MASTERSCHOOL CTF

Richard Giadom

TABLE OF CONTENTS

INTRODUCTION

Hey there,

Just recently, I took on a challenge that pushed my boundaries in the world of cybersecurity – the MasterSchool CTF. If you are not familiar, CTF stands for Capture The Flag, it’s a mixture of hacking skills and puzzle-solving.

During this event, I got hands on with a variety of challenges that ranged from understanding Linux basics and managing files to deciphering hidden messages buried within file systems and web pages. I even cracked some hashes and deep dived into Nmap scan reports to unravel network secrets.

Believe me, it was not all smooth sailing. There were moments of frustration where I felt like throwing in the towel, but I persevered. Each challenge I conquered was like a puzzle piece, fitting into the bigger picture of what I was learning.

More than just technical skills, this experience changed my mindset. It made me realize that growth happens when you step out of your comfort zone and tackle the things that seem impossible at first. This CTF was not just about winning; it was about the journey of self-discovery, confidence and skill-building.

I am excited to continue my journey into the world of cybersecurity, armed with the knowledge and confidence I gained from this CTF. As I move forward, I am eager to dive deeper into Linux, explore hidden gems in files and web pages, and crack even more codes, hashes and passwords.

This MasterSchool CTF was just the beginning, and I am ready to face whatever challenges come next.

LINUX BASICS: USERS AND FILE MANAGEMENT

This task was created to test how proficient I am at using the Linux command line. It was split into four separate tasks.

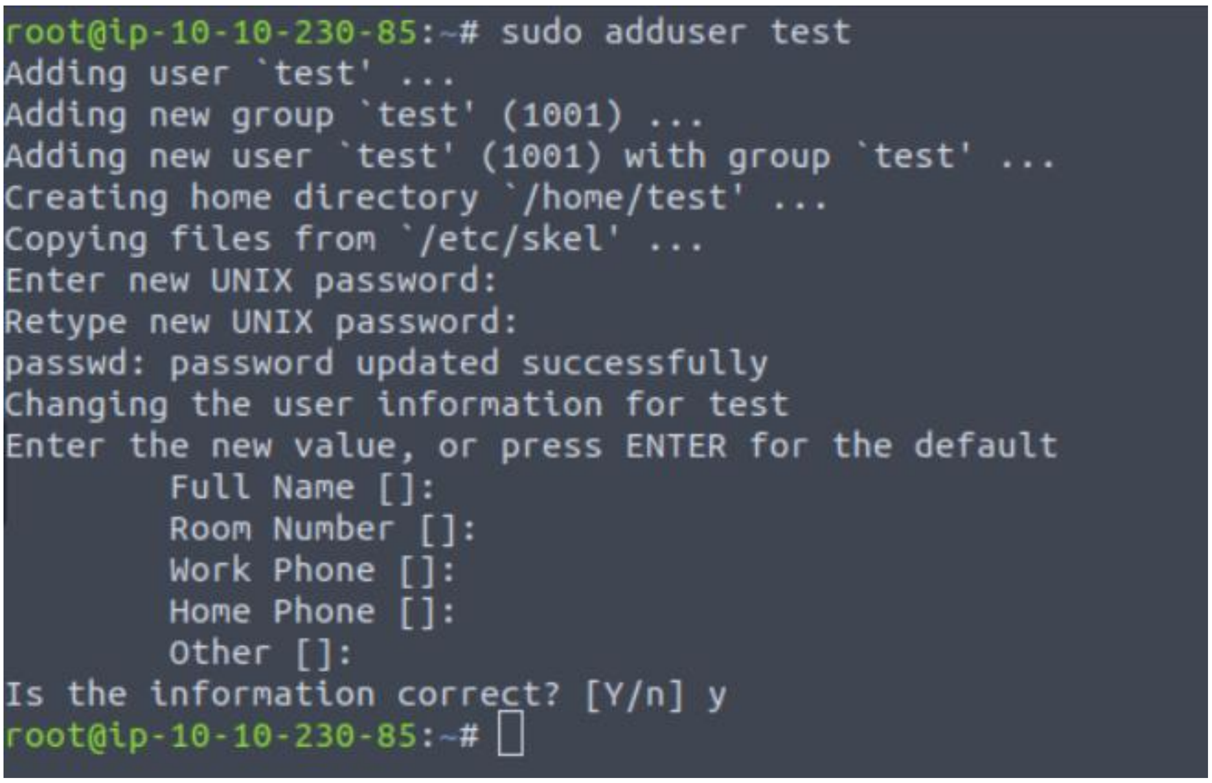
1) create a new user and assign a password.

