FIREWALL DRAGON

Try Hack Me

CCT2019

11 / 26 / 2023

FIREWALL DRAGON

Description

“Legacy challenges from the US Navy Cyber competition Team 2019 Assessment sponsored by US TENTH Fleet”

Difficulty: Insane

Link: https://tryhackme.com/room/cct2019

What does this report contain?

This report exclusively contains the complete procedure needed to obtain the six flags of this room. It does not include the usual steps of the penetration testing process (reconnaissance, scanning, etc.).

Task 1: CCT2019 – pcap1

Description

This is a pcap-focused challenge originally created for the U.S. Navy Cyber Competition Team 2019 Assessment. Although the assessment is over, the created challenges are provided for community consumption here.

If you find the right clues, they will guide you to the next step. I did include some red herrings in this challenge, but you can stay on track by focusing on pcap-related skills.

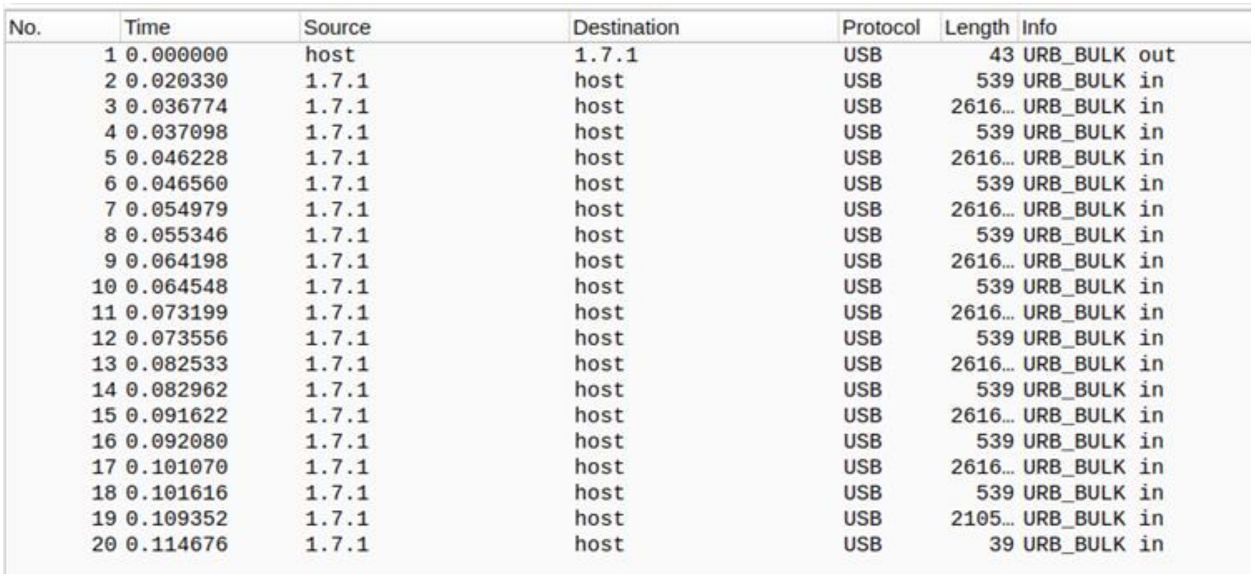
HINT1: It's a pcap challenge. If you're doing stego or re, you're either down a rabbit hole or there's an easier way.

HINT2: It is very important to do the first step correctly. If you don't recover the first file in its entirety, you may not be able to complete steps later on in the challenge. The second pcap file has 4588 packets in it. Contact me on Discord (send a DM to username zoobah) if you are struggling with this step.

HINT3: On the final step, this was built to run in a amd64 Kali Linux environment. If you are using a different Linux distro, you may run into some problems.

