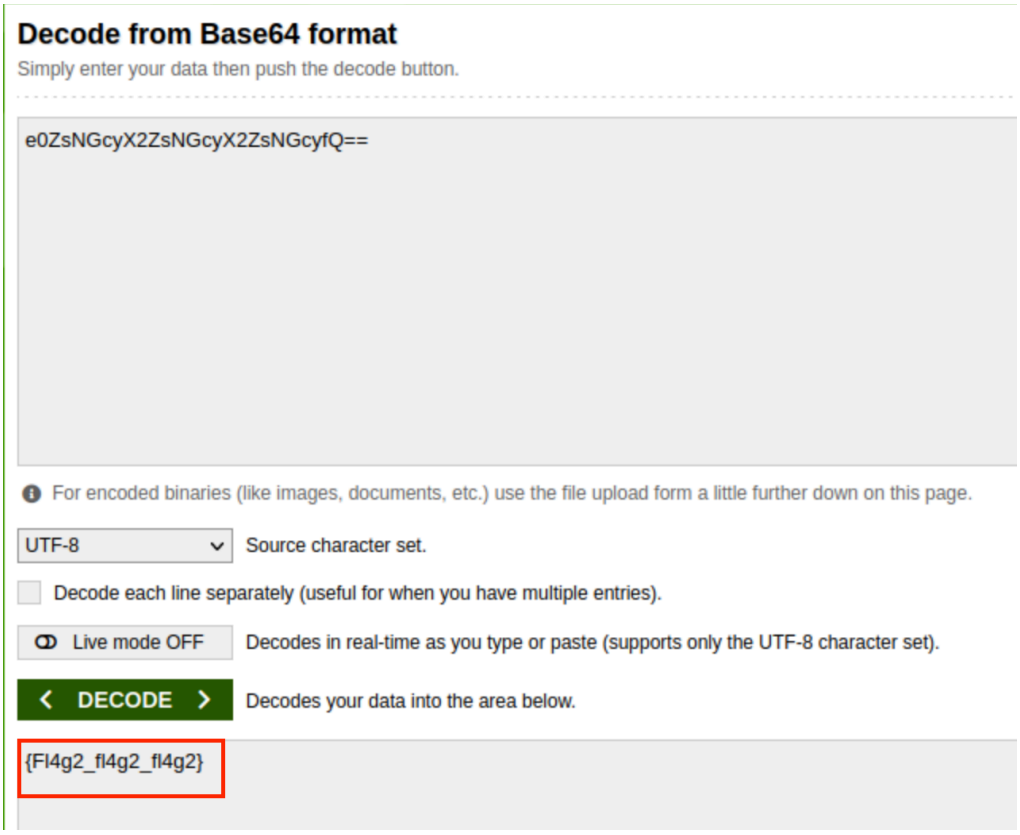
```
ctf@Masterschool:~$ find /var -type f -name "*.txt" 2>/dev/null
/var/www/html/secret.txt
/var/www/html/robots.txt
/var/www/html/flag/flag/flag.txt
/var/www/html/flag/flag/flag2.txt
/var/backups/find_flag.txt
/var/log/installer/installer-journal.txt
/var/lib/cloud/instances/iid-datasource-none/vendor-data2.txt
/var/lib/cloud/instances/iid-datasource-none/vendor-data.txt
/var/lib/cloud/instances/iid-datasource-none/user-data.txt
/var/lib/cloud/instances/iid-datasource-none/cloud-config.txt
ctf@Masterschool:~$
```

```
ctf@Masterschool:~$ cat /var/www/html/flag/flag/flag.txt
{Fl4g_fl4g_fl4g}
ctf@Masterschool:~$ cat /var/www/html/flag/flag/flag2.txt
e0ZsNGcyX2ZsNGcyX2ZsNGcyfQ==
ctf@Masterschool:~$
```

I used the cat command to unveil the hidden flag. Where we found the twelfth flag.

| {Fl4g_fl4g_fl4g}

The next flag needed to be decoded, so I needed to use a web source to decode. I used a Base64 format to decode the thirteenth flag.



The thirteenth flag found.

| {Fl4g2_fl4g2_fl4g2}

Third and Fourth Challenge done!!

Hashing

Hash cracking is part of cryptography which plays a significant role in cybersecurity. This challenge requires the use of tools such as hashcat, john the ripper, hashid, and other websites to decrypt hashes.

For this challenge, the first step was to set up proper tools on the attack machine. I needed to connect to the root server and install hashid and john the ripper which will allow me to identify hashes and hash types. To start the process, I connected to Masterschool CTF server using scp and downloaded **/hash_to_crack** directory which contained several hashes .txt files.