2) switch the session to the new user.

3) As the new user create a new directory then create a new file, leave a message inside the directory, and save it.

4) Switch back to the original CTF user.

In order to complete this challenge, I had to complete each step one at time. Starting at the beginning I used the command sudo adduser test (test) is the name of the new user created. Once that was set up, I was prompted to create a password and then I simply followed the instructions to complete the task as you can see in the image below.

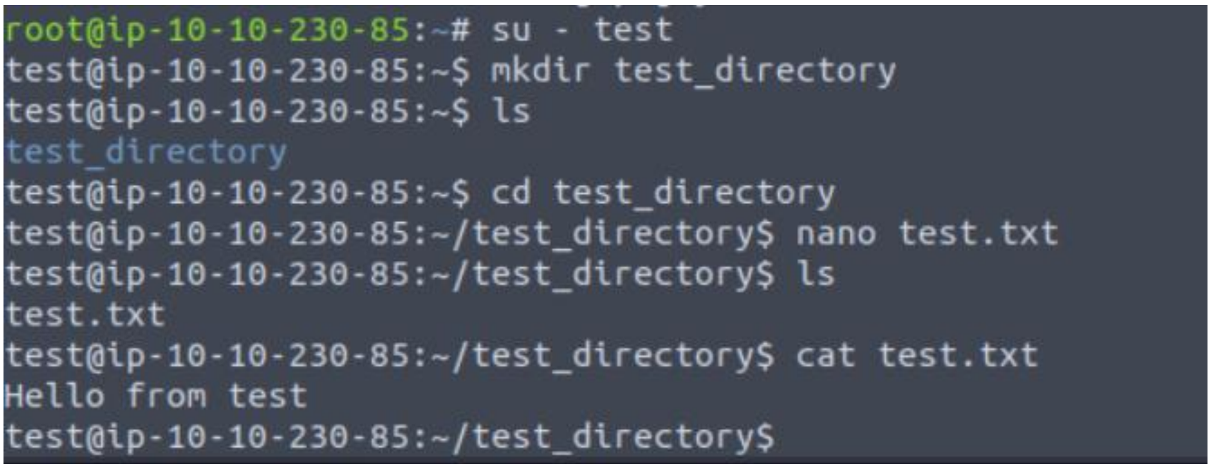


```
root@ip-10-10-230-85:~# sudo adduser test
Adding user `test' ...
Adding new group `test' (1001) ...
Adding new user `test' (1001) with group `test' ...
Creating home directory `/home/test' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for test
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
root@ip-10-10-230-85:~# 
```

To complete task 2 I simply used the command su – test which switched straight to the new user.

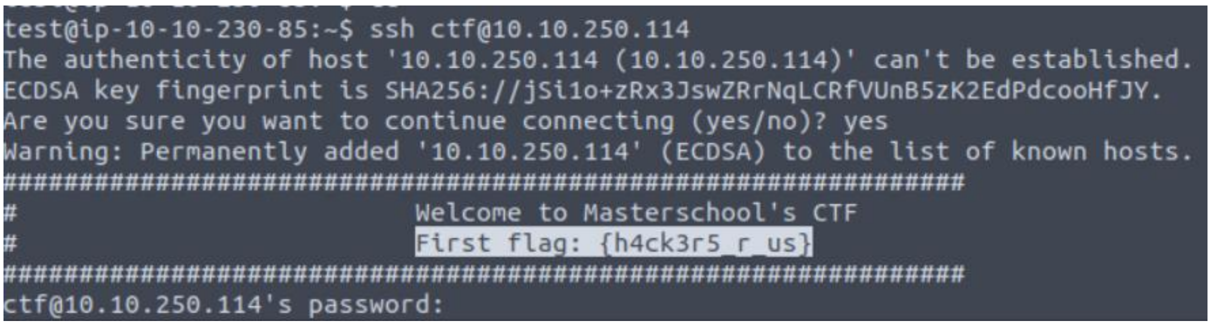
Task three was slightly more complicated. I firstly used the command mkdir test_directory which allowed me to create a new directory and then to make sure it was created I used ls, the list command which showed the directory was there highlighted in blue.

