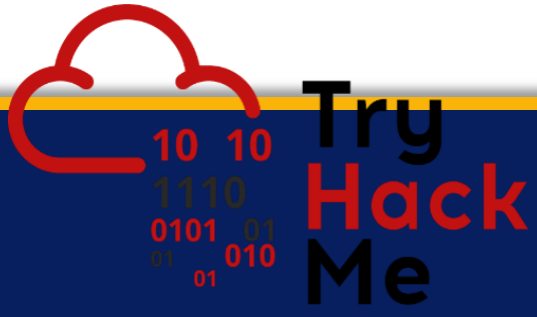


Catch The Flag

CTF Collection Vol.1 Walkthrough

Original Author: *Dheeraj Gupta*

Table of Contents

Abstract

This report will provide a walkthrough of another popular TryHackMe Catch The Flag (CTF) challenge. The main objective of the challenge is to test your CTF skills, the challenge consists of 20 tasks that require you to stay calm and capture the flag.

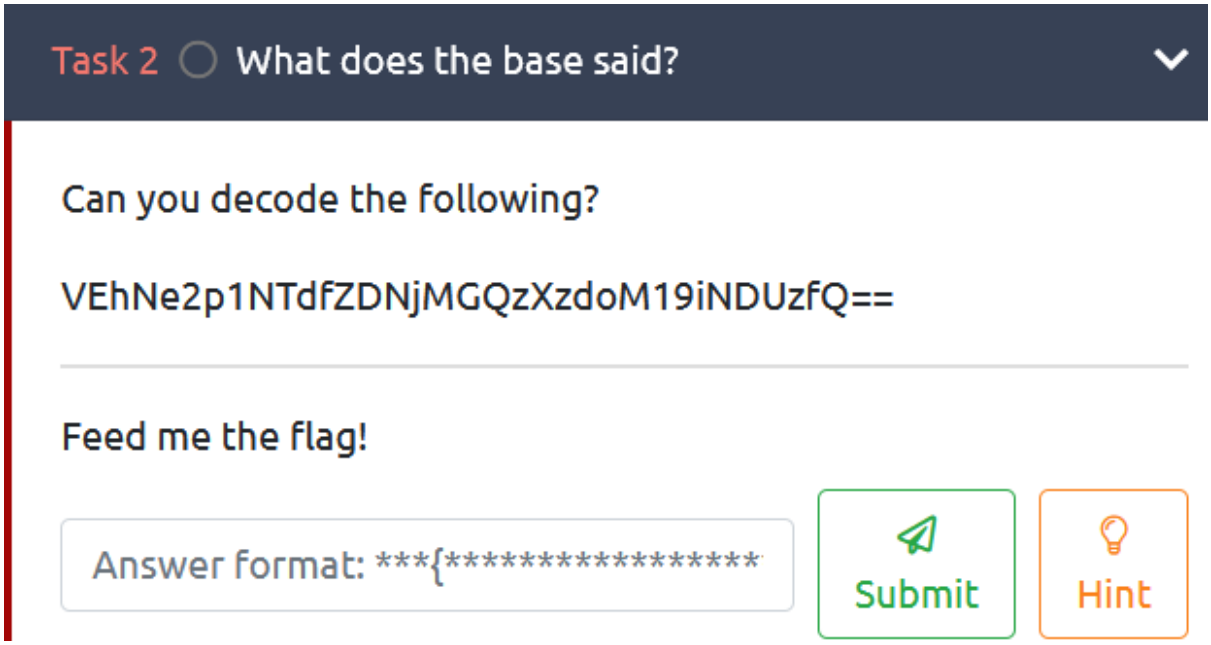
This CTF challenge called "CTF collection Vol.1" is available at TryHackMe for penetration testing practice. This lab is not difficult if we have the right basic knowledge of cryptography and steganography.

Disclaimer: This report is provided for educational and informational purpose only (Penetration Testing). Penetration Testing refers to legal intrusion tests that aim to identify vulnerabilities and improve cybersecurity, rather than for malicious purposes.

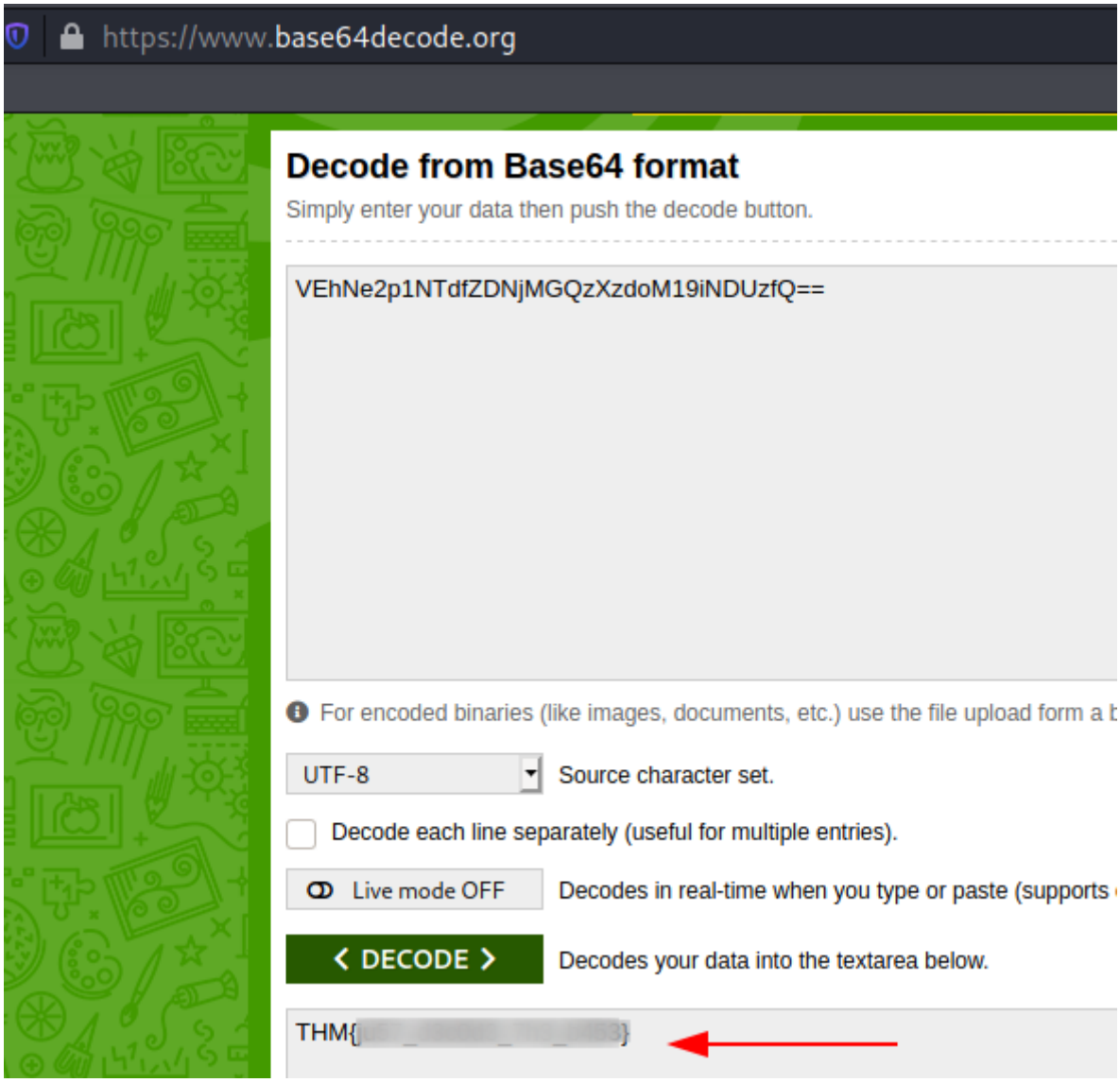
[TASK 1] Author note

This task just provides information about this CTF challenge.

[TASK 2] What does the base say?

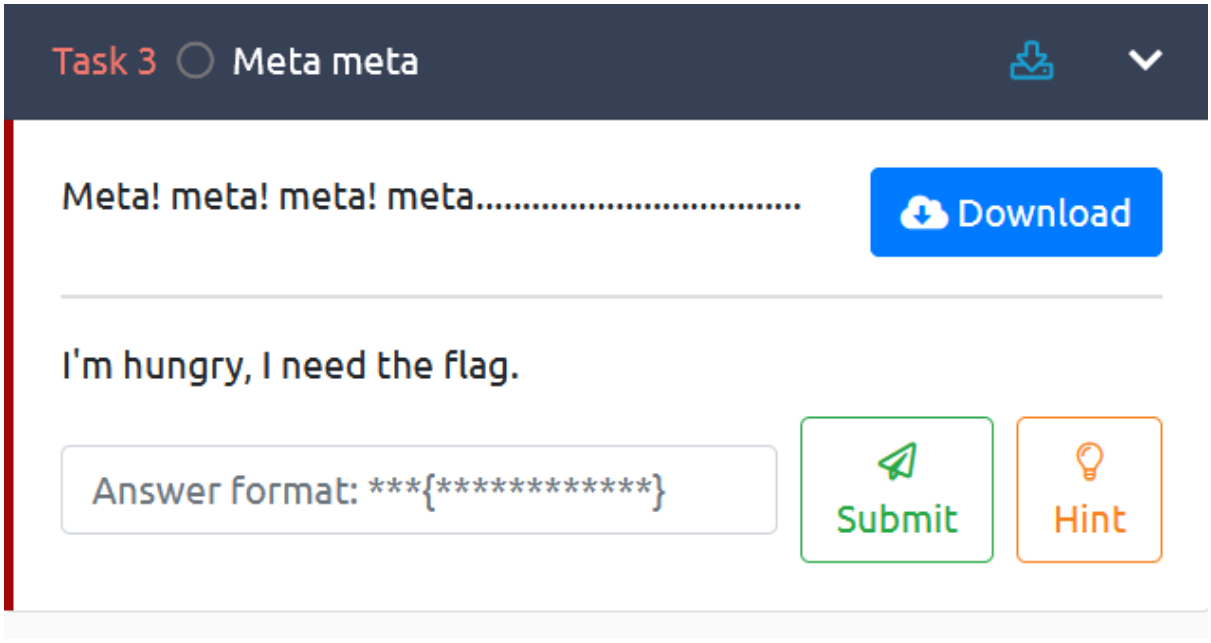


In this task, we can conveniently recognize that the value is base64 encoded. We will then attempt to decode it using the base64 decoder.



And there is how we got our 2nd flag easily.