The "SCP" command allows secure transferring of files between local host and the remote host, or between two remote hosts.

```
root@ip-10-10-88-128:~# scp -r ctf@10.10.121.128:~/hash_to_crack ~/hash_to_crack
##################################################################
#                        Welcome to Masterschool's CTF
#                        First flag: {h4ck3r5_r_us}
##################################################################
ctf@10.10.121.128's password:
hash1.txt                                     100%   33     0.4KB/s   00:00
hash2.txt                                     100%   41     0.6KB/s   00:00
hash3.txt                                     100%  129     1.8KB/s   00:00
hash4.txt                                     100%   65     0.8KB/s   00:00
hash5.txt                                     100%   33     0.4KB/s   00:00
wordlist.txt                                  100% 6446     1.6MB/s   00:00
root@ip-10-10-88-128:~#
```

The next step is to download hashid. To do that, I must switch directory to *"/hash_to_crack*" and install hashid.

```
root@ip-10-10-88-128:~# cd hash_to_crack
root@ip-10-10-88-128:~/hash_to_crack# apt install hashid
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docutils-common gir1.2-goa-1.0 gir1.2-snapd-1 libpkcs11-helper1
  linux-headers-4.15.0-115 linux-headers-4.15.0-115-generic
  linux-image-4.15.0-115-generic linux-modules-4.15.0-115-generic
  linux-modules-extra-4.15.0-115-generic python-bs4 python-chardet
  python-dicttoxml python-dnspython python-html5lib python-jsonrpclib
  python-lxml python-mechanize python-olefile python-pypdf2 python-slowaes
  python-webencodings python-xlsxwriter python3-botocore python3-docutils
  python3-jmespath python3-pygments python3-roman python3-rsa
  python3-s3transfer xml-core
Use 'apt autoremove' to remove them.
The following NEW packages will be installed
  hashid
0 to upgrade, 1 to newly install, 0 to remove and 107 not to upgrade.
Need to get 12.9 kB of archives.
After this operation, 86.0 kB of additional disk space will be used.
Get:1 http://eu-west-1.ec2.archive.ubuntu.com/ubuntu bionic/universe amd64 hashi
d all 3.1.4-2 [12.9 kB]
Fetched 12.9 kB in 0s (0 B/s)
Selecting previously unselected package hashid.
(Reading database ... 482629 files and directories currently installed.)
Preparing to unpack .../hashid 3.1.4-2 all.deb ...
```

Figure 21: installation of hashid

Now we must install John the Ripper. Same process as we took to install hashid.

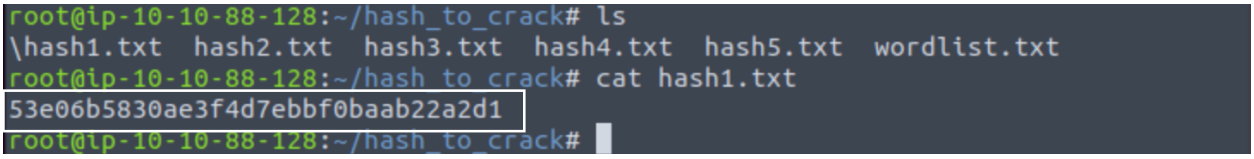
```
root@ip-10-10-88-128:~/hash_to_crack# apt install john
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  docutils-common gir1.2-goa-1.0 gir1.2-snapd-1 libpkcs11-helper1
  linux-headers-4.15.0-115 linux-headers-4.15.0-115-generic
  linux-image-4.15.0-115-generic linux-modules-4.15.0-115-generic
  linux-modules-extra-4.15.0-115-generic python-bs4 python-chardet
  python-dicttoxml python-dnspython python-html5lib python-jsonrpclib
  python-lxml python-mechanize python-olefile python-pypdf2 python-slowaes
  python-webencodings python-xlsxwriter python3-botocore python3-docutils
  python3-jmespath python3-pygments python3-roman python3-rsa
  python3-s3transfer xml-core
Use 'apt autoremove' to remove them.
The following additional packages will be installed:
  john-data
The following NEW packages will be installed
  john john-data
0 to upgrade, 2 to newly install, 0 to remove and 107 not to upgrade.
Need to get 4,466 kB of archives.
After this operation, 7,875 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Figure 22: Installation of John The Ripper

Now that we have all the necessary tools, we can start hashing. We have stay in the hash directory to be able to finish this smoothly.

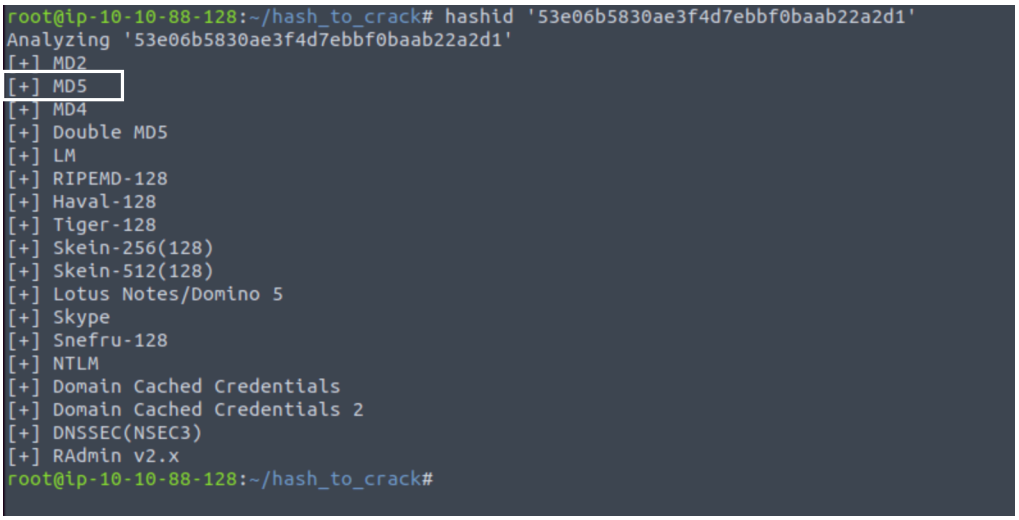
As always, using ls to see the list of files and directories and we can see that there are five hashes. I will be using the cat command unveil the hashes.

In this case, our command is “cat hash1.txt”. which showed a 32-digit hexadecimal numbers.



A couple things to keep in mind that, the format of the hash will help us crack the hash. It will tell us which hash algorithm to use to crack the hash. For example, if the hash is MD5, the syntax will contain “*—format=raw-MD5*”. And if its SHA1, it will be formatted as “*—format=raw-SHA!*”. You get the idea.

To identify the format of the hash, I type in hashid and pasted the hash using single or double quotation marks.



Now that we know what kind of hash algorithm is used, let’s get cracking.

I ran john with –format=raw—MD5 -wordlist=wordlist.txt option.