The next step was to create a new file within the directory. I used the nano editor creating a file called test.txt. As you can see in the image below the file was created and stored with a message inside reading Hello from test.



```
root@ip-10-10-230-85:~# su - test
test@ip-10-10-230-85:~$ mkdir test_directory
test@ip-10-10-230-85:~$ ls
test_directory
test@ip-10-10-230-85:~$ cd test_directory
test@ip-10-10-230-85:~/test_directory$ nano test.txt
test@ip-10-10-230-85:~/test_directory$ ls
test.txt
test@ip-10-10-230-85:~/test_directory$ cat test.txt
Hello from test
test@ip-10-10-230-85:~/test_directory$
```

To finish off this part of the CTF I had to switch to the CTF user. In order to do this I used the command ssh ctf@10.10.250.114. This ip address is the ip assigned to me by Try Hack Me. Once I reached this page, I was immediately greeted with my first flag {h4ck3r5_r_us}.



```
test@ip-10-10-230-85:~$ ssh ctf@10.10.250.114
The authenticity of host '10.10.250.114 (10.10.250.114)' can't be established.
ECDSA key fingerprint is SHA256://jSi1o+zRx3JswZRrNqLCRfVUnB5zK2EdPdcooHfJY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.250.114' (ECDSA) to the list of known hosts.
###############################################################
#                          Welcome to Masterschool's CTF
#                          First flag: {h4ck3r5_r_us}
###############################################################
ctf@10.10.250.114's password:
```

FILE SYSTEM FLAGS

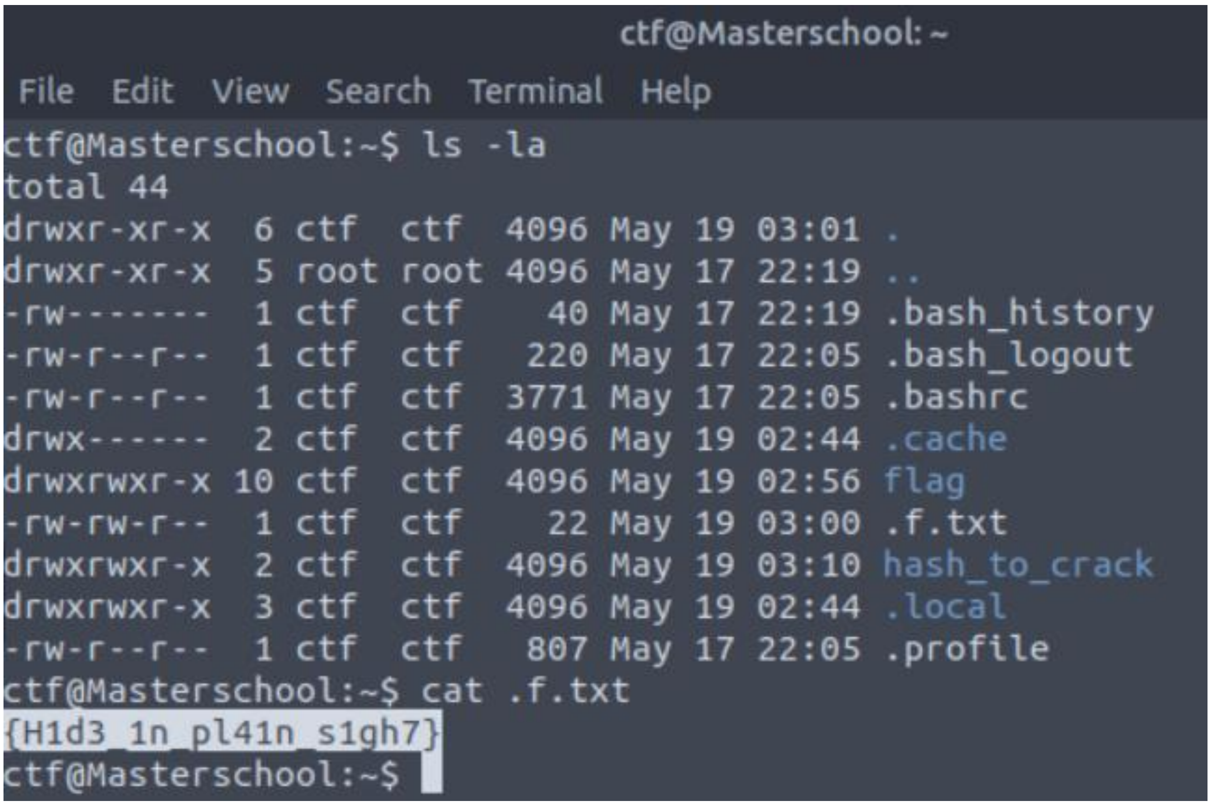
The goal of this challenge was really just to find as many flags as possible. I started off by using the command ls which showed two directories. The first a directory called flag and the second named hash_to_crack.