[TASK 3] Meta meta



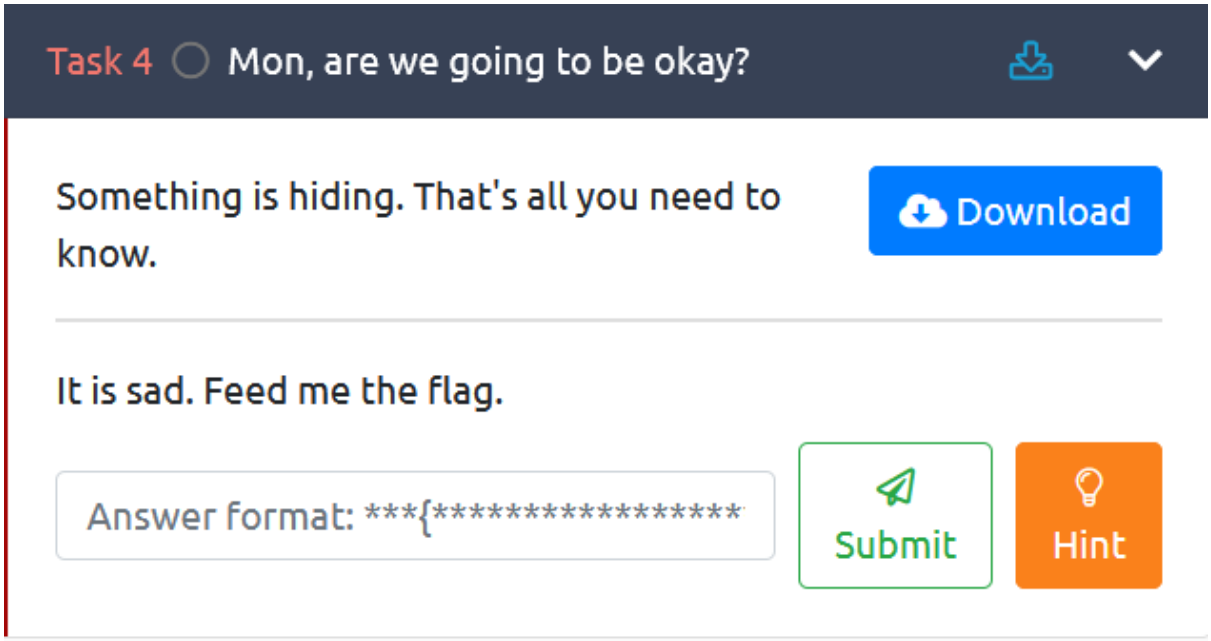
As the role name itself specifies the term meta, therefore, we are clear that here we have to take out the metadata of the image. Thereby we've used the EXIF tool as it is the most powerful one.

```
exiftool findme.jpg /root/Downloads/Findme.jpg
```

```
root@kali:~# exiftool findme.jpg /root/Downloads/Findme.jpg
Error: File not found - findme.jpg
======== /root/Downloads/Findme.jpg
ExifTool Version Number         : 11.94
File Name                       : Findme.jpg
Directory                       : /root/Downloads
File Size                       : 34 kB
File Modification Date/Time     : 2020:12:22 05:39:55-05:00
File Access Date/Time           : 2020:12:22 05:40:04-05:00
File Inode Change Date/Time     : 2020:12:22 05:39:58-05:00
File Permissions                : rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
X Resolution                    : 96
Y Resolution                    : 96
Exif Byte Order                 : Big-endian (Motorola, MM)
Resolution Unit                 : inches
Y Cb Cr Positioning             : Centered
Exif Version                    : 0231
Components Configuration        : Y, Cb, Cr, -
Flashpix Version                : 0100
Owner Name                      : THM{[redacted]}
Comment                         : CREATOR: gd-jpeg v1.0 (using IJG JPEG v62), quality = 60.
Image Width                     : 800
Image Height                    : 480
Encoding Process                : Progressive DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 800x480
Megapixels                      : 0.384
```

And as expected we get our 3rd flag in the metadata of the file stating in the parameter of owner name.

[TASK 4] Mon, are we going to be okay?



This task was based on steganography because in the task description they are indicating that "something is hiding" where it's gone fishy, moreover there is one file to download.

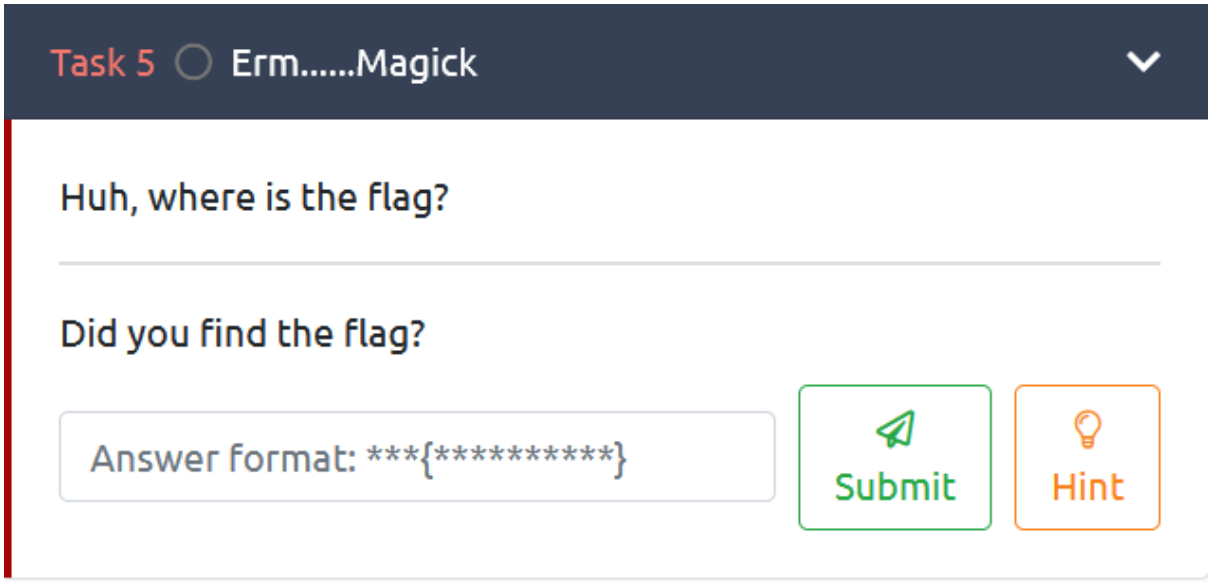
So, here we thought they are talking about steganography and decided to use steghide.

```
steghide extract -sf /root/Downloads/Extinction.jpg
```

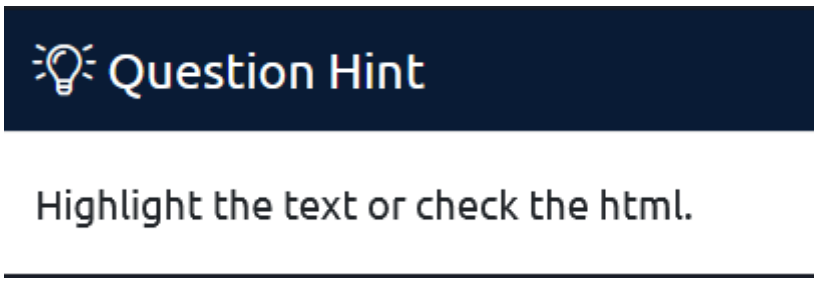
```
┌──(root💀kali)-[/home/kali/Downloads]
└─# steghide extract -sf /home/kali/Downloads/Extinction.jpg
Enter passphrase:
wrote extracted data to "Final_message.txt".

┌──(root💀kali)-[/home/kali/Downloads]
└─# cat Final_message.txt
It going to be over soon. Sleep my child.

THM{500n3r_0r_l473r_17_15_0ur_7urn}
```

But interestingly we got another file bind within it when we used to extract the data from the image. And here our arrow goes on point!! We got our 4th flag in this text file.

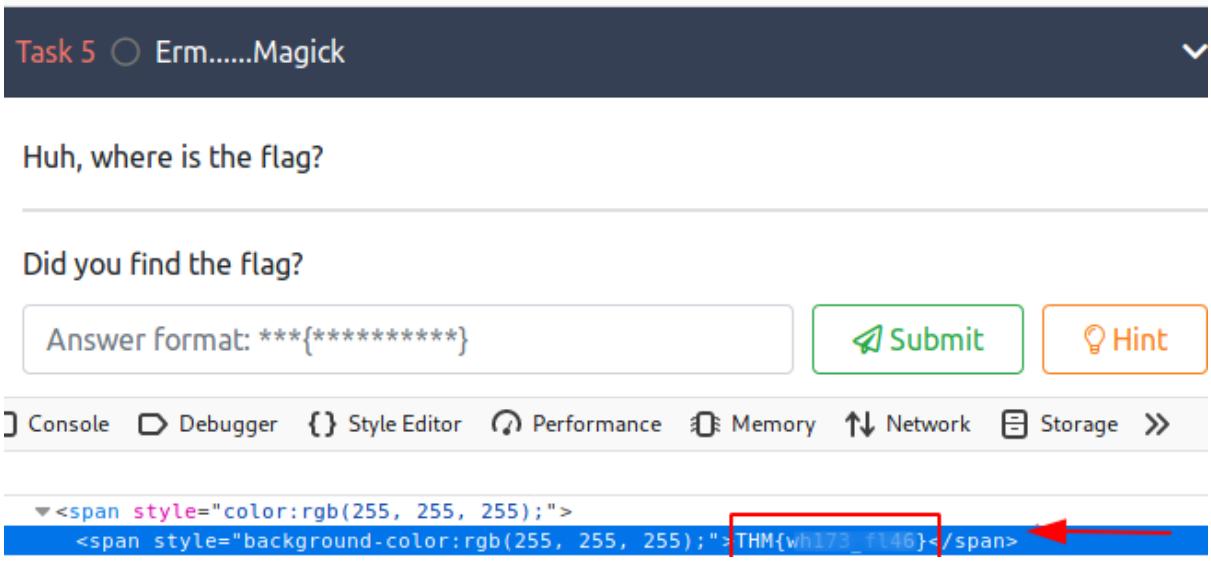
[TASK 5] Erm.....Magick



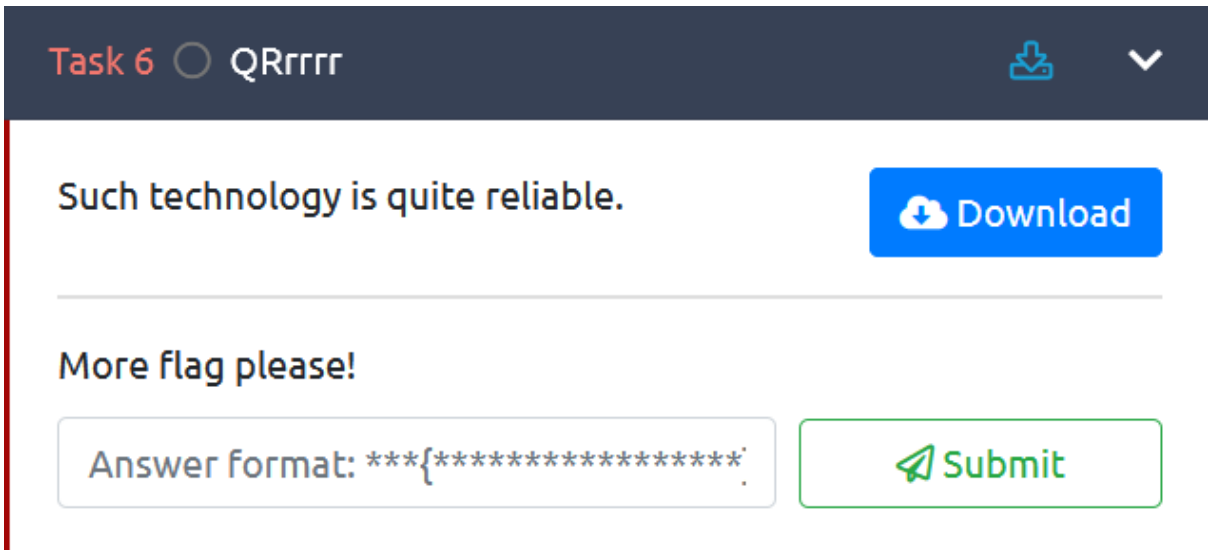
This task was interested and easiest one. Here we got a bit stuck as there was no file or not no clue from there. So, then we decided to see hints and thus it drops