```
root@ip-10-10-88-128:~/hash_to_crack# john hash1.txt --format=raw-MD5 -wordlist=wordlist.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
C0d3_0b5cur3r_Flag (?)
1g 0:00:00:00 DONE (2023-08-13 01:46) 25.00g/s 7500p/s 7500c/s 7500C/s 7h3_H4ck3r_FL4g..C0d3_Fl4g
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

It ran successfully and we have found our first hash and fourteenth flag.

| Cod3_0b5cur3r_flag

This methodology will be repeated to crack the remaining hashes.

```
root@ip-10-10-88-128:~/hash_to_crack# cat hash2.txt
a6938e05ec33e356ff4b9aa961fe1e51138b4758
root@ip-10-10-88-128:~/hash_to_crack# hashid 'a6938e05ec33e356ff4b9aa961fe1e51138b4758'
Analyzing 'a6938e05ec33e356ff4b9aa961fe1e51138b4758'
[+] SHA-1
[+] Double SHA-1
[+] RIPEMD-160
[+] Haval-160
[+] Tiger-160
[+] HAS-160
[+] LinkedIn
[+] Skein-256(160)
[+] Skein-512(160)
root@ip-10-10-88-128:~/hash_to_crack#
```

```
root@ip-10-10-88-128:~/hash_to_crack# john hash2.txt --format=raw-SHA1 -wordlist=wordlist.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA1 [SHA1 256/256 AVX2 8x])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
C0d3_5l4y3r_Flag (?)
1g 0:00:00:00 DONE (2023-08-13 01:55) 100.0g/s 20800p/s 20800c/s 20800C/s S3cure_S0c14l_3ng1n33r_3xp3rt_Flag
..C0d3_5l4y3r_Flag
Use the "--show --format=Raw-SHA1" options to display all of the cracked passwords reliably
Session completed.
root@ip-10-10-88-128:~/hash_to_crack#
```

Fifteenth flag found.

| Cod3_5l4y3r_Flag

```
root@ip-10-10-88-128:~/hash_to_crack# cat hash3.txt
a15c292682ac51a76b7f25ec341707fc8967025d007a52c0fa8e565dfe2f7a5bca162e6b2fe8cd8f75c62192604f66df73d1a4028299
f03c07fbc2dc6650b029
root@ip-10-10-88-128:~/hash_to_crack# hashid 'a15c292682ac51a76b7f25ec341707fc8967025d007a52c0fa8e565dfe2f7a
5bca162e6b2fe8cd8f75c62192604f66df73d1a4028299f03c07fbc2dc6650b029'
Analyzing 'a15c292682ac51a76b7f25ec341707fc8967025d007a52c0fa8e565dfe2f7a5bca162e6b2fe8cd8f75c62192604f66df7
3d1a4028299f03c07fbc2dc6650b029'
[+] SHA-512
[+] Whirlpool
[+] Salsa10
[+] Salsa20
[+] SHA3-512
[+] Skein-512
[+] Skein-1024(512)
root@ip-10-10-88-128:~/hash_to_crack# john hash3.txt --format=raw-SHA512 -wordlist=wordlist.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA512 [SHA512 256/256 AVX2 4x])
Warning: poor OpenMP scalability for this hash type, consider --fork=2
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
H4ck3r_Flag      (?)
1g 0:00:00:00 DONE (2023-08-13 01:58) 33.33g/s 10000p/s 10000c/s 10000C/s 7h3_H4ck3r_FL4g..C0d3_Fl4g
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
root@ip-10-10-88-128:~/hash_to_crack#
```

Sixteenth Flag found.

| H4ck3r_Flag

```
root@ip-10-10-88-128:~/hash_to_crack# cat hash4.txt
d1d0f39e3be116c81453d7af22c3623ec555d007cdf77a9813e9647dfcc2cfaa
root@ip-10-10-88-128:~/hash_to_crack# hashid 'd1d0f39e3be116c81453d7af22c3623ec555d007cdf77a9813e9647dfcc2cf
aa'
Analyzing 'd1d0f39e3be116c81453d7af22c3623ec555d007cdf77a9813e9647dfcc2cfaa'
[+] Snefru-256
[+] SHA-256
[+] RIPEMD-256
[+] Haval-256
[+] GOST R 34.11-94
[+] GOST CryptoPro S-Box
[+] SHA3-256
[+] Skein-256
[+] Skein-512(256)
root@ip-10-10-88-128:~/hash_to_crack# john hash4.txt --format=raw-SHA256 -wordlist=wordlist.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA256 [SHA256 256/256 AVX2 8x])
Warning: poor OpenMP scalability for this hash type, consider --fork=2
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
L0ck_Flag        (?)
1g 0:00:00:00 DONE (2023-08-13 02:02) 100.0g/s 30000p/s 30000c/s 30000C/s 7h3_H4ck3r_FL4g..C0d3_Fl4g
Use the "--show --format=Raw-SHA256" options to display all of the cracked passwords reliably
Session completed.
root@ip-10-10-88-128:~/hash_to_crack#
```

Seventeenth Flag found.

| Lock_Flag

Last hash and our Eighteenth flag found.

| S3cur1ty_Flag

Fifth Challenge Complete!!