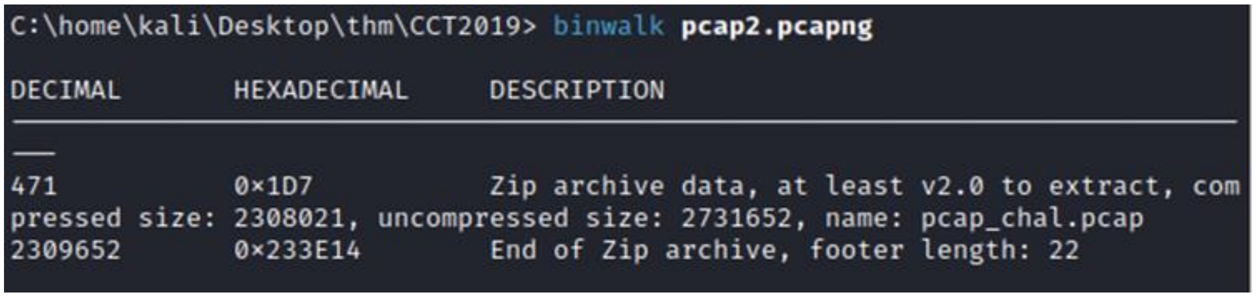
Completion

An initial analysis shows that few USB packages are contained inside.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	host	1.7.1	USB	43	URB_BULK out
2	0.020330	1.7.1	host	USB	539	URB_BULK in
3	0.036774	1.7.1	host	USB	2616…	URB_BULK in
4	0.037098	1.7.1	host	USB	539	URB_BULK in
5	0.046228	1.7.1	host	USB	2616…	URB_BULK in
6	0.046560	1.7.1	host	USB	539	URB_BULK in
7	0.054979	1.7.1	host	USB	2616…	URB_BULK in
8	0.055346	1.7.1	host	USB	539	URB_BULK in
9	0.064198	1.7.1	host	USB	2616…	URB_BULK in
10	0.064548	1.7.1	host	USB	539	URB_BULK in
11	0.073199	1.7.1	host	USB	2616…	URB_BULK in
12	0.073556	1.7.1	host	USB	539	URB_BULK in
13	0.082533	1.7.1	host	USB	2616…	URB_BULK in
14	0.082962	1.7.1	host	USB	539	URB_BULK in
15	0.091622	1.7.1	host	USB	2616…	URB_BULK in
16	0.092080	1.7.1	host	USB	539	URB_BULK in
17	0.101070	1.7.1	host	USB	2616…	URB_BULK in
18	0.101616	1.7.1	host	USB	539	URB_BULK in
19	0.109352	1.7.1	host	USB	2105…	URB_BULK in
20	0.114676	1.7.1	host	USB	39	URB_BULK in

Running the binwalk command reveals that it contains a second compressed file named "pcap_chal.pcapng": binwalk pcap2.pcapng

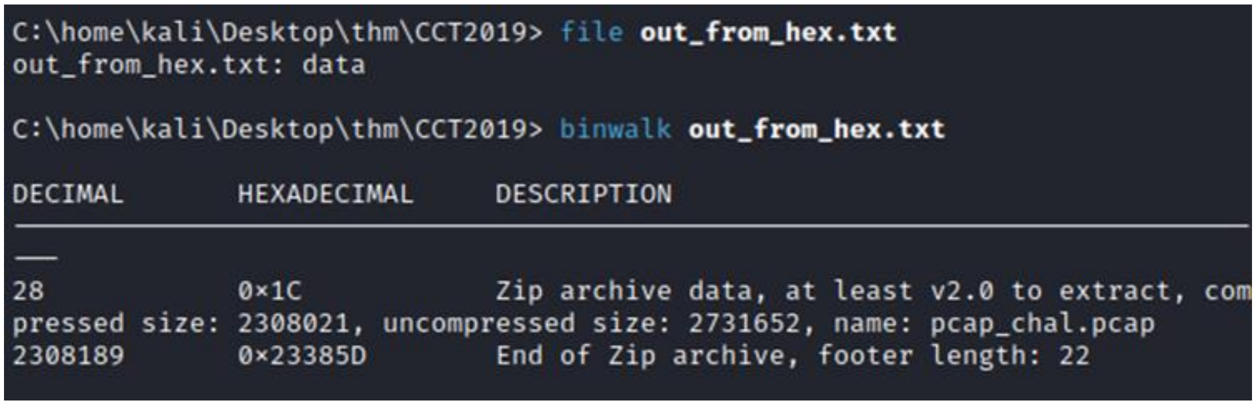


```
C:\home\kali\Desktop\thm\CCT2019> binwalk pcap2.pcapng

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
--
471           0x1D7           Zip archive data, at least v2.0 to extract, com
pressed size: 2308021, uncompressed size: 2731652, name: pcap_chal.pcap
2309652       0x233E14        End of Zip archive, footer length: 22
```

Initially, an attempt was made to extract the contents directly, but this approach encountered various issues. After conducting research, a solution was found by extracting the USB-exchanged contents using "tshark" via CMD with the following command: `tshark -r pcap2.pcapng -T fields -e usb.capdata > out`

This command enabled the extraction of hexadecimal data from the transfer. However, in this format, reading the contents accurately was not feasible.

Ultimately, the file was successfully converted into a usable form using the "From Hex" function in CyberChef.