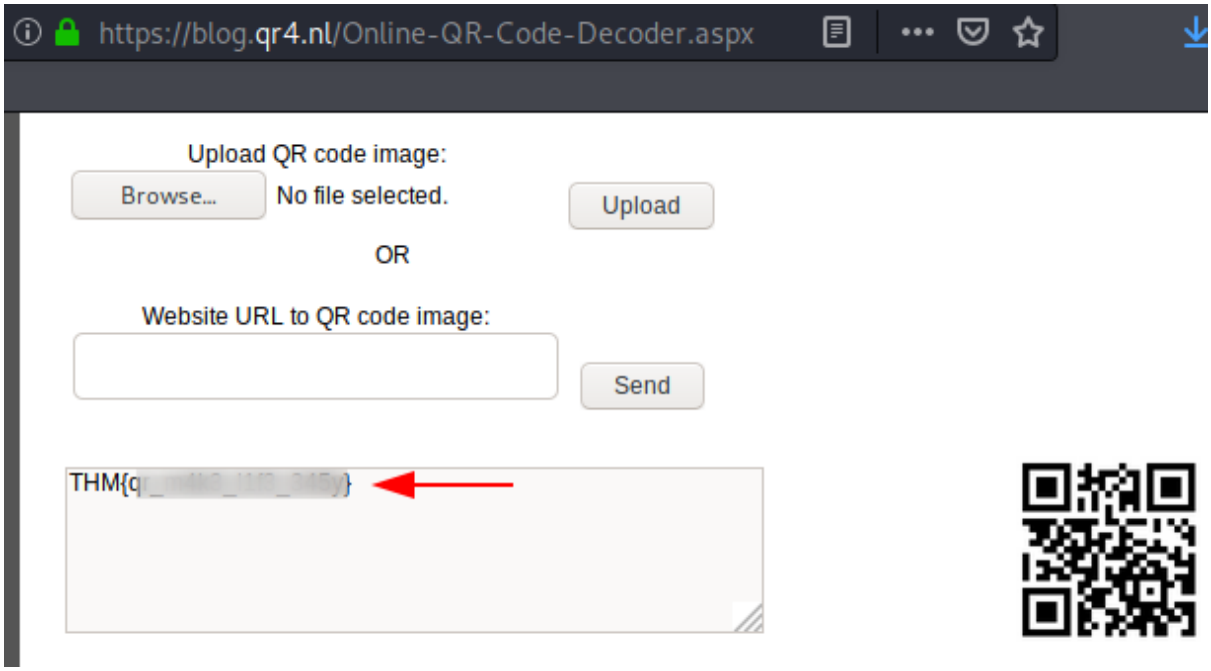
So, the thing that comes to mind is that check the code and when we inspected the element, we got the flag directly over there, or alternately we can also get the flag by highlighting the phrase.



[TASK 6] QRrrrr

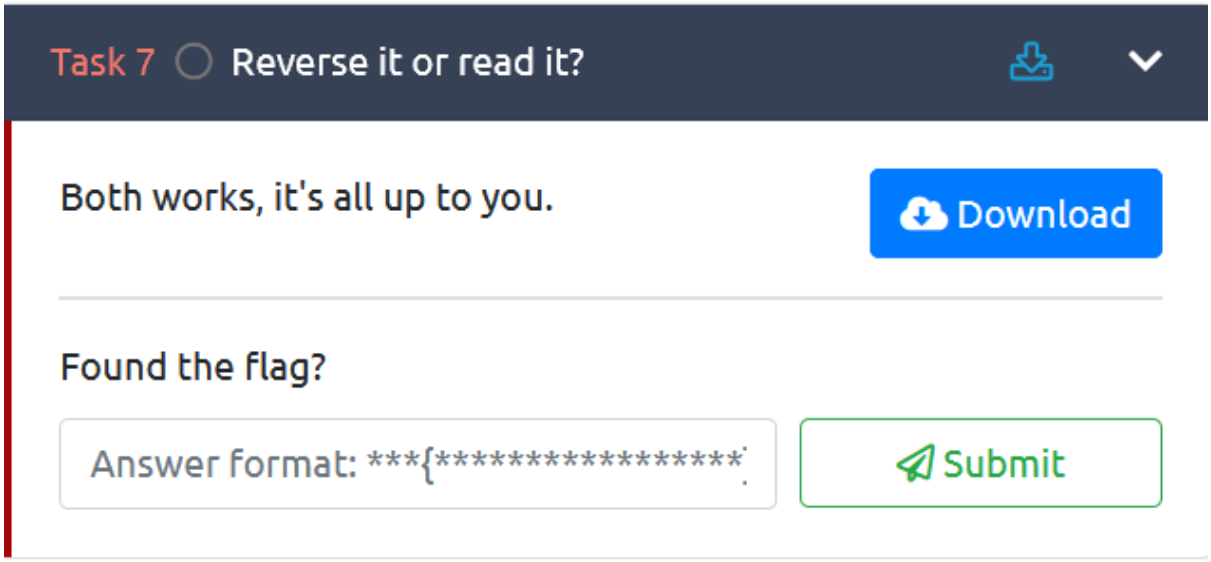


This was the simplest one since the title indicates **QR"** and it was a QR picture when we downloaded the file. So, we already know that the next move is to scan the code.



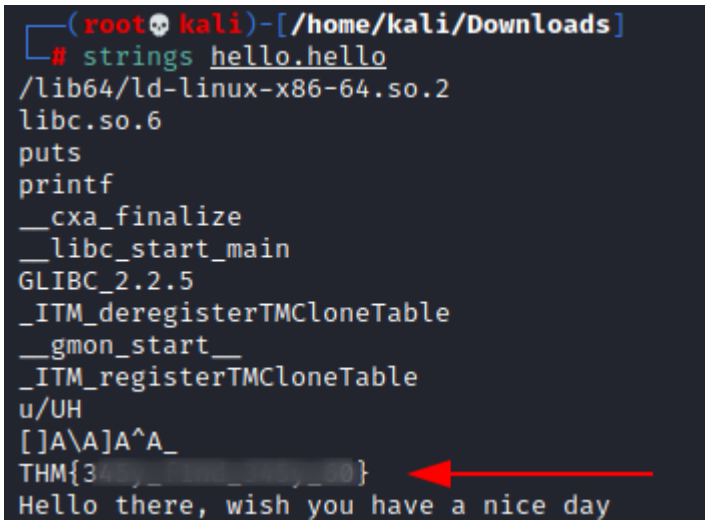
AND BOOM!! We got our flag only after scanning it.

[TASK 7] Reverse it or read it?

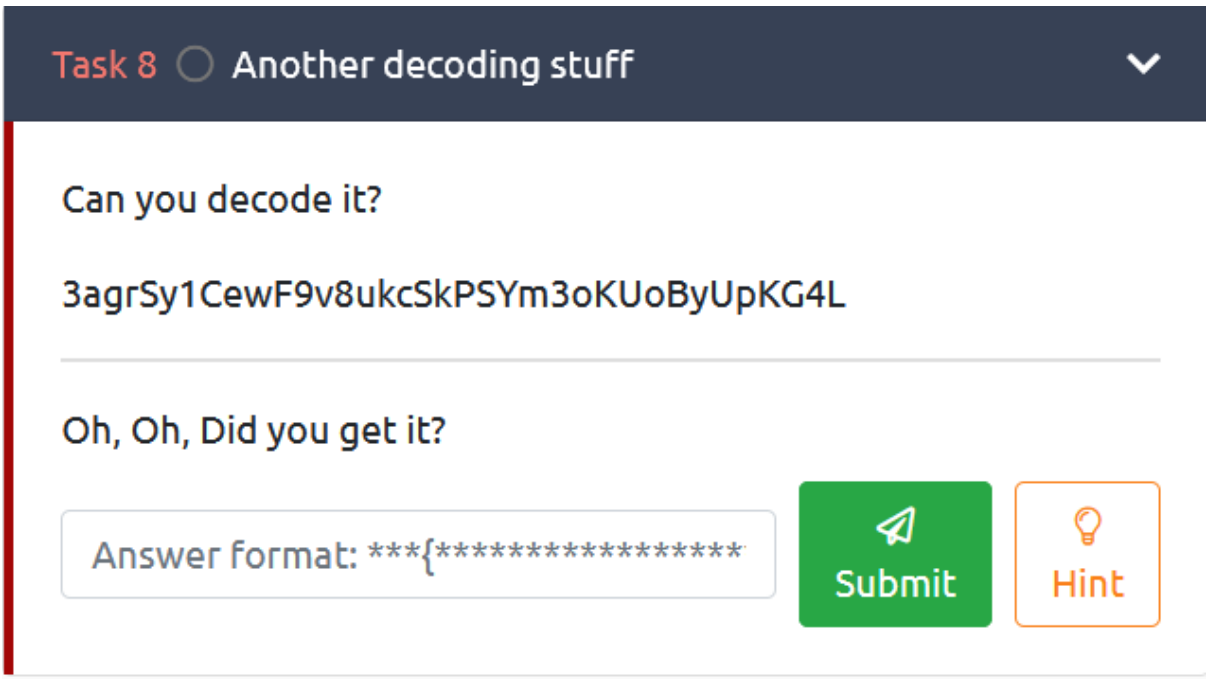


In this task, we should only know about the command to open the elf file (a type of .exe). As soon as we open the file with the following command, we got the flag listed.