Nmap Scan Report

Nmap stands for “Network Mapper” which is a free security scanner. It is used for network discovery, host discovery and security auditing. I had to research and utilize chatgpt and google to start the process of Nmap.

The process of nmap allows the evaluation of potential vulnerabilities.

There are a few ways to get in the nmap system. I tried the aggressive (-a) method which was supposedly most useful in some of my peers’ opinions. This was also a very quick and simple way.

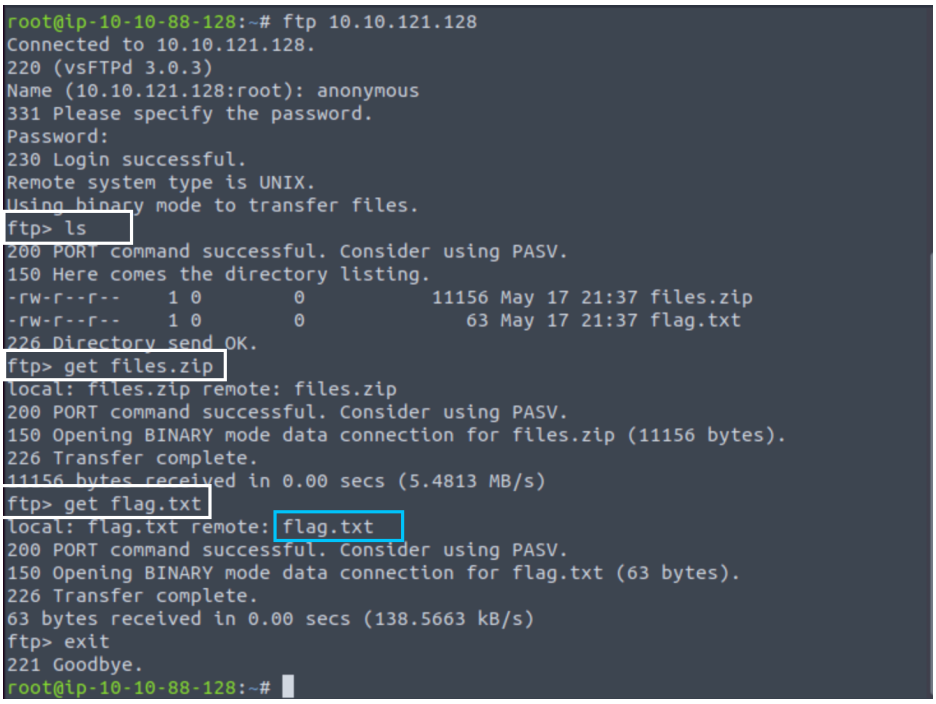
To being, I connected to my root server and typed in “nmap -A ‘machine IP Address’ ”.

```
root@ip-10-10-47-53:~# nmap -A 10.10.153.127

Starting Nmap 7.60 ( https://nmap.org ) at 2023-09-04 00:44 BST
Nmap scan report for ip-10-10-153-127.eu-west-1.compute.internal (10.10.153.127)
Host is up (0.00057s latency).
Not shown: 991 closed ports
PORT    STATE SERVICE    VERSION
21/tcp  open  ftp        vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| -rw-r--r--    1 0        0           11156 May 17 21:37 files.zip
|_-rw-r--r--    1 0        0              63 May 17 21:37 flag.txt
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to ::ffff:10.10.47.53
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 1
|      vsFTPd 3.0.3 - secure, fast, stable
|_End of status
22/tcp  open  ssh        OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
25/tcp  open  smtp       Postfix smtpd
|_smtp-commands: Masterschool.Masterschool.com, PIPELINING, SIZE 10240000, VRFY, ETRN, STA
RTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN, SMTPUTF8, CHUNKING,
53/tcp  open  domain     ISC BIND 9.16.1-Ubuntu
| dns-nsid:
|_  bind.version: 9.16.1-Ubuntu
80/tcp  open  http       Apache httpd 2.4.41 ((Ubuntu))
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
110/tcp open  pop3       Dovecot pop3d
|_pop3-capabilities: SASL TOP RESP-CODES AUTH-RESP-CODE STLS CAPA PIPELINING UIDL
143/tcp open  imap       Dovecot imapd (Ubuntu)
|_imap-capabilities: capabilities SASL-IR OK more LITERAL+ IDLE have post-login IMAP4rev1
Pre-login STARTTLS LOGINDISABLEDA0001 ID ENABLE LOGIN-REFERRALS listed
993/tcp open  tcpwrapped
995/tcp open  tcpwrapped
MAC Address: 02:00:79:03:E2:25 (Unknown)
```