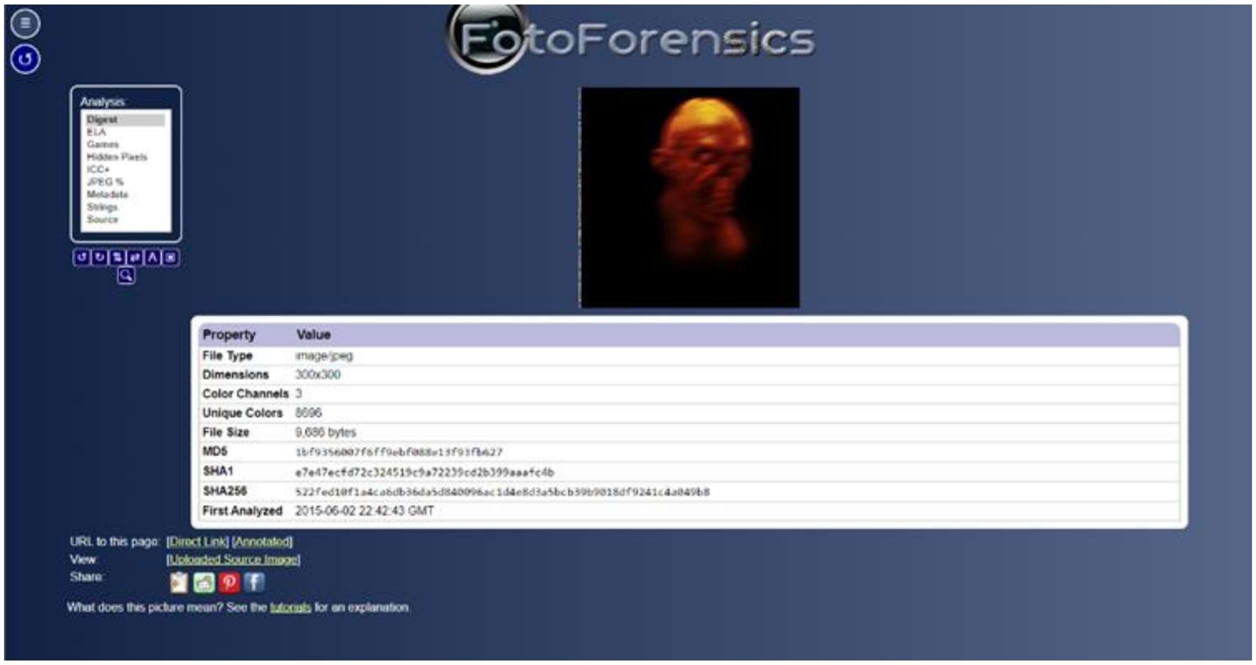
```
C:\home\kali\Desktop\thm\CCT2019> file out_from_hex.txt
out_from_hex.txt: data

C:\home\kali\Desktop\thm\CCT2019> binwalk out_from_hex.txt

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
--
28            0x1C            Zip archive data, at least v2.0 to extract, com
pressed size: 2308021, uncompressed size: 2731652, name: pcap_chal.pcap
2308189       0x23385D        End of Zip archive, footer length: 22
```

Now, let's proceed to analyze the traffic within the "pcap_chal.pcapng" file. This file encompasses various types of traffic, and in such situations, I prefer commencing the analysis with HTTP/HTTPS, as it often provides valuable insights into the ongoing activities.

Initially, my attention was drawn to the following GET request:

‘fotoforensics.com/analysis.php?id=e7e47ecfd72c324519c9a72239cd2b399aaafc4b.9686&fmt=card’

By removing the last parameter (`&fmt=card`), we can trace back to the original analysis of the image and subsequently download it.



Property	Value
File Type	image/jpeg
Dimensions	300x300
Color Channels	3
Unique Colors	8696
File Size	9,686 bytes
MD5	1bf9356007f6ff9ebf088e13f93fb627
SHA1	e7e47ecfd72c324519c9a72239cd2b399aaafc4b
SHA256	522fed10f1a4ca6db36da5d840096ac1d4e8d3a5bcb39b9018df9241c4a049b8
First Analyzed	2015-06-02 22:42:43 GMT

URL to this page: [Direct Link] [Annotated]
View: [Uploaded Source Image]
Share:
What does this picture mean? See the tutorials for an explanation.

Once the image is downloaded, via binwalk, one can see that it contains a RAR archive inside.
Analysis conducted on the image unfortunately was a washout as it turns out to be a rabbit hole.

By then delving deeper into the traffic, IRC connections could be identified from which it was possible to extract the following credentials:

USERNAME: binaryphalanx (nick: zoobah)

PASSWORD: Red********Rover$$

Uniq ID: 108AAAAAC

Still having no useful detections to work on, the rest of the traffic was delved into, particularly ICMP as it reports anomalous packets:
Delving then into the unanswered ICMP traffic, it is possible to find a conversation between two attackers embedded in the network, in which they exchange a chat in which they talk about which password to use. From the messages, it is possible to identify the following:

FIREWALL DRAGON

1) Hackers passed around a file on port 4444

2) The file was encrypted using cryptcat

3) The password is the one used by Angela Bennett in the movie “the net” (BER5348833)

From the pcap file we then extract the traffic on port 4444, retrieving the data as RAW following the traffic flow and saving the data.

Now we have to decipher them!