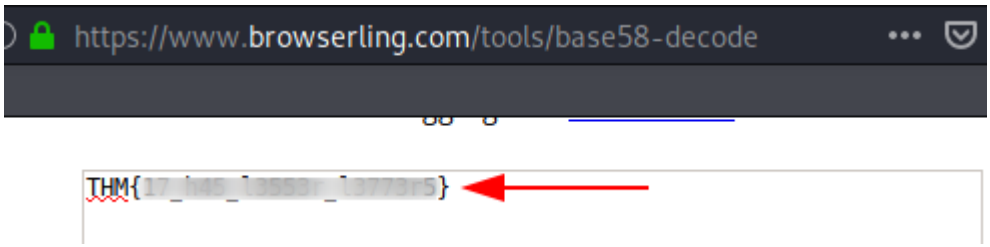
```
strings hello.hello
```



[Task 8] Another decoding stuff

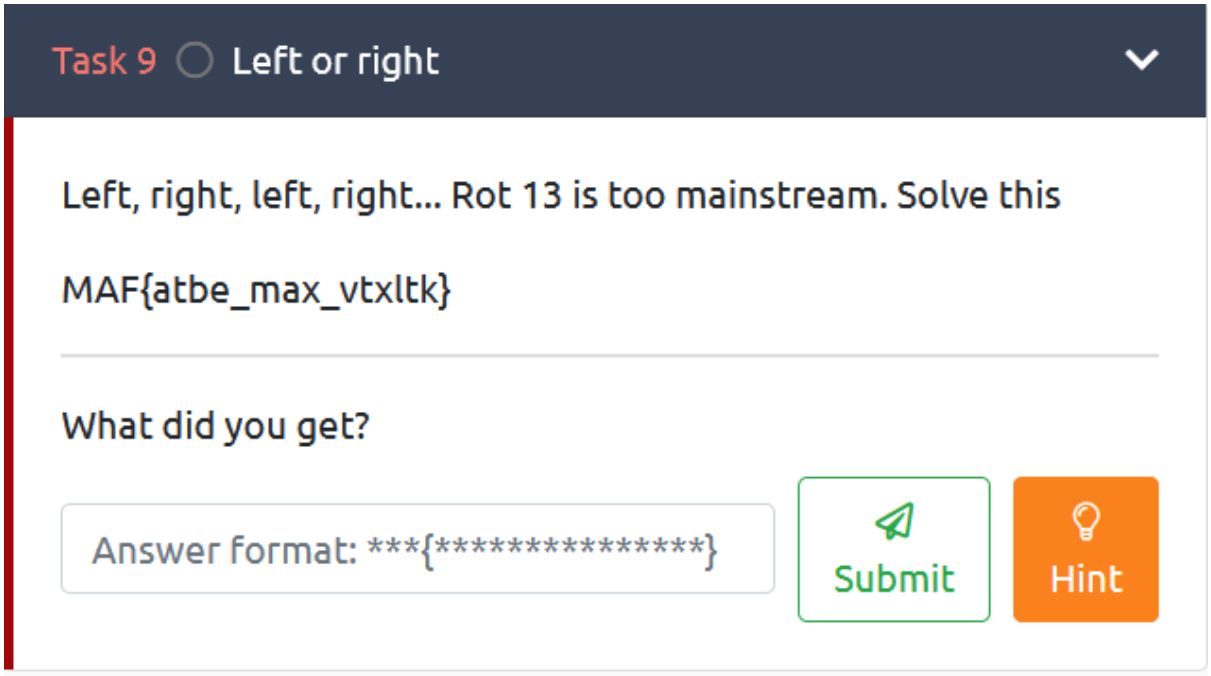


Here, as it says, it is a decoding task which is like most of the cryptography challenges. And we were also led to the base58 algorithm as clues, so we use the online base58 decoder directly.

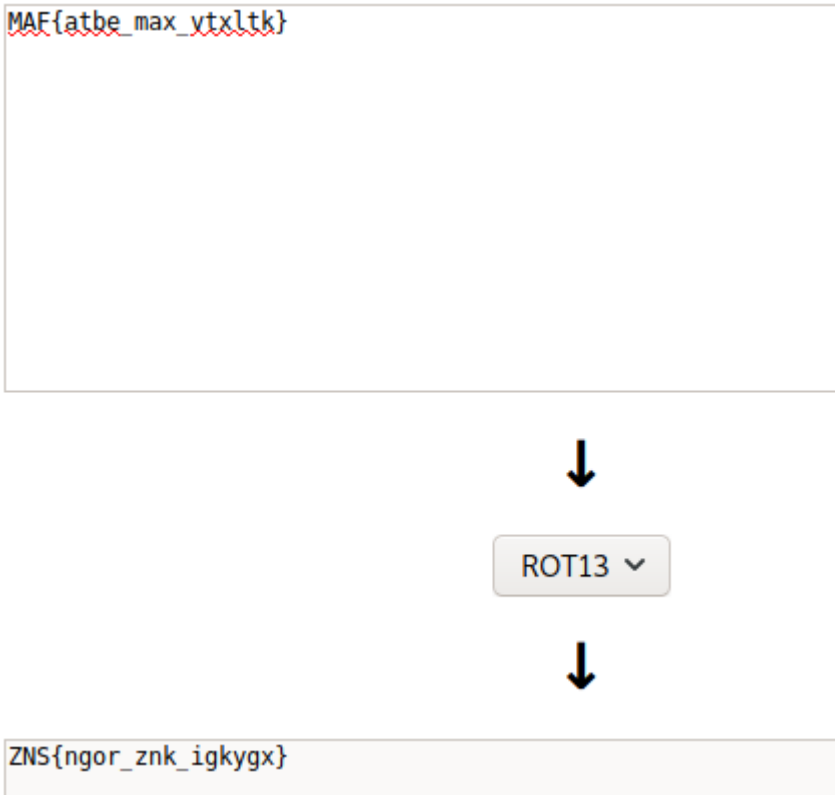


And within a second We got our 8th flag by just decoding it.

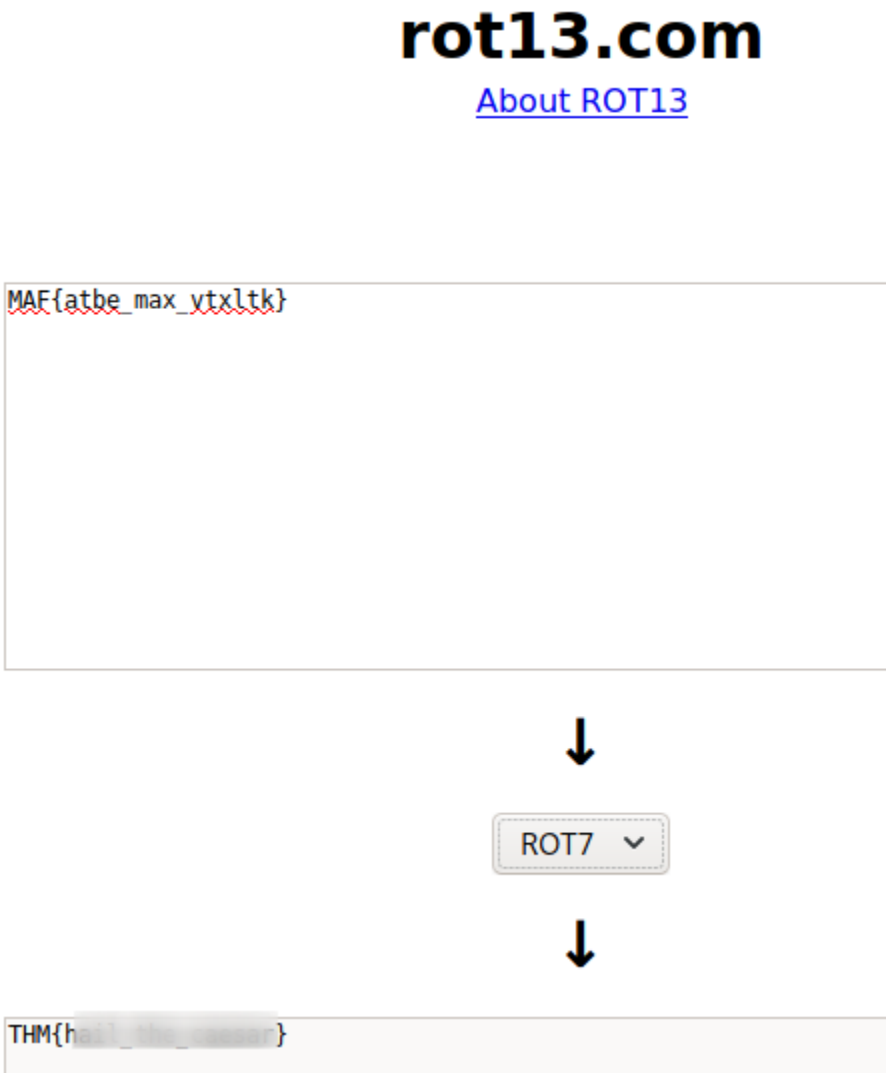
[Task 9] Left or right



We all need a basic presence of mind in this task as it explains that it is a rot 13 algorithm that is a special case of Ceaser cipher encryption technique in which we replace the plain character with the next 13th letter.

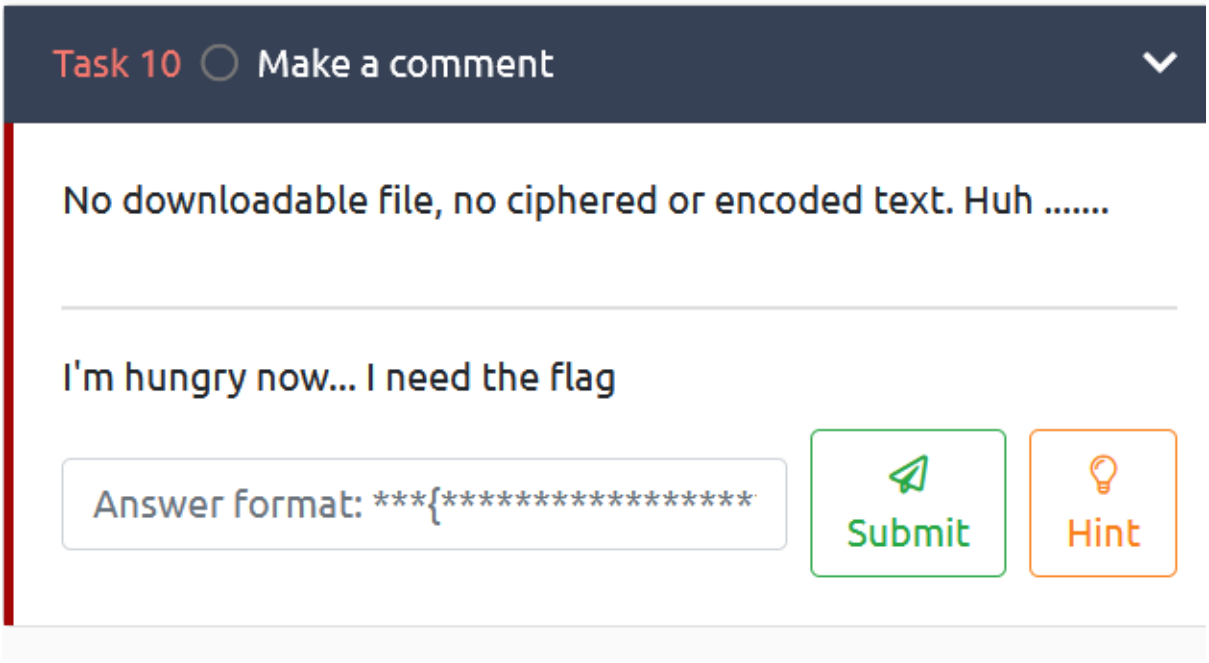


So, here we tried rot13 decoder but it didn't work, so here our next step was to brute force on shift that is 14,15,16 and so on, as this is all about Ceaser cipher substitute