I started off by navigating the flag directory. Inside of there I found an additional eight directories and a file.

Having looked through all of them I found a flag hidden in the file named story.txt {St0ry_Fl4g} and in directory number 6 (Y0u_G0T_1t}.

All of the files in the hash_to_crack directory were hashes which will be useful for later on in this CTF but for now my focus was on finding flags, so I decided to navigate deeper within the file system.

By using the command ls -la I was able to locate some more files and directories and navigating through them I found another flag in the .f.txt {H1d3_1n_pl41n_s1gh7}.



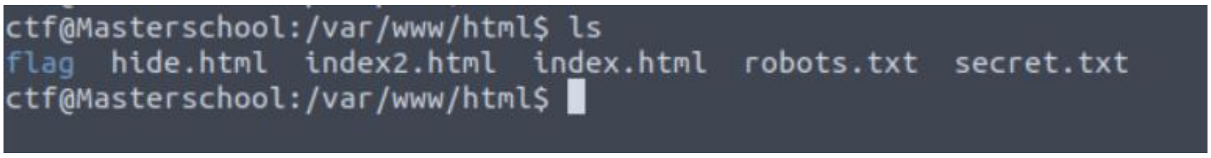
```
ctf@Masterschool:~$ ls -la
total 44
drwxr-xr-x  6 ctf  ctf  4096 May 19 03:01 .
drwxr-xr-x  5 root root 4096 May 17 22:19 ..
-rw-------  1 ctf  ctf    40 May 17 22:19 .bash_history
-rw-r--r--  1 ctf  ctf   220 May 17 22:05 .bash_logout
-rw-r--r--  1 ctf  ctf  3771 May 17 22:05 .bashrc
drwx------  2 ctf  ctf  4096 May 19 02:44 .cache
drwxrwxr-x 10 ctf  ctf  4096 May 19 02:56 flag
-rw-rw-r--  1 ctf  ctf    22 May 19 03:00 .f.txt
drwxrwxr-x  2 ctf  ctf  4096 May 19 03:10 hash_to_crack
drwxrwxr-x  3 ctf  ctf  4096 May 19 02:44 .local
-rw-r--r--  1 ctf  ctf   807 May 17 22:05 .profile
ctf@Masterschool:~$ cat .f.txt
{H1d3_1n_pl41n_s1gh7}
ctf@Masterschool:~$
```

WEBPAGE & HIDDEN FLAGS

The purpose of this task was to locate hidden flags within webpages.

At first I was going to use gobuster as its mostly used for reconnaissance and information gathering on web servers but then I decided to look internally on this system specifically on the /var/www/html path as it generally holds the information related to web servers as well as website files and content that is served to site visitors.

After navigating to the var directory there was a file named flag and some accompanying files.



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

In the index.html I found {STUDENT_CTF_Web}

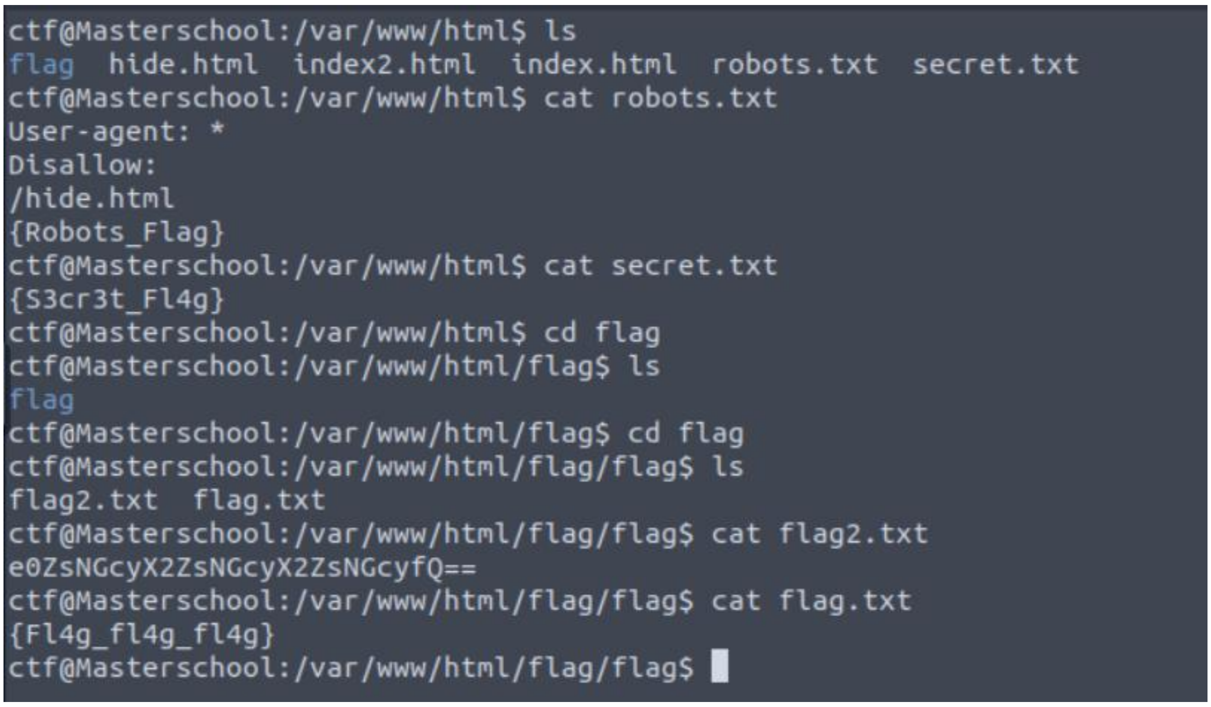
Index2.html, I found {C0nf1gur4t10n_Fl4g}