To do this, we will use cryptcat to listen and decrypt the file and netcat to relay it:

- Server: cryptcat -vv -k BER5348833 -l -p 4444 > decrypt
- Client: cat file_crypted > nc 127.0.0.1 4444

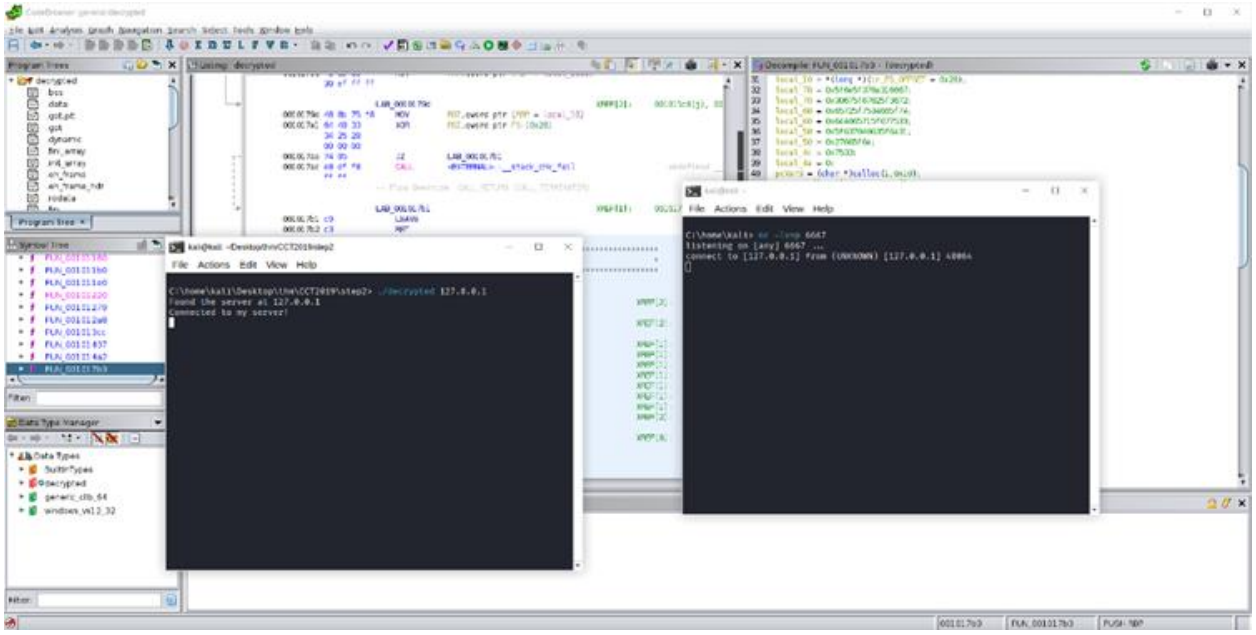
NOW the file can be analyzed, by doing some reverse engineering we can see that it tries to connect to a server, as it performs a DNS resolution for the domain “irc.cct”

By inserting the entry “127.0.0.1 irc.cct” in the “/etc/hosts” file and running the nc command listening on port 6667 (default IRC port) with the following command.

- CMD: nc -lvnp 6667

It will be possible to receive the connections made by the file, and analyze the requests/traffic generated.

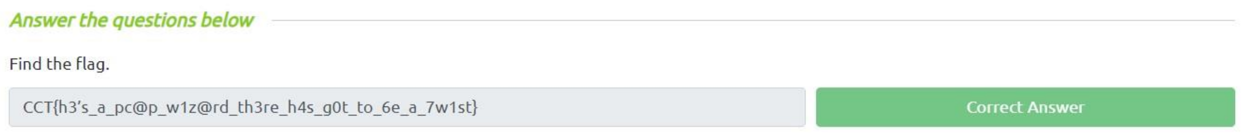
Now running the file will generate traffic:



```
C:\home\kali\Desktop\thm\CCT2019\step2> ./decrypted 127.0.0.1
Found the server at 127.0.0.1
Connected to my server!
```

```
C:\home\kali> nc -lvnp 6667
listening on [any] 6667 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 40866
```

And finally, we manage to get the FLAG!



Answer the questions below

Find the flag.

CCT{h3's_a_pc@p_w1z@rd_th3re_h4s_g0t_to_6e_a_7w1st}	Correct Answer

Task 2 CCT2019 – re3

Description

There's some kind of a high security lock blocking the way. Defeat the GUI to claim your key!

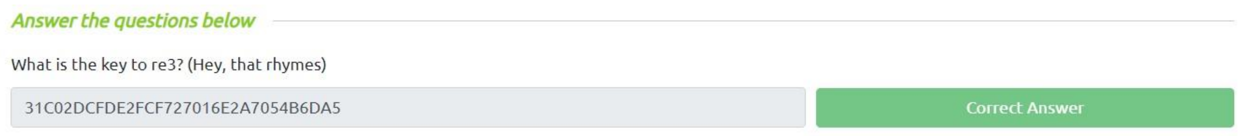
NOTE: The key is a 32-character hex blob and doesn't follow the CCT{.*} format. It'll be apparent when you've found it.

If you need a Windows machine to help reverse engineering this, please use the Windows base room.

Completion

Analyzing the code written in .NET identifies an IF statement whose solution allows the flag to be extracted; the operation requires that the sum of four numbers has to be equal to 711 and their product must be 7110000.

A mathematical resolution dictated by two equations (one addition and one product) with four unknowns was used, also using common factors to identify the span of values that could be used.