Figure 31: access to namp scan

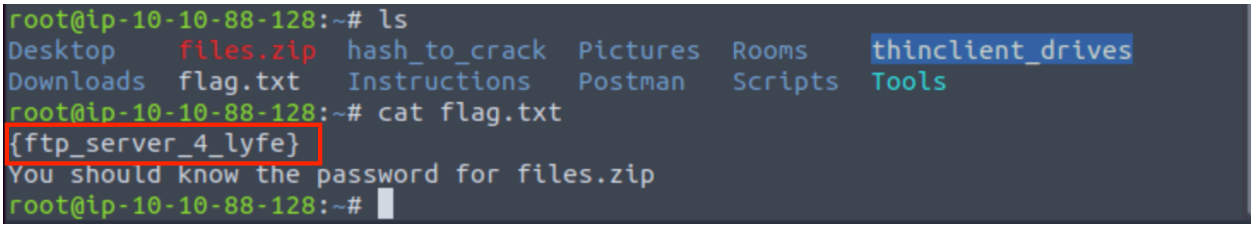
As soon as it connects, we can see that anonyms log in was allowed for the FTP port. There are few other ports that are open and accessible such as HTTP. These ports had a huge hand in finding hidden and webpage flags. Since HTTP is open and accessible, I was able to find webpage flags by just entering the IP address of the machine.

Getting into the FTP server was easy. I entered in "*ftp 'machine IP address'*" which connected me to the machine, I than had to log in anonymously, entered in the password and I was in the system.



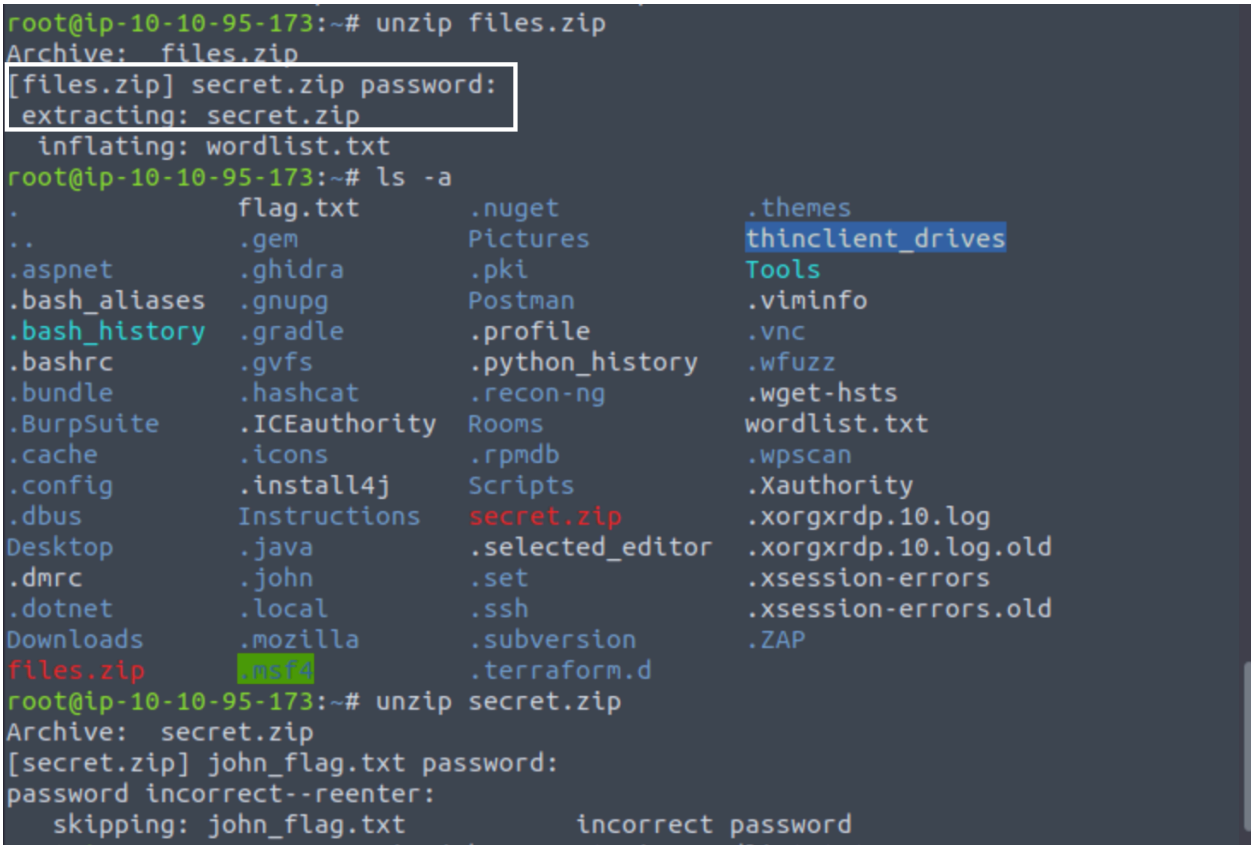
From there, I entered ls to see the list of contents in the FTP directory, which displayed there are two files available for download, **files.zip** and **flag.txt**. The next step is to download these files. To download the files, all I had to type in was "***get files.zip***" and get "***flag.txt***", as displayed in above figure.

I know that there's a flag hidden in that FTP server because of the "***flag.txt***". To find our hidden flag, all I had to do was "**cat flag.txt**" and we found the nineteenth flag.



| {ftp_server_4_lyfe}

To find the last flag, I will be analyzing the file.zip. The first step is to unzip the file. When I tried to unzip the file, it asked me for a password. Since this CTF was created by Masterschool, I guessed “Masterschool” as the password and now I had access.