In hide.html I found the flag {H1d3_Fl4g}

In the file robots.txt I found the flag {Robots_Flag}

In the file secret.txt I also found {S3cr3t_Fl4g}

In the directory of flag there was also a hidden flag {Fl4g_fl4g_fl4g}



```
ctf@Masterschool:/var/www/html$ ls
flag   hide.html   index2.html   index.html   robots.txt   secret.txt
ctf@Masterschool:/var/www/html$ cat robots.txt
User-agent: *
Disallow:
/hide.html
{Robots_Flag}
ctf@Masterschool:/var/www/html$ cat secret.txt
{S3cr3t_Fl4g}
ctf@Masterschool:/var/www/html$ cd flag
ctf@Masterschool:/var/www/html/flag$ ls
flag
ctf@Masterschool:/var/www/html/flag$ cd flag
ctf@Masterschool:/var/www/html/flag/flag$ ls
flag2.txt   flag.txt
ctf@Masterschool:/var/www/html/flag/flag$ cat flag2.txt
e0ZsNGcyX2ZsNGcyX2ZsNGcyfQ==
ctf@Masterschool:/var/www/html/flag/flag$ cat flag.txt
{Fl4g_fl4g_fl4g}
ctf@Masterschool:/var/www/html/flag/flag$ 
```

HASH CRACKING

For this task I will be cracking the hashes found earlier in the hash_to_crack directory. The hashes I found were: -

hash1.txt - 53e06b5830ae3f4d7ebbf0baab22a2d1

hash2.txt - a6938e05ec33e356ff4b9aa961fe1e51138b4758

hash3.txt - a15c292682ac51a76b7f25ec341707fc8967025d007a52c0fa8e565dfe2f7a5bca162e6b2fe8cd8f75c62192604f66df73d1a4028299f03c07fbc2dc6650b029

hash4.txt - d1d0f39e3be116c81453d7af22c3623ec555d007cdf77a9813e9647dfcc2cfaa

hash5.txt - b9c86725a1c15a6af0e7b595b25b8d3a

I tried cracking them on Crackstation but none of them worked there so instead I used a website called https://www.tunnelsup.com/hash-analyzer/ for the hash types and then I decided to use https://hashcat.net/wiki/doku.php?id=example_hashes for the hash id. An alternative to this would have been Hash-ID.

Hash1 is Raw md5 – code 0

Hash2 is sha1 – code 100

Hash3 is sha 512 – code 1700

Hash4 is sha 256 – code 1400

Hah5 is md5 – code 0

I will be using john the ripper to crack these hashes. John the Ripper is a widely used open-source password cracking tool designed to uncover weak passwords and hash values. The tool employs various techniques including brute force and dictionary attacks to crack password hashes. John the Ripper can crack various types of password hashes including MD5, SHA-1, SHA-512 and SHA 256 as well as many others.

The commands I will be using for each hash are: -

Hash1.txt

john --format=raw-md5 –wordlist=wordlists.txt

Hash2.txt

john --format=raw-sha1 --wordlist=wordlist.txt hash2.txt

Hash3.txt

john --format=raw-sha512 --wordlist=wordlist.txt hash3.txt

Hash4.txt

john --format=raw-sha256 --wordlist=wordlist.txt hash4.txt

Hash5.txt

john --format=raw-md5 --wordlist=wordlist.txt hash5.txt

This was slightly challenging as john the ripper would not work on the ctf terminal so I ended up having to use an SCP command to securely transfer the wordlist.txt file onto the root terminal and then I was able to run john and find the following flags.