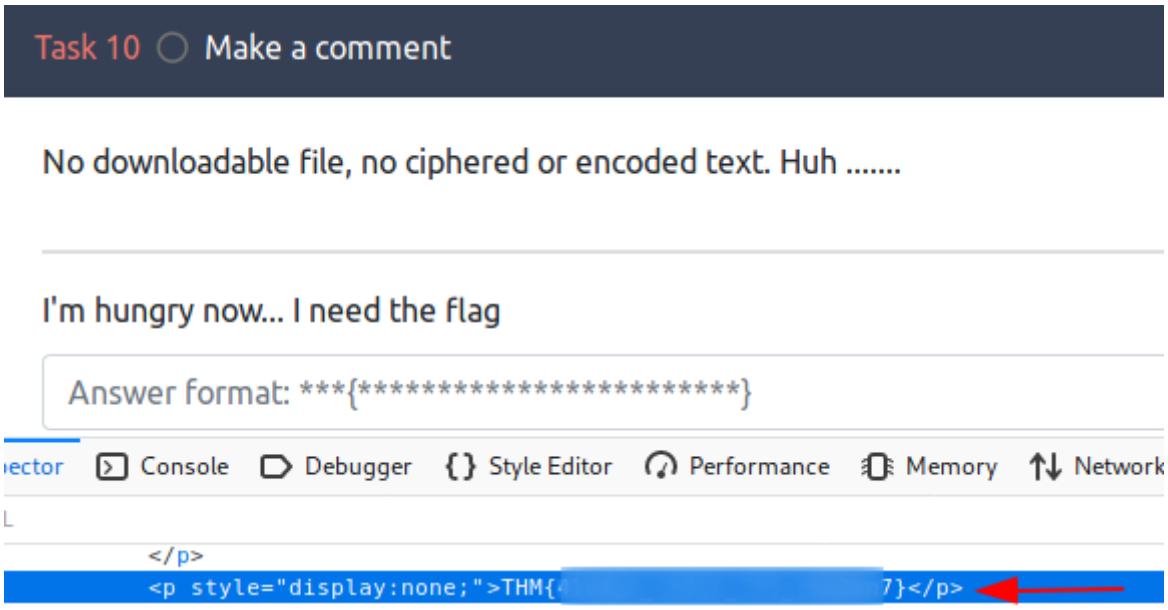


And we got our flag on the 7th shift

[Task 10] Make a comment

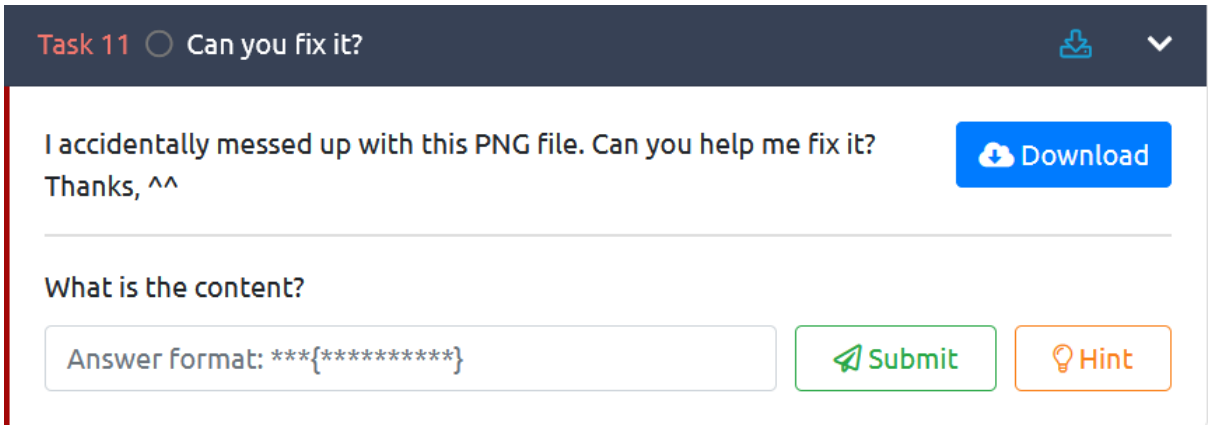


This assignment was the same as Task 5. There was no encoding and no files. So here we thought performing the same steps of task 5 because this task was looking alike as task 5 – No file, No clue in the title. So, here we had to do the inspection again and look for our flag.



And luckily, we received our 10th flag after looking for a while.

[TASK 11] Can you fix it??



This was the most interesting task of the entire lab. As there was a corrupted png but so here we tried to extract the hexadecimal code of that image but after studying a while, we found that its magic numbers vary from the regular magic number.

Magic numbers are initial 8 characters or numbers in hexadecimal code of any file.

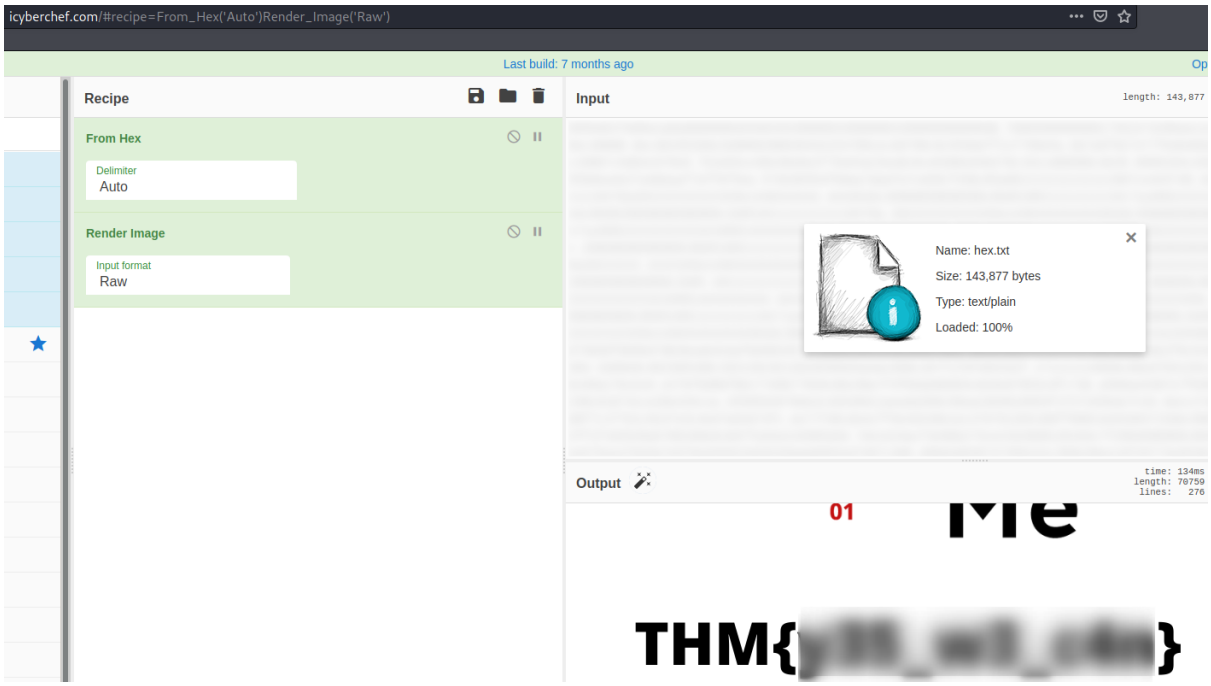
```
┌──(root💀kali)-[/home/kali/Downloads]
└─# xxd --plain spoil.png > hex.txt

┌──(root💀kali)-[/home/kali/Downloads]
└─# head hex.txt
2333445f0d0a1a0a0000000d49484452000003200000032008060000000db
70066800000001735247420 0aece1ce900000009704859730000 0ec40000
0ec401952b0e1b0000200049444154789cecdd799c9c559deff1cf799e5a
bb7a5f927477f640480209201150c420bba288a8805c19067c5d64c079e9
752e03ce38e30e8e2f75e63a23ea8c0ce8308e036470c191cd80880c4b20
0909184c42b64ed2e9f4bed7f23ce7fe51559dea4e27a4bbaaf7effbf5ea
57d2d5554f9daa7abafa7ceb9cf33bc65a6b1111111111111907ce443740
4444444444660e05101111111111119370a202222222222326e1440444444
444464dc28808888888888c8b851001111111119171a300222222222222
e3460144444444444c68d028888888888888c1b0510111111111119370a
```

So, as we searched on google and we found that the standard magic numbers of the png image are **89 50 4E 47**

We have to just replace it.