Answer the questions below

What is the key to re3? (Hey, that rhymes)

31C02DCFDE2FCF727016E2A7054B6DA5

Task 3 CCT2019 – for1

Description

UPDATE: There was a bug found in cryptii that has now been fixed, but will cause issues on the final step of the challenge. For now, when you find the the cipher text FSXL PXTH EKYT DJXS PYMO JLAY VPRP VO, replace it with this cipher text instead: JHSL PGLW YSQO DQVL PFAO TPCY KPUD TF. Everything else at that step, e.g., the configuration file can remain as-is. I intend to update the challenge file to correct this issue, but this will serve as a temporary fix until that time.

Our former employee Ed is suspected of suspicious activity. We found this image on his work desktop and we believe it is something worth analyzing. Can you assist us in extracting any information of value?

HINT1: if you're not sure if a password is upper- or lower-case, try all lower-case.

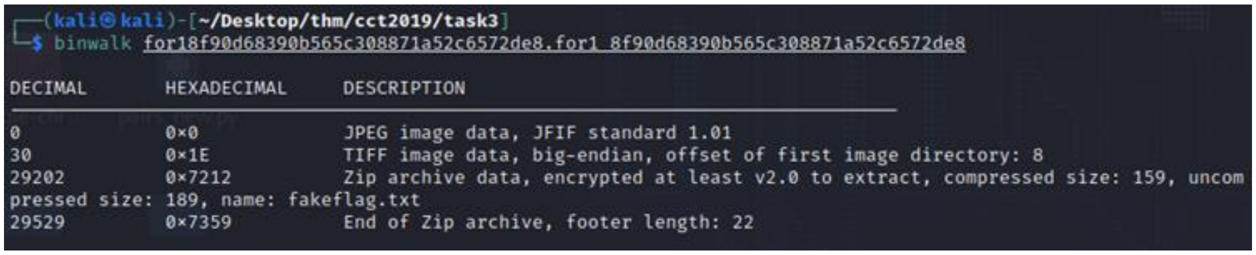
HINT2: There are many steps that can be done concurrently in this challenge. If you find you need something, you may have not found the key to unlock it yet. If you have something useful and you're not sure where to use it, it's possible the file you need is still hidden somewhere.

HINT3: https://cryptii.com/ - Cool website, bro

HINT4: the flag will follow the format CCT{.*}

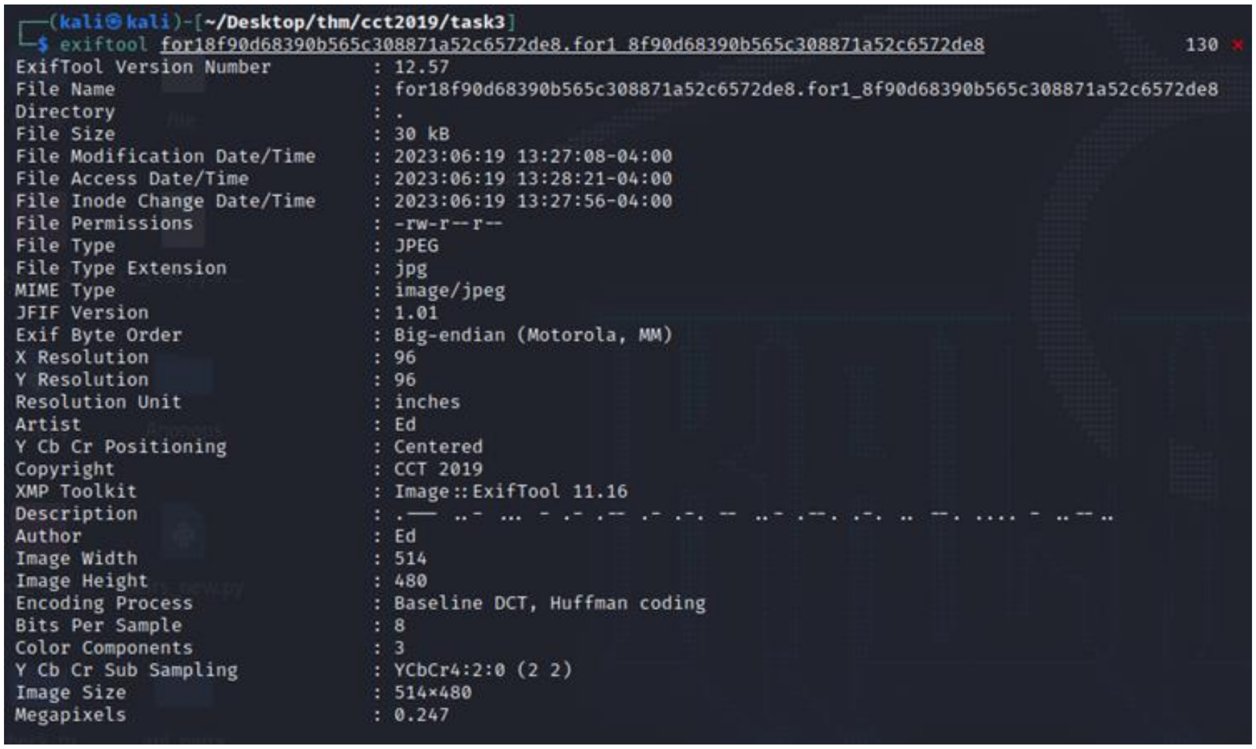
Completion

FIREWALL DRAGON



```
┌──(kali㉿kali)-[~/Desktop/thm/cct2019/task3]
└─$ binwalk for18f90d68390b565c308871a52c6572de8.for1_8f90d68390b565c308871a52c6572de8

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
30            0x1E            TIFF image data, big-endian, offset of first image directory: 8
29202         0x7212          Zip archive data, encrypted at least v2.0 to extract, compressed size: 159, uncom
pressed size: 189, name: fakeflag.txt
29529         0x7359          End of Zip archive, footer length: 22
```