Hash1.txt – C0de_0b5cur3r_Flag

Hash2.txt – C0d3_5l4y3r_Flag

Hash3.txt - H4ck3r-Flag

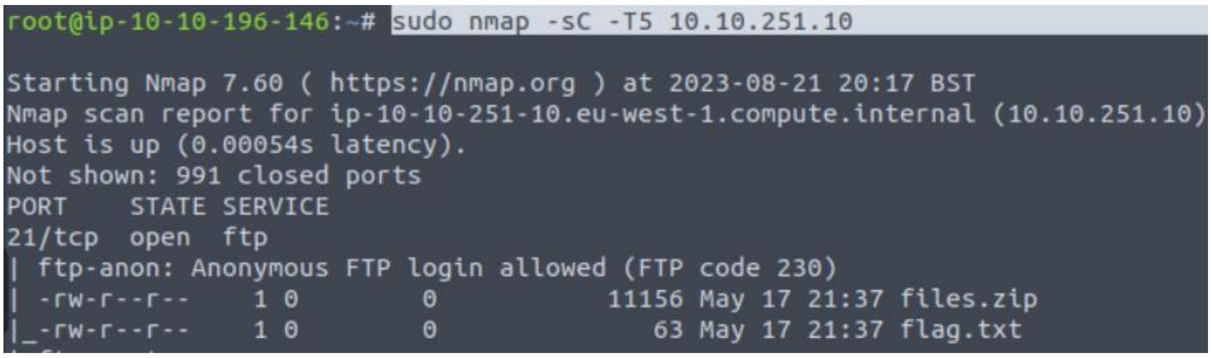
Hash4.txt – L0ck_Flag

Hash5.txt –S3cur1ty_Flag

NMAP SCAN REPORT

Nmap is an open-source powerful network scanning tool used to discover hosts, services and vulnerabilities on computer networks. It provides detailed information about networked devices, open ports and services they are running.

For this task I ran an Nmap scan using the command sudo nmap -sC -T5 10.10.251.10. From this scan as you can see in the diagram below there is an FTP port open with the files file.zip and flag.txt available.

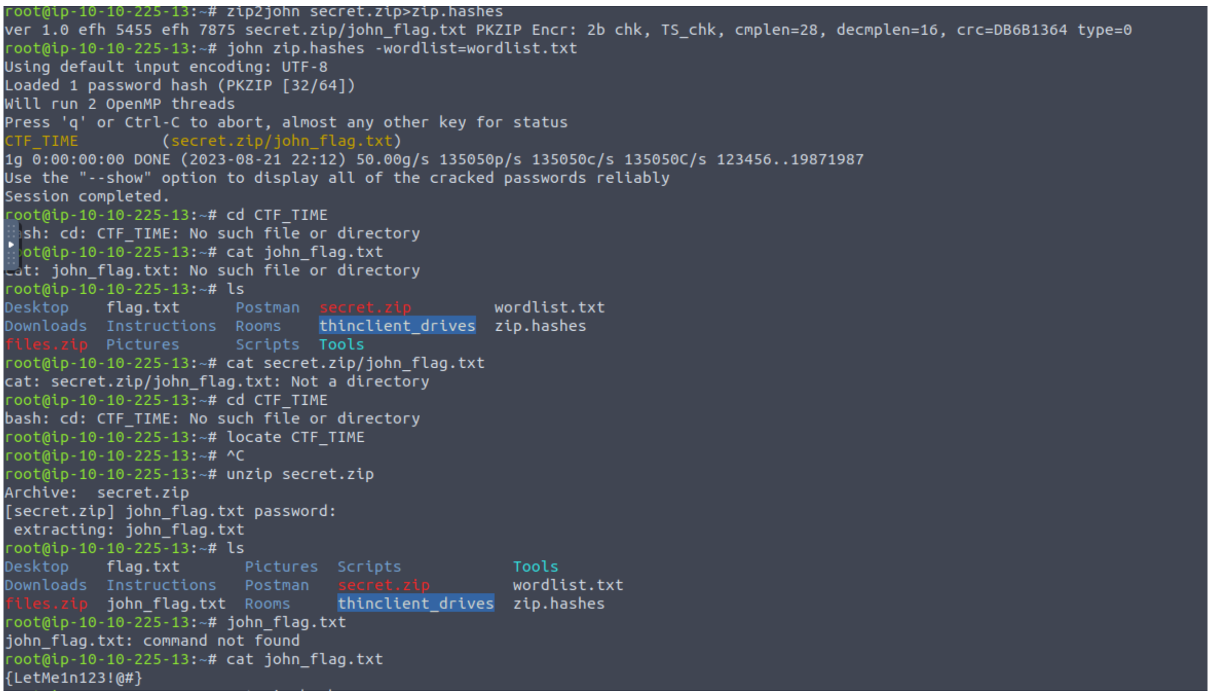


```
root@ip-10-10-196-146:~# sudo nmap -sC -T5 10.10.251.10

Starting Nmap 7.60 ( https://nmap.org ) at 2023-08-21 20:17 BST
Nmap scan report for ip-10-10-251-10.eu-west-1.compute.internal (10.10.251.10)
Host is up (0.00054s latency).
Not shown: 991 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
| 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
```