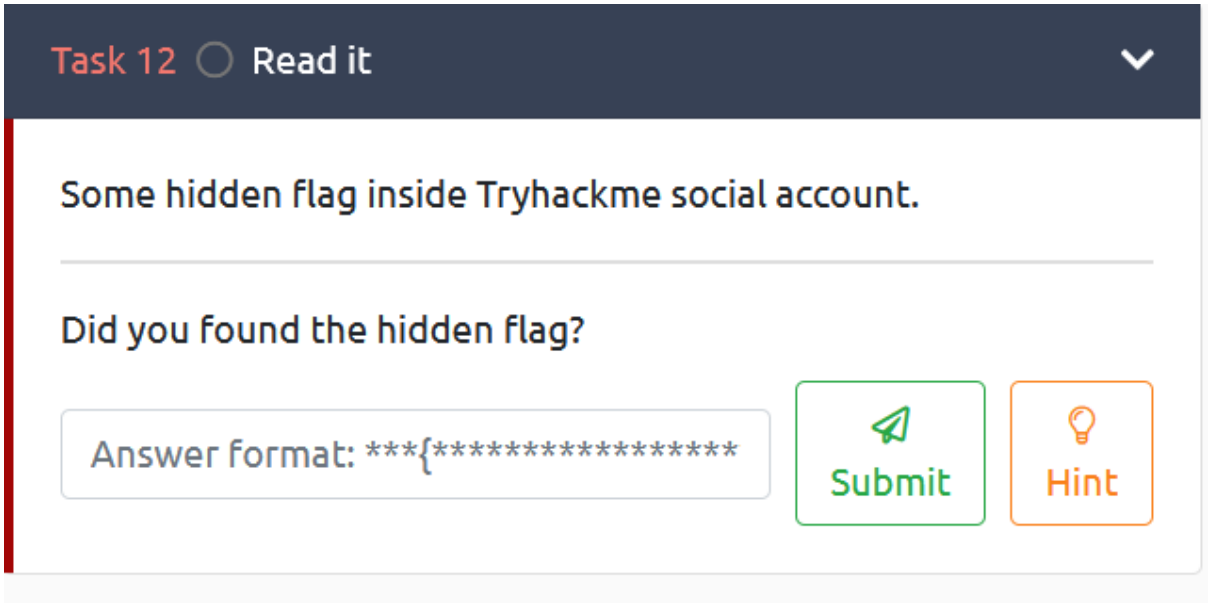
```
┌──(root💀kali)-[/home/kali/Downloads]
└─# head hex.txt
89504E470d0a1a0a0000000d4948445200000320000003200806000000db
70066800000001735247420 0aece1ce900000009704859730000 0ec40000
0ec401952b0e1b0000200049444154789cecdd799c9c559deff1cf799e5a
bb7a5f927477f640480209201150c420bba288a8805c19067c5d64c079e9
752e03ce38e30e8e2f75e63a23ea8c0ce8308e036470c191cd80880c4b20
0909184c42b64ed2e9f4bed7f23ce7fe51559dea4e27a4bbaaf7effbf5ea
57d2d5554f9daa7abafa7ceb9cf33bc65a6b1111111111111907ce443740
4444444444660e0510111111111119370a2022222222223 26e1440444444
444464dc28808888888888c8b8510011111111119171a300222222222222
e34601444444444444c68d028888888888888c1b0510111111111119370a
```

Now our last step is to render the image from updated hexadecimal code which we did use cyber chef where we have to render the image from updated code

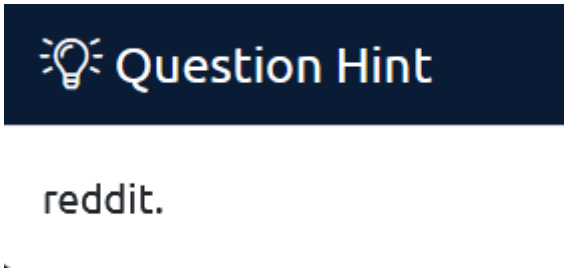


And hence we got our 11th flag in the image.

[TASK 12] Read it

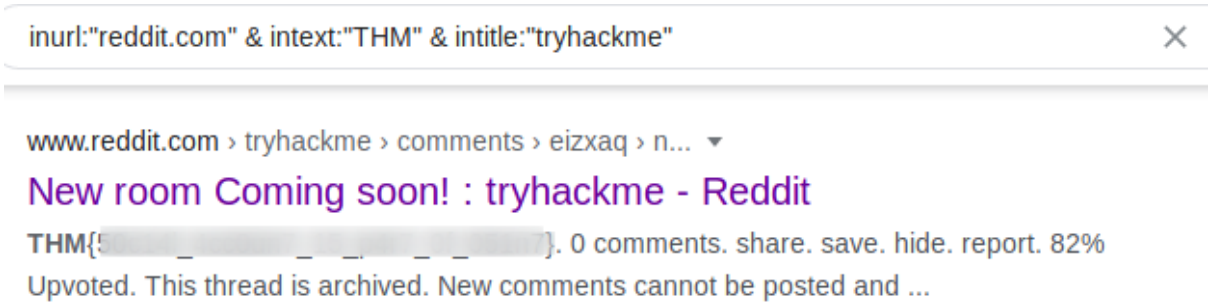


In this task, our flag is present on the social media account of tryhackme. But our confusion was to explore on which platform either Twitter or something else.



But provided hint cleared it for us. So here we got a chance to explore our google dork technique. And the perfect dork came to be

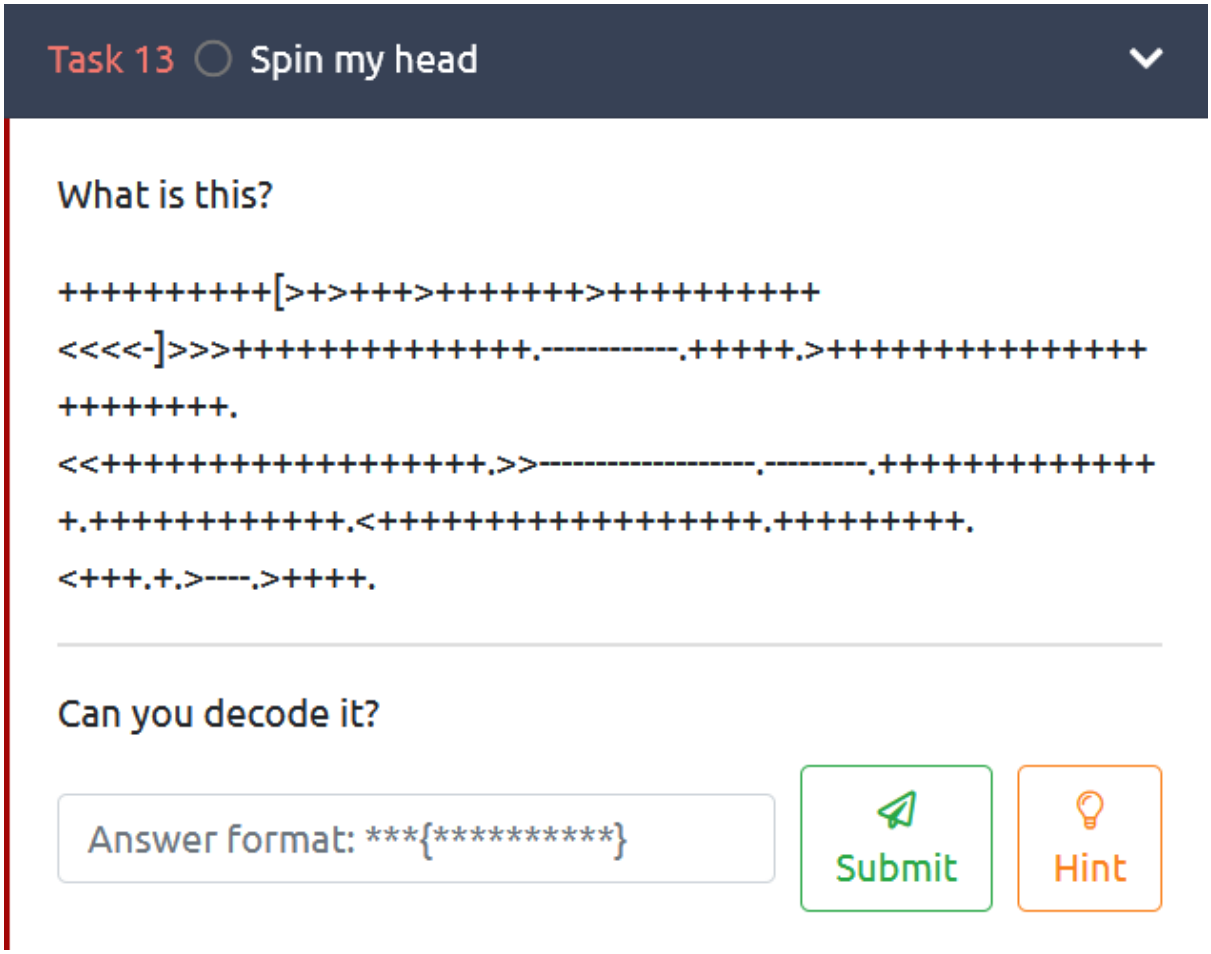
```
inurl:"reddit.com" &intext:"THM" & intitle:"tryhackme"
```



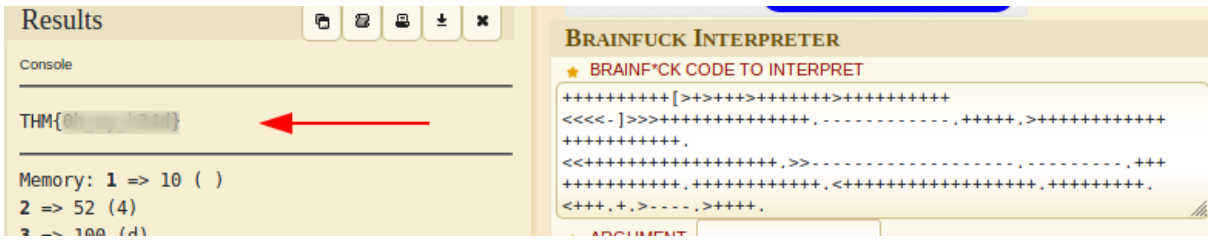
After going through it we can easily spot our flag where the flag was hidden in the image

THM{50c14l_4cc0un7_15_p4r7_0f_051n7}

[TASK 13] Spin my head

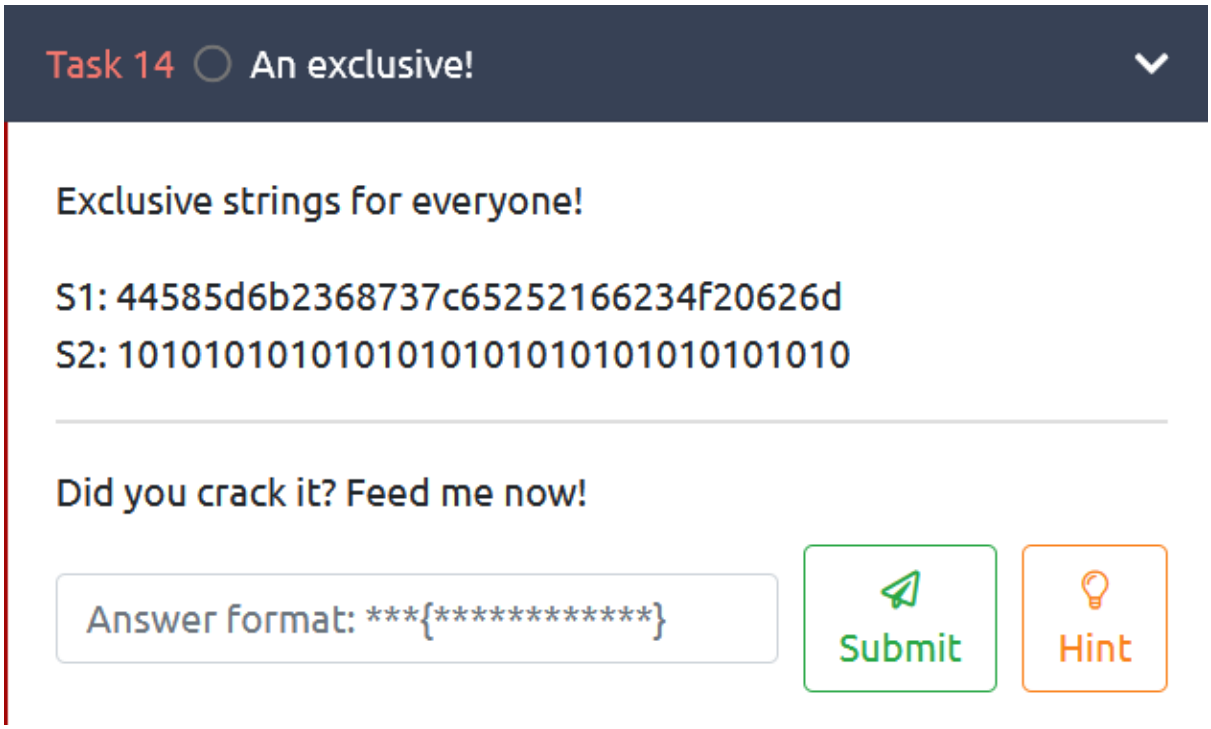


This task was spinning my mind as it told me to decode it but we were clueless about it that's why we had to go for a hint, and they indicated to its **"BINARYFUCK"**

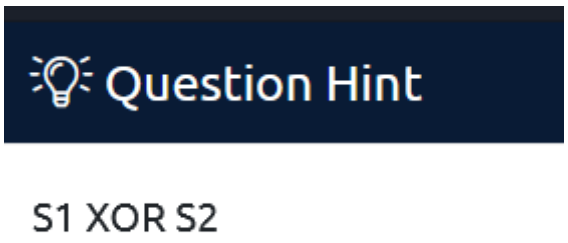


So, we directly go to binaryfuck decoder and hence we got our 13th flag just by decoding it.

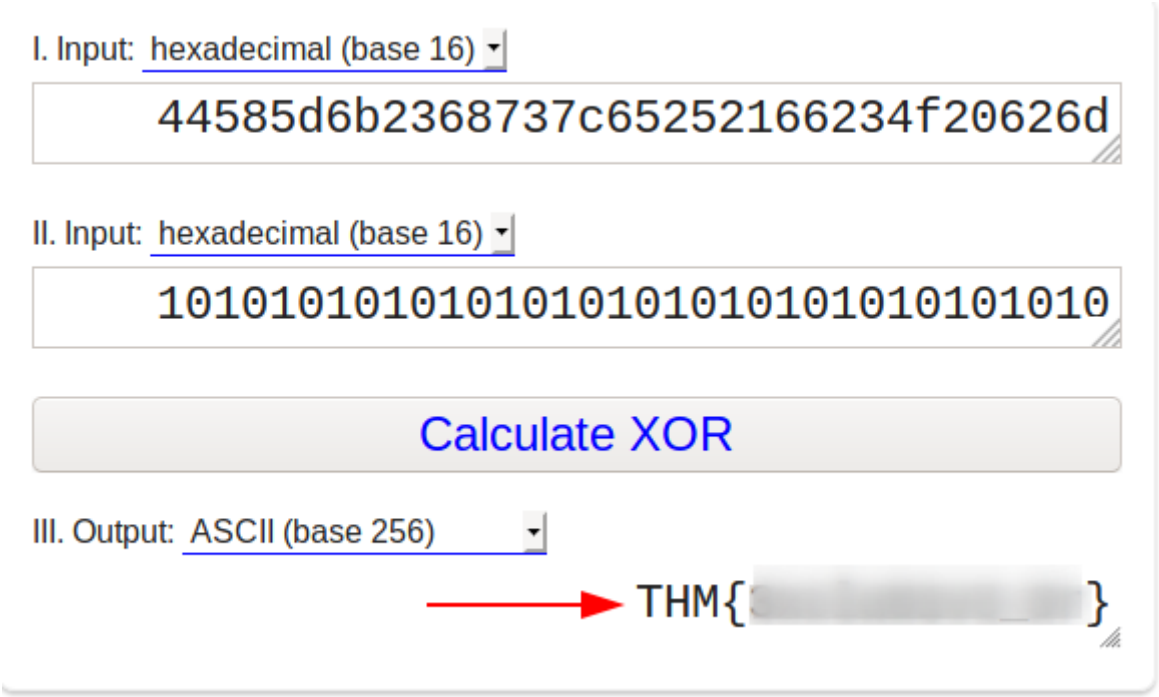
[TASK 14] An exclusive!



In this task, we got 2 strings but we didn't know what to do with these strings but our saviour hint shows to XOR these strings.

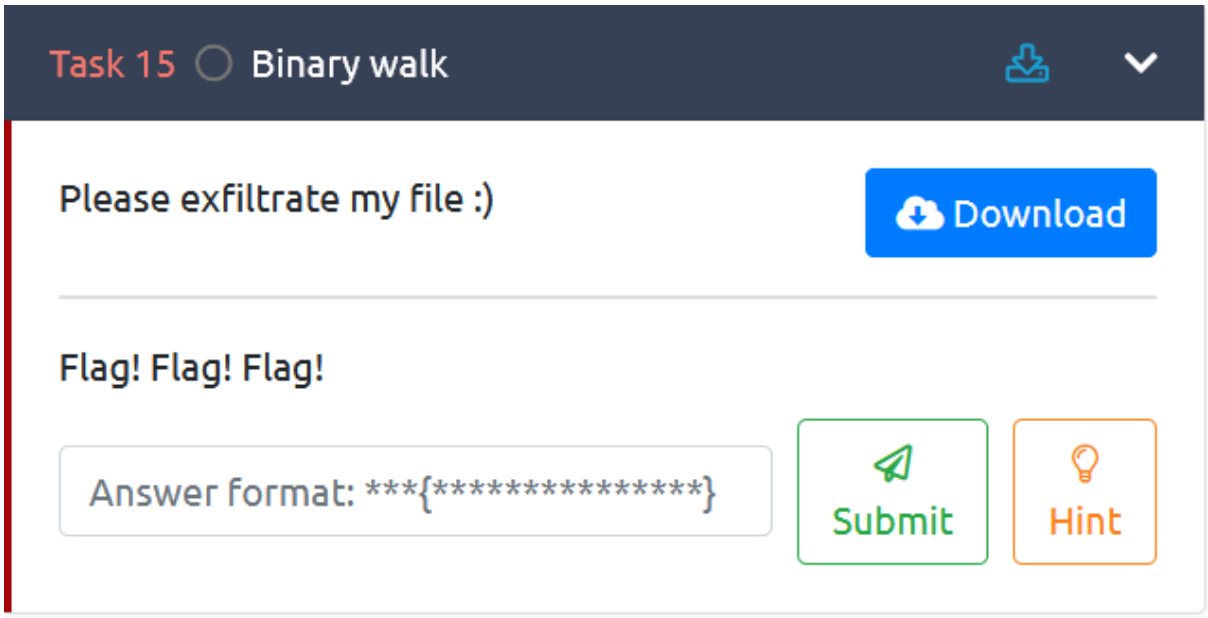


Now we can relate exclusive and XOR.



So, we can do it easily, as many tools are available online. And we got our 14th flag without effort.

[TASK 15] Binary walk

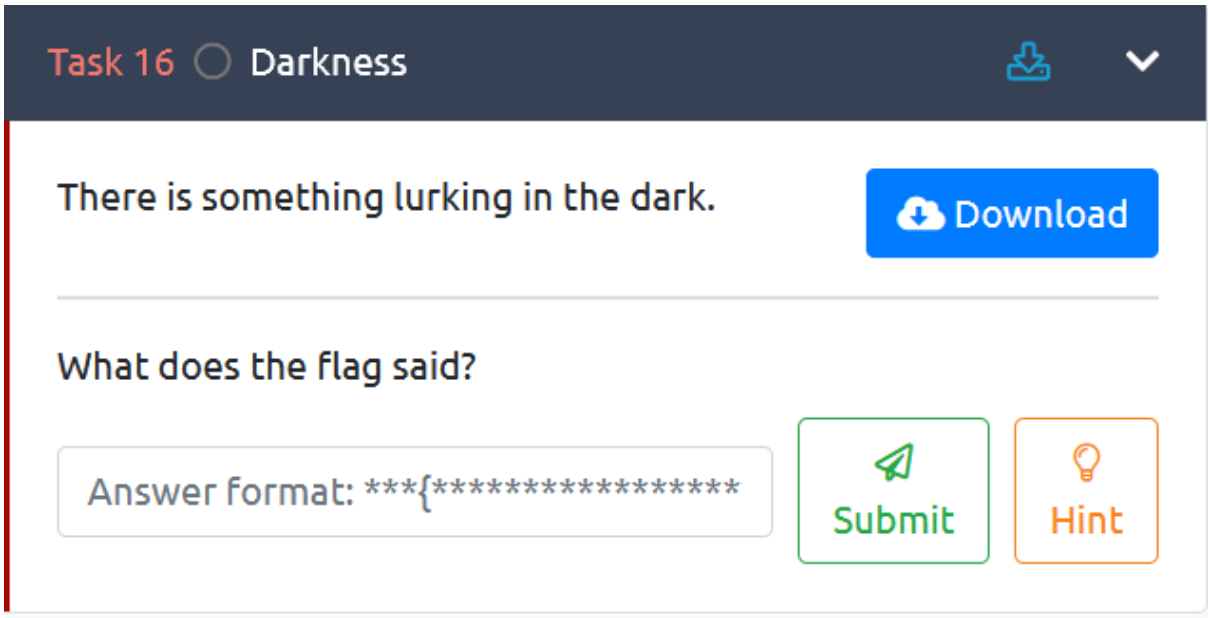


As this task name suggests a binary walk and clue also directed us to binwalk, it is a tool binwalk that is helpful in steganography. But it is only used to extract the hidden file from the images.

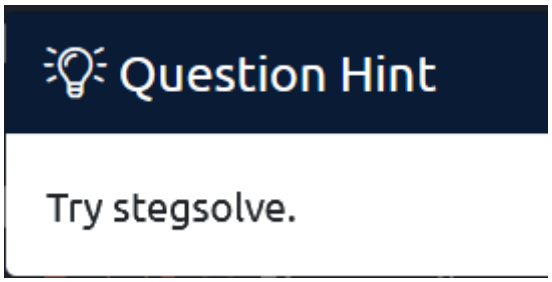
```
root@kali:~/Downloads# binwalk hell.jpg -e

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.02
30            0x1E            TIFF image data, big-endian, offset
265845        0x40E75         Zip archive data, at least v2.0 to
.txt
266099        0x40F73         End of Zip archive, footer length:

root@kali:~/Downloads# cd _hell.jpg.extracted/
root@kali:~/Downloads/_hell.jpg.extracted# ls
40E75.zip  hello_there.txt
root@kali:~/Downloads/_hell.jpg.extracted# cat hello_there.txt
Thank you for extracting me, you are the best!

THM{y0u_w1ll_n3_0u7}
```

So, we extracted files from jpg and found one txt file. And thus, we got our 15th flag in this file.

[TASK 16] Darkness



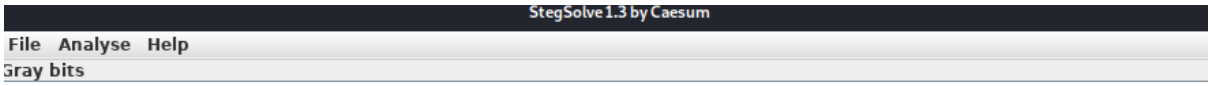
In this task our hint is stegsolve. Stegsolve is a steganographic image analyzer, solver, and data extractor.



We have to evaluate our image by filtering various colours in an image using stegsolve. We can download stegsolve with these terminal commands

```
wget http://www.caesum.com/handbook/Stegsolve.jar -O stegsolve.jar
chmod +x stegsolve.jar
```

Now just we have to open the image with the stegsolve



THM{7h3r3_15_h0p3_1n_7h3_d4rkn355}

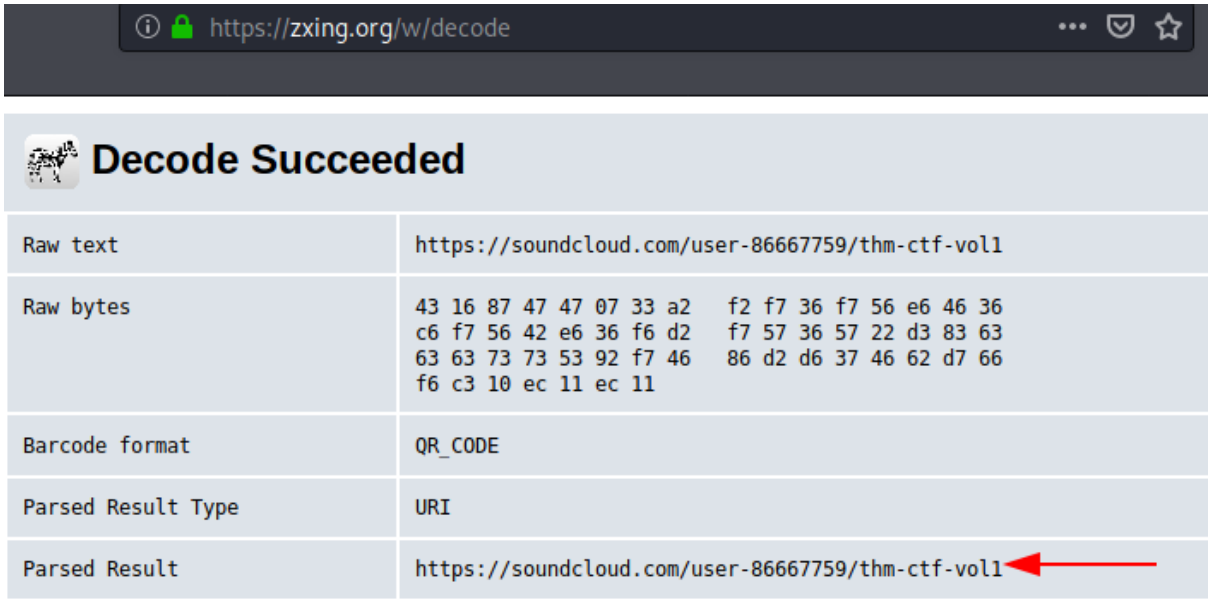
In the first go, it is just the black image but we have to constantly analyze the image under different colours so the odd one can shine out like this.

And hence we got our 16th flag

[TASK 17] A sounding QR

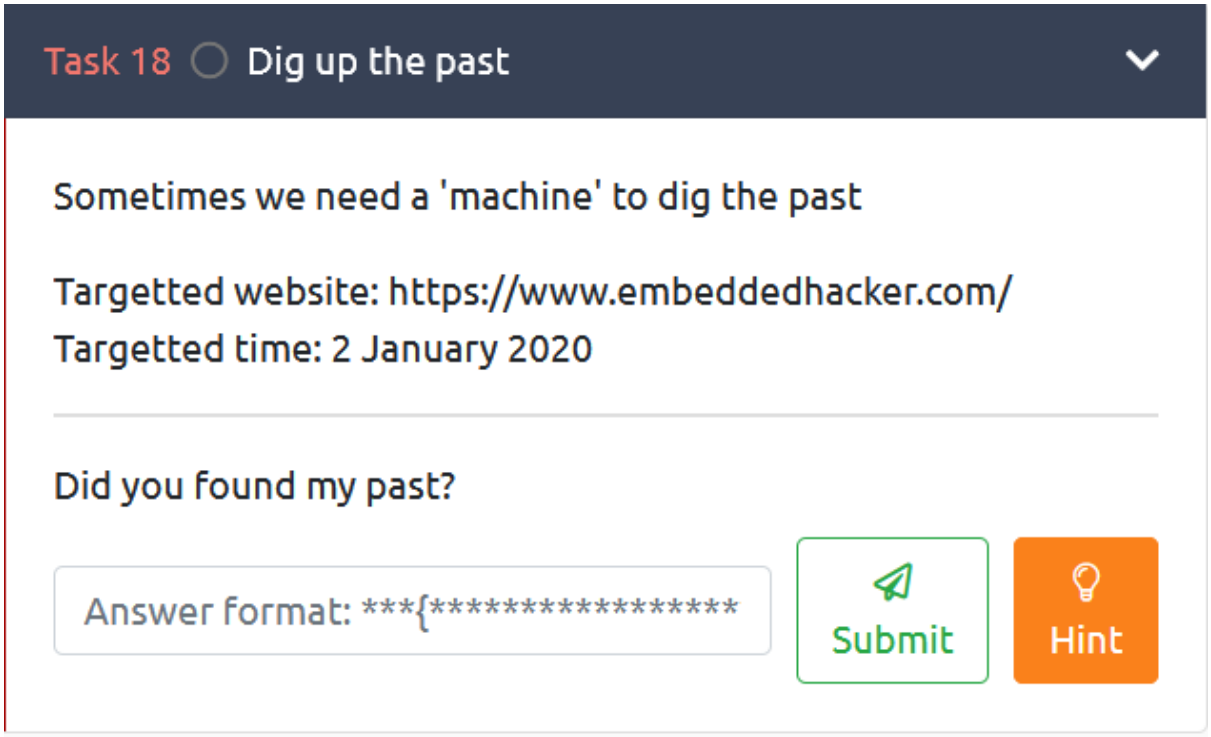
This task is quite similar to task 6. In this task, we got a QR image and we will scan it and fetch information from there.

But interestingly we got a link over there which redirects to an audio link



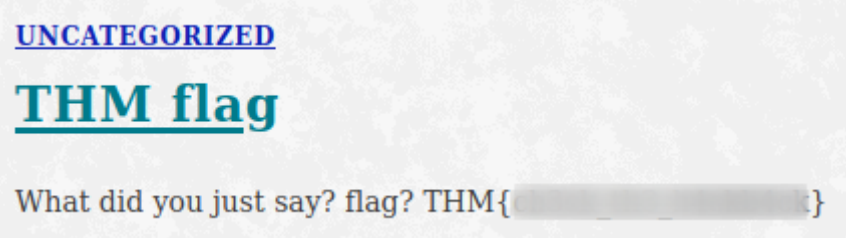
But after listening to it carefully in slow motion we got our 17th flag.

[TASK 18] Dig up the past



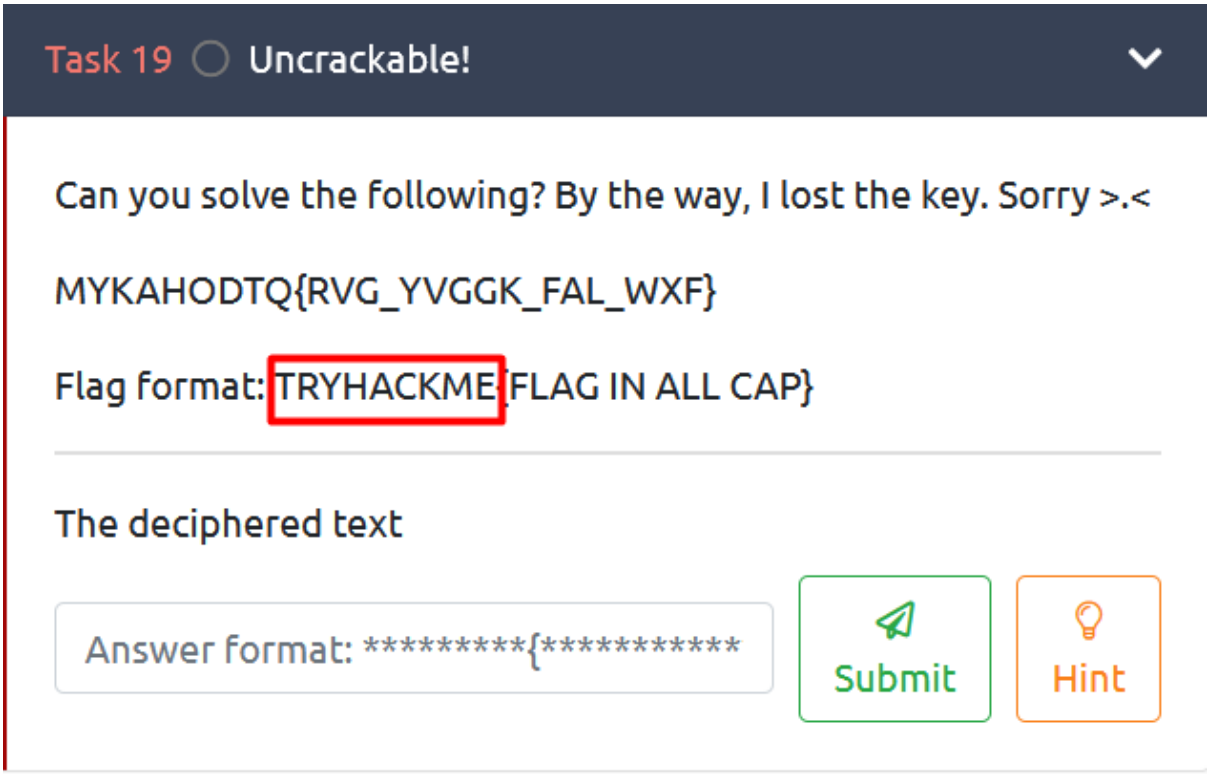
In this task as a title relates to the past the only thing that pops up in our mind is the Wayback machine and the time and date in the description make our doubt clear.

So, for checking past captures or archives of any site we can use the Wayback Machine. After analyzing the output, we can see there is a capture of the website on 2nd Jan 2020



But when we visited that website, we got our 18th flag.

[TASK 19] Uncrackable



We have an embedded text in this role so that we can attempt to decipher it. So, we tried it with Ceaser cipher and several other methods, but we get to know this from clues that this is vigenere cipher.