By analyzing photos with exitfool we can find a morse code in the description section:



```
┌──(kali㉿kali)-[~/Desktop/thm/cct2019/task3]
└─$ exiftool for18f90d68390b565c308871a52c6572de8.for1_8f90d68390b565c308871a52c6572de8            130 ✗
ExifTool Version Number         : 12.57
File Name                       : for18f90d68390b565c308871a52c6572de8.for1_8f90d68390b565c308871a52c6572de8
Directory                       : .
File Size                       : 30 kB
File Modification Date/Time     : 2023:06:19 13:27:08-04:00
File Access Date/Time           : 2023:06:19 13:28:21-04:00
File Inode Change Date/Time     : 2023:06:19 13:27:56-04:00
File Permissions                : -rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Exif Byte Order                 : Big-endian (Motorola, MM)
X Resolution                    : 96
Y Resolution                    : 96
Resolution Unit                 : inches
Artist                          : Ed
Y Cb Cr Positioning             : Centered
Copyright                       : CCT 2019
XMP Toolkit                     : Image::ExifTool 11.16
Description                     : .--- ..- ... - .- .-- .- .-. -- ..- .--. .-. .. --. .... - ..--..
Author                          : Ed
Image Width                     : 514
Image Height                    : 480
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 514x480
Megapixels                      : 0.247
```

Which result to be the string: “jus********right?”

I tried to use it as a password for the previous ZIP found with binwalk and it worked!

From the extraction i got the file “fakeflag.txt” which contains the following text:

- I didn’t say it would be easy, Neo. Peer into the Matrix. See what others cannot and witness the truth. Though I caution that it may be more than what you expect.

- Morpheus

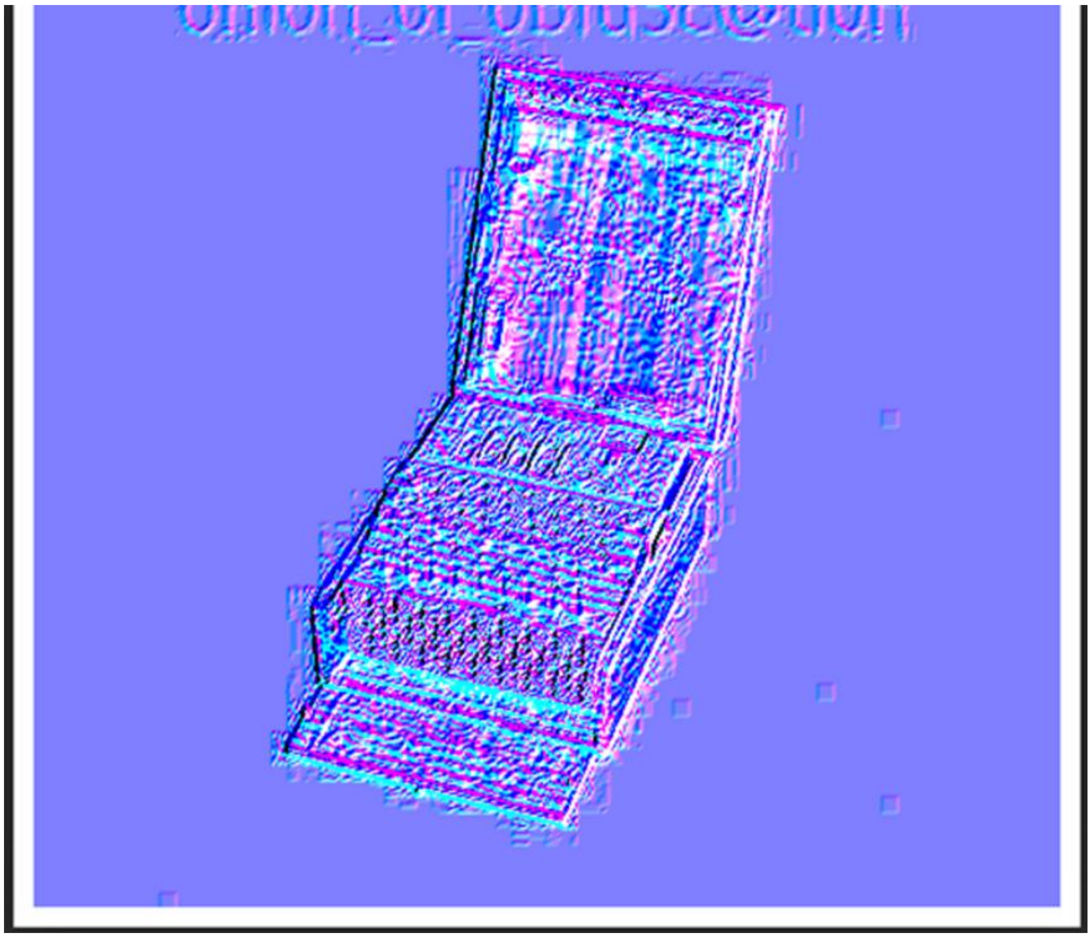
- PW: Z10N****

Now we have a new password!