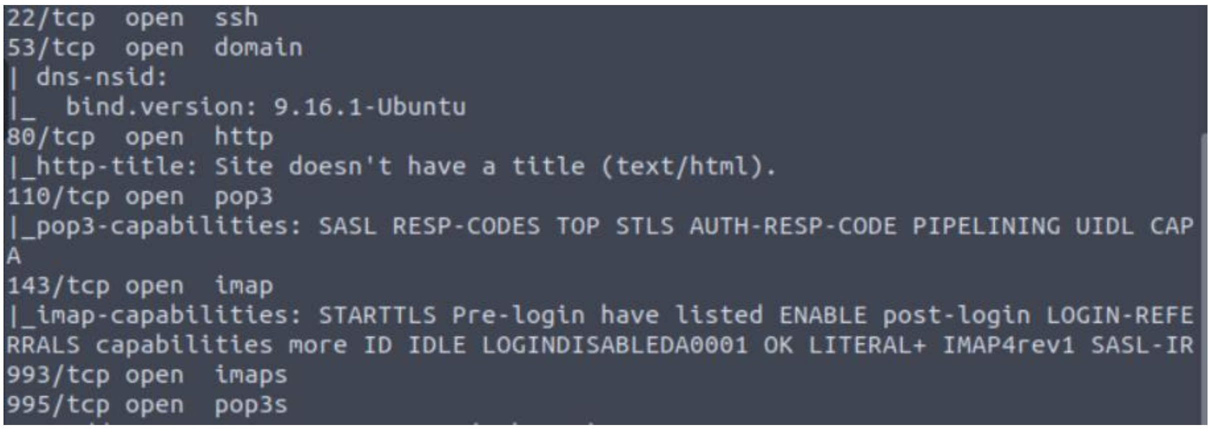
By simply running the command FTP 10.10.251.10 I am able to gain access to the system and download these two files back onto my root account.

The file flag.txt released the flag {ftp_server_4_lyfe}. After successfully unzipping the file files.zip and eventually guessing the password “Masterschool” I was able to access a second file named secret.zip. By running the command zip2john secret.zip > zip.hashes I was then able to crack the password and retrieve the flag {LetMe1n123!@#}.



```
root@ip-10-10-225-13:~# zip2john secret.zip>zip.hashes
ver 1.0 efh 5455 efh 7875 secret.zip/john_flag.txt PKZIP Encr: 2b chk, TS_chk, cmplen=28, decmplen=16, crc=DB6B1364 type=0
root@ip-10-10-225-13:~# john zip.hashes -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         (secret.zip/john_flag.txt)
1g 0:00:00:00 DONE (2023-08-21 22:12) 50.00g/s 135050p/s 135050c/s 135050C/s 123456..19871987
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
root@ip-10-10-225-13:~# cd CTF_TIME
 sh: cd: CTF_TIME: No such file or directory
 ot@ip-10-10-225-13:~# cat john_flag.txt
 at: john_flag.txt: No such file or directory
root@ip-10-10-225-13:~# ls
Desktop    flag.txt      Postman  secret.zip        wordlist.txt
Downloads  Instructions  Rooms    thinclient_drives  zip.hashes
files.zip  Pictures      Scripts  Tools
root@ip-10-10-225-13:~# cat secret.zip/john_flag.txt
cat: secret.zip/john_flag.txt: Not a directory
root@ip-10-10-225-13:~# cd CTF_TIME
bash: cd: CTF_TIME: No such file or directory
root@ip-10-10-225-13:~# locate CTF_TIME
root@ip-10-10-225-13:~# ^C
root@ip-10-10-225-13:~# unzip secret.zip
Archive:  secret.zip
[secret.zip] john_flag.txt password:
 extracting: john_flag.txt
root@ip-10-10-225-13:~# ls
Desktop    flag.txt       Pictures  Scripts            Tools
Downloads  Instructions   Postman   secret.zip         wordlist.txt
files.zip  john_flag.txt  Rooms     thinclient_drives  zip.hashes
root@ip-10-10-225-13:~# john_flag.txt
john_flag.txt: command not found
root@ip-10-10-225-13:~# cat john_flag.txt
{LetMe1n123!@#}
```