Highlighted in red, there’s a file named “secret.zip”. I knew this is a file I need to get access to as it may contain the last hidden flag. I was unsuccessful when I tired the same password, “masterschool”, so I had to seek aid from my peers. They gave me a hint. They suggested I use zip2john.

```
root@ip-10-10-95-173:~# unzip secret.zip
Archive:  secret.zip
[secret.zip] john_flag.txt password:
password incorrect--reenter:
   skipping: john_flag.txt           incorrect password
root@ip-10-10-95-173:~# zip2john secret.zip wordlist.txt
ver 1.0 efh 5455 efh 7875 secret.zip/john_flag.txt PKZIP Encr: 2b chk, TS_chk, c
mplen=28, decmplen=16, crc=DB6B1364 type=0
secret.zip/john_flag.txt:$pkzip2$1*2*2*0*1c*10*db6b1364*0*47*0*1c*db6b*ac74*fd53
ecb2754b91e36f3e1fe2e4f8a3cc64fdcdbb38fe4ddc1f3504f4*$/pkzip2$:john_flag.txt:sec
ret.zip::secret.zip
Did not find End Of Central Directory.
```

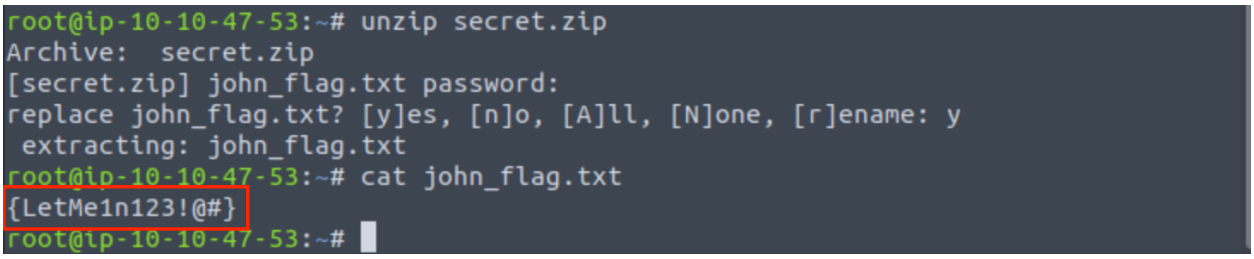
I converted the secret.zip file to "**zip2john secret.zip wordlist.txt**", as a hash and it gave me a very long hash boxed in blue. I know I can now use John The Ripper to crack the hash and get the password.

```
root@ip-10-10-95-173:~# john hash.txt --wordlist=wordlist.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
CTF_TIME         (?)
1g 0:00:00:00 DONE (2023-08-13 03:58) 50.00g/s 135050p/s 135050c/s 135050C/s 123
456..19871987
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

I have been successful, and the password is "CTF_TIME". Now that I know the password, I can unzip the file.

I can now brute force my way into the file. The command I executed for this was "**unzip -P CTF_TIME secret.zip**" The "-P" option looks for executable files. Another way of doing this was by going back to unzip the file and entering in the password which is "CTF_TIME". For some reason it gave me extra line of command which I typed "y".

```
root@ip-10-10-95-173:~# unzip -P CTF_TIME secret.zip
Archive:  secret.zip
 extracting: john_flag.txt
root@ip-10-10-95-173:~# cat john_flag.txt
{LetMe1n123!@#}
root@ip-10-10-95-173:~#
```



There we have it, the last flag. Twentieth flag found.

| {LetMe1n123!@#}

After completing this nmap scan, it is clear the server needs stronger security. I was able to get access to vulnerable files easily. Here are some things I noticed:

Unauthorized access was allowed on FTP port 21, which is a risk as anyone can download and get access to the files. This can be avoided by disabling unauthorized log in's and put an extra layer of security such as "security questions".

There were plain text protocols such as FTP, POP3 and SMPT, which means traffic was sent through these protocols uncrypted making it more vulnerable for theft and fraud. One way to avoid this is by allowing a version of SSL or TLS of FTP, SMTP, IMAP etc.

It is crucial to have secure and protected data. Few ways this can be done is by:

1. Encrypting the connection.
2. Defending user accounts and permissions
3. Enforcing password compliance
4. Detecting and responding to password attack.
    a. Number of times a password can be incorrect.
5. Enabling time and IP limits.

Conclusion

The process of finding the flags was a bit challenging. There were times when I was unable to come up with a solution, so I sought aid from peers and AI. Which is encouraged as a cybersecurity professional.

The challenge allowed me to think outside the box and practice my communication with online aid, and my attention to detail as flags were hidden anywhere and everywhere. The most challenging part about this practice was the nmap scan, finding the last hidden flag. I was unsure how to unzip the file. Unzipping secret.zip gave me the name of the file but I was unsuccessful with guessing the password. After asking my peers, I was able to figure out I had to hash the file and find the password by using john the ripper. This process took me about a couple hours.

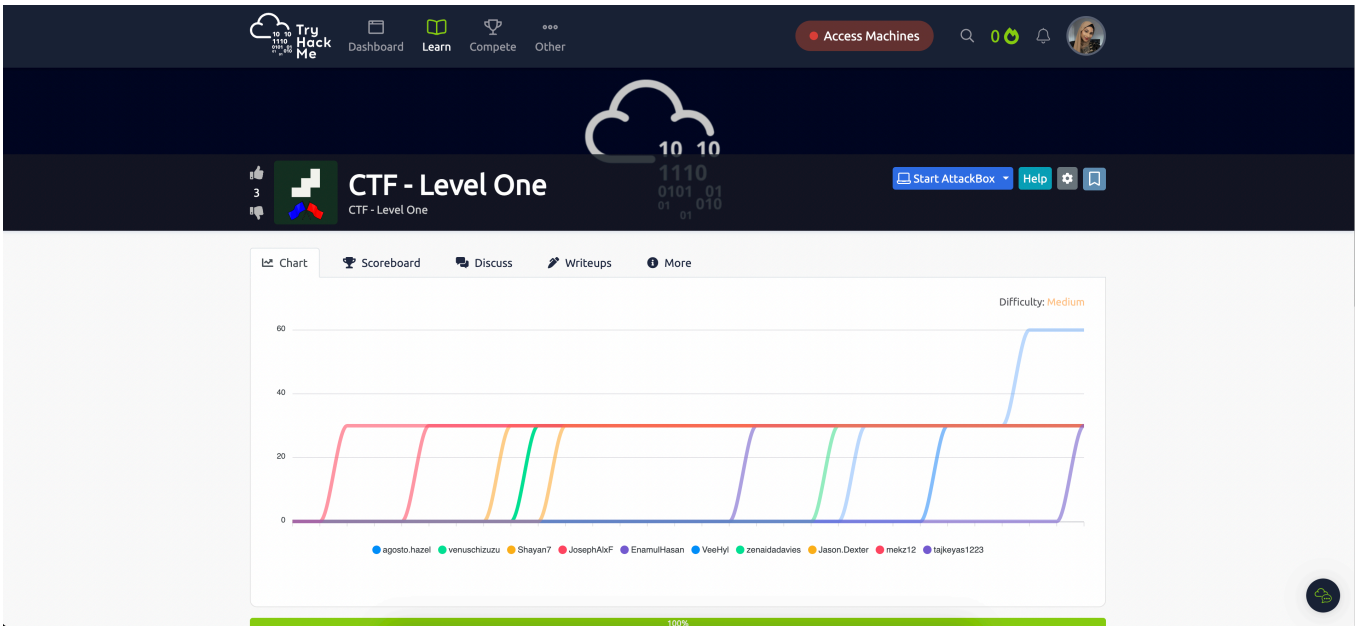
The best and the easiest part about this challenge was the hashing. It was quite simple, and I had so much fun cracking the hashes. There were five hashes, and the same steps were taken to crack the hash. The only challenge I faced during this part of the challenge was copying and pasting the hash into “hashid”. I figured it out in the end.

Overall, this challenge gave me a lot of insights on security, encryption, SSL protocols and much more. If I were to do this challenge again, I will be able to complete it without much struggle.

This challenge has also proven the skills I am learning at masterschools will continue to enlighten my knowledge of cybersecurity.

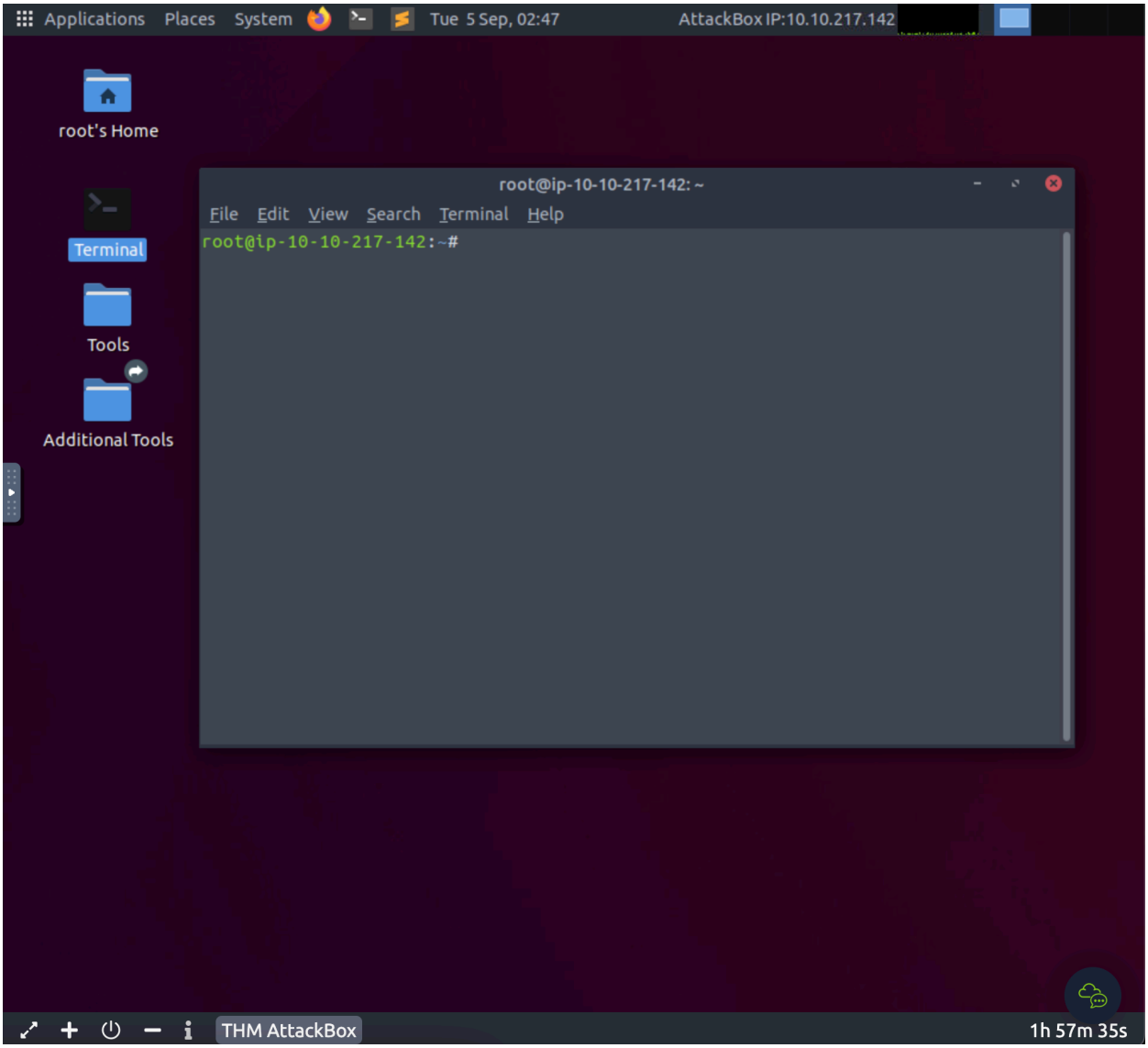
Appendix

TryhackMe

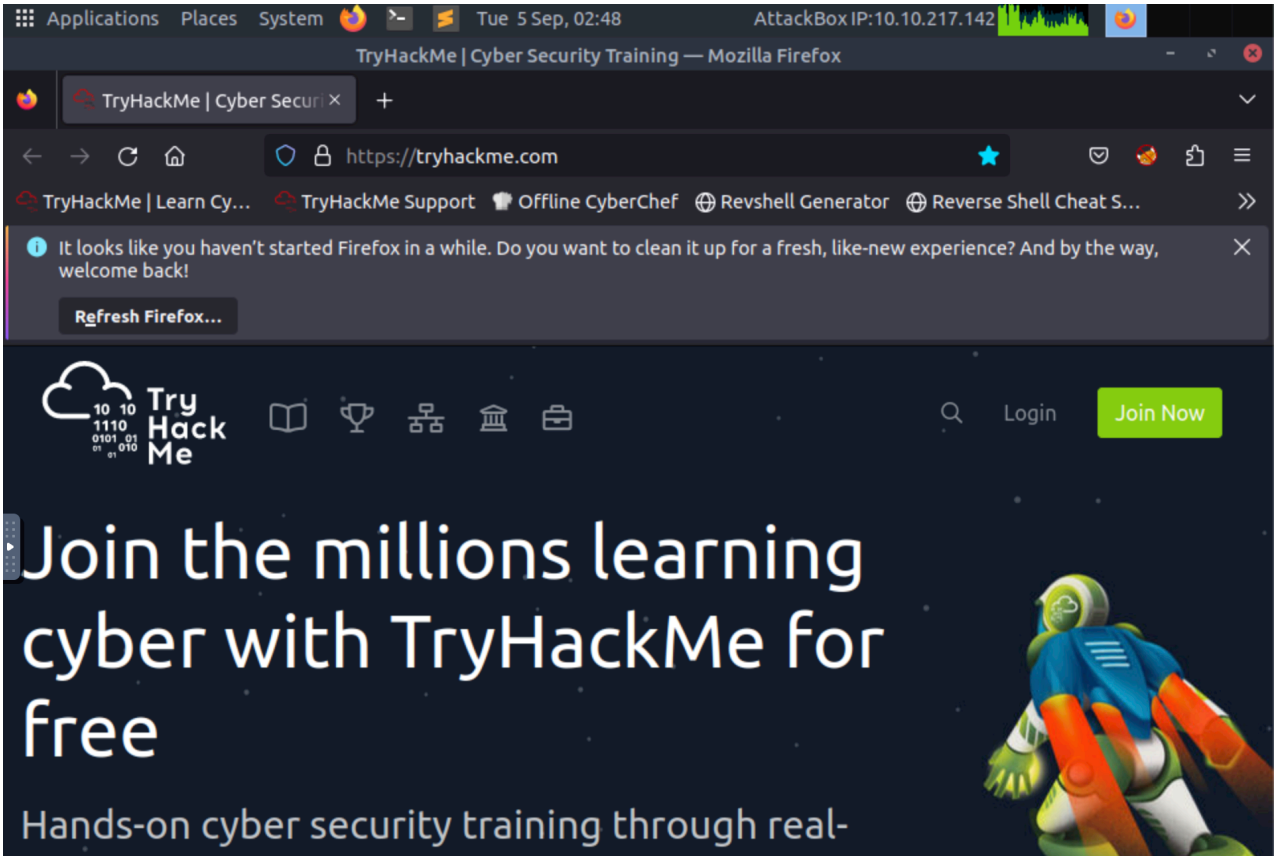


ChatGPT – for proper command lines and syntax to use.

AttackBox (Virtual Machine)



VM Firefox Web browser – Used when finding webpage flags



Tools used to decode:

Hashcat

John The Ripper

HashID