In addition, through the tool “steghide” I was able to extract an archive “archive.zipper,” which contains two files (cipher.txt and config.txt) and the flag.zipper.

To extract it, you can use the password just found in the file “fakeflag.txt” via the cmd “steghide -extract -sf file_original_CTF”

Deepening the image analysis, thanks to the luminescence gradient we obtain:



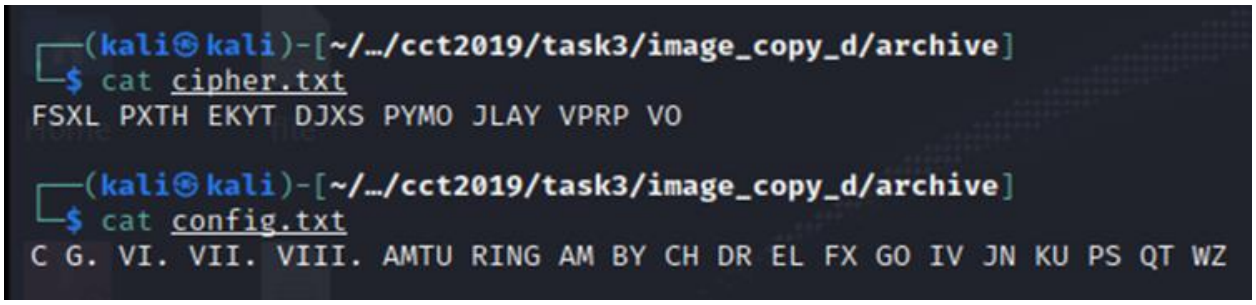
And from the image we can derive a plausible password:

- 0ni********fu5c@ti0n

We can use this password to extract the zipper file found earlier ("archive zipper") which will give us three files:

- Cipher.txt
- Config.txt
- Flag.zipper

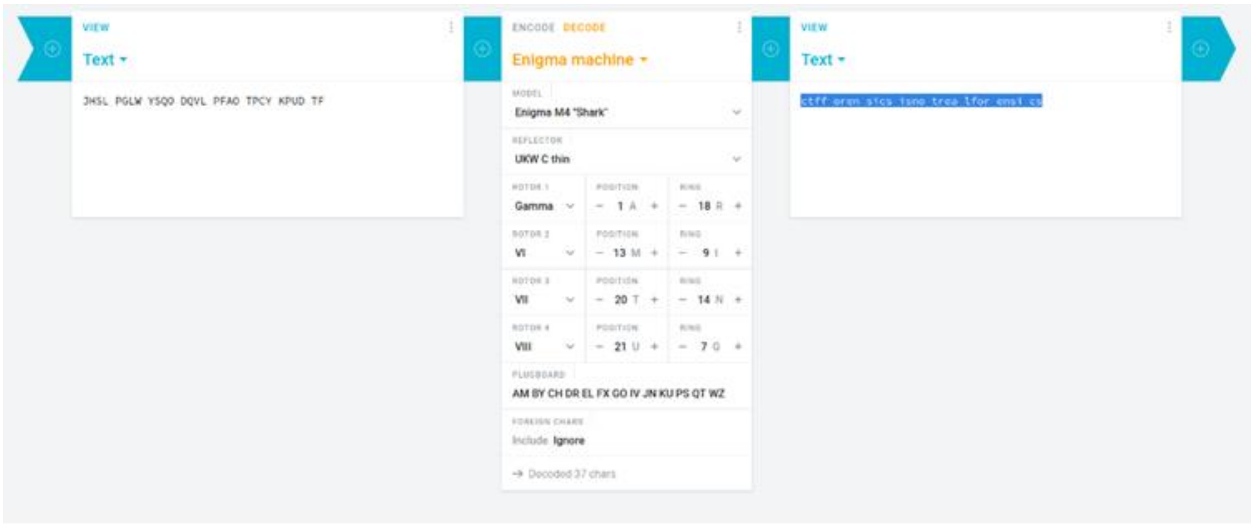
Below are the contents of the two txt files:



```
┌──(kali㉿kali)-[~/…/cct2019/task3/image_copy_d/archive]
└─$ cat cipher.txt
FSXL PXTH EKYT DJXS PYMO JLAY VPRP VO

┌──(kali㉿kali)-[~/…/cct2019/task3/image_copy_d/archive]
└─$ cat config.txt
C G. VI. VII. VIII. AMTU RING AM BY CH DR EL FX GO IV JN KU PS QT WZ
```

The former of which contain the text to be deciphered and the latter the configurations for the "enigma" machine.