The Nmap scan also showed that Ports 22,53,80, 110,143, 993 and 995 were all open as shown in the diagram below.



```
22/tcp  open  ssh
53/tcp  open  domain
| dns-nsid:
|_  bind.version: 9.16.1-Ubuntu
80/tcp  open  http
|_http-title: Site doesn't have a title (text/html).
110/tcp open  pop3
|_pop3-capabilities: SASL RESP-CODES TOP STLS AUTH-RESP-CODE PIPELINING UIDL CAP
A
143/tcp open  imap
|_imap-capabilities: STARTTLS Pre-login have listed ENABLE post-login LOGIN-REFE
RRALS capabilities more ID IDLE LOGINDISABLEDA0001 OK LITERAL+ IMAP4rev1 SASL-IR
993/tcp open  imaps
995/tcp open  pop3s
```

Port 22 - (SSH) secure shell.

It is used for ensuring secure access to servers however it is possible to attack by using leaked SSH keys or by running a brute force attacks. Ensuring that the SSH service is running the latest version, using strong passwords or key-based authentication to secure SSH access will mitigate potential vulnerabilities.

Port 53 - Domain Name System (DNS).

It is runs on UDP and TCP and is used for queries and transfers. This port is particularly vulnerable to DDoS attacks. By Implementing proper DNS security measures, such as limiting zone transfers and securing against DNS cache poisoning attacks will mitigate the risk against a DDoS attack happening.

Port 80 (HTTP)

This port provides a HTTP connection under the TCP protocol. It is an unencrypted connection between the web browser and the web servers, which leaves the sensitive user data exposed to Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and SQL Injection. By ensuring that the web server software is up-to-date and patching against any known vulnerabilities will increase the likelihood of staying safe.

Port 110 and 995 - POP3

POP3 is an email retrieval protocol used for fetching emails from a mail server. Port 110 uses plaintext and 995 uses encrypted. POP3 is an older protocol and therefore more susceptible to attacks. Ensure that strong email account passwords are used but if possible consider migrating to a more secure email protocols if feasible.

Port 143 and 993/TCP (IMAP)

This is another email retrieval protocol although port 143 is used for plaintext and 993 for encrypted they are slightly more advanced than POP3 but would still be vulnerable to attacks. Ensure strong email account passwords are used. Implement SSL/TLS encryption for secure communication and consider using more modern email protocols for improved security.

Recommendations:

1. Regularly update and patch all software and services to ensure they are protected against known vulnerabilities.
2. Implement strong authentication mechanisms, such as key-based authentication for SSH.
3. Configure the DNS server (BIND) to minimize the risk of zone transfers and cache poisoning attacks.
4. Implement security best practices for web applications, including input validation and output encoding.
5. Consider just using the more modern email protocols of IMAPS and POP3S, which provide encrypted communication.

Conclusion: The Nmap scan identified several open ports and services on the target system. While specific vulnerabilities were not determined from the scan alone, further research and testing are required to identify potential vulnerabilities and their corresponding mitigation techniques. It is essential to maintain a proactive approach to security by keeping software up to date, applying security patches, and following best practices for securing each service.

FLAGS

1. {h4ck3r5_r_us}
2. {St0ry_Fl4g}
3. (Y0u_G0T_1t}
4. {H1d3_1n_pl41n_s1gh7}
5. {STUDENT_CTF_Web}
6. {C0nf1gur4t10n_Fl4g}
7. {H1d3_Fl4g}
8. {Robots_Flag}
9. {S3cr3t_Fl4g}
10. {Fl4g_fl4g_fl4g}
11. C0de_0b5cur3r_Flag
12. C0d3_5l4y3r_Flag
13. H4ck3r-Flag
14. L0ck_Flag
15. S3cur1ty_Flag
16. {ftp_server_4_lyfe}
17. {LetMe1n123!@#}