By going to the site recommended by CTf (https://cryptii.com/) and doing some "truble shooting" (based on the data I have and the different options on the site) I was eventually able to find an enigma type (Enigma M4 “shark”) that allowed me to enter all the data obtained from the config.txt:

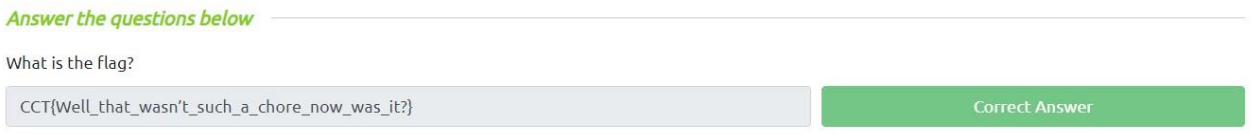


```
JHSL PGLW YSQO DQVL PFAO TPCY KPUD TF
```

MODEL: Enigma M4 "Shark"
REFLECTOR: UKW C thin
ROTOR 1: Gamma — POSITION: 1 A — RING: 18 R
ROTOR 2: VI — POSITION: 13 M — RING: 9 I
ROTOR 3: VII — POSITION: 20 T — RING: 14 N
ROTOR 4: VIII — POSITION: 21 U — RING: 7 G
PLUGBOARD: AM BY CH DR EL FX GO IV JN KU PS QT WZ
FOREIGN CHARS: Include Ignore
→ Decoded 37 chars

```
ctff orgn sics isno trea 1for ensi cs
```

I then eventually got the password for the flag.zipper file



What is the flag?

CCT{Well_that_wasn’t_such_a_chore_now_was_it?}

Task 4 CCT2019 – crypto1

Description

Find ye some flags. There are three parts to this challenge, each with its own flag. Solve crypto1a obtain the crypto1a flag and to unlock crypto1b. Solve crypto1b to obtain the crypto1b flag and unlock crypto1c. Solve crypto1c and you'll have all three flags.

HINT1: crypto1a and crypto1b can be solved with freely available online tools

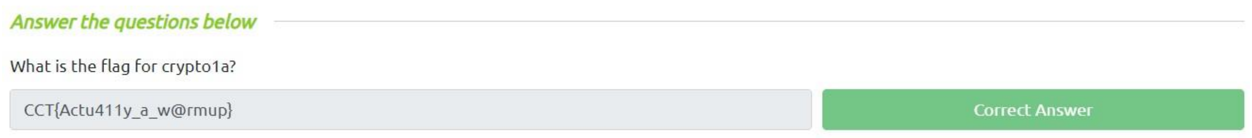
HINT2: For crypto1c, you probably have to code a solution to solve it as I'm not aware of any online tools for this variant. It's not complex to solve if you can figure out the scheme and it is possible to solve by hand although it could be a bit tedious.

HINT3: For crypto1c, start with "0" not "1".

Completion

The last task is divided into three consequential parts, and in each of these there is a first file containing a clue and a second with text to be deciphered:

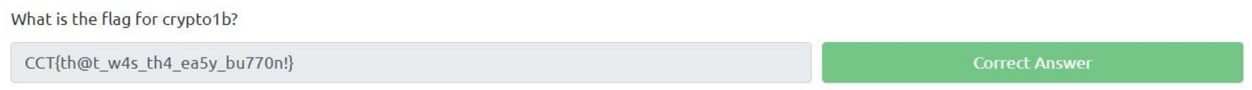
1.The first step requires conversion between keyboard layouts (Keybord layout substitution)



What is the flag for crypto1a?

CCT{Actu411y_a_w@rmup}

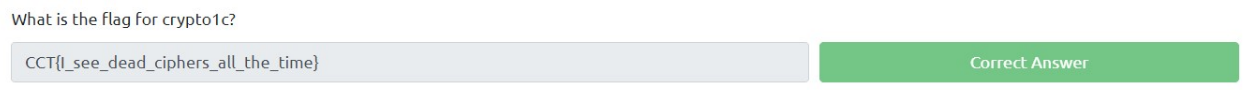
2. For this particular file, an "OSINT" (Open-Source Intelligence) investigation on YouTube is required. The objective is to search for a specific video that contains information enabling the identification of the password necessary for decrypting the file contents, which have been encrypted using the railfence cipher.



What is the flag for crypto1b?

CCT{th@t_w4s_th4_ea5y_bu770n!}

3. For this final section, Python code needs to be written. In this code, each number in the encoded file (where all characters are within the range of 0 to 6) will be converted. By applying the "%" operator to calculate the remainder, the number sequence can be transformed into binary. Once the binary sequence is converted to ASCII, the final text, and consequently the FLAG, will be obtained.



What is the flag for crypto1c?

CCT{I_see_dead_ciphers_all_the_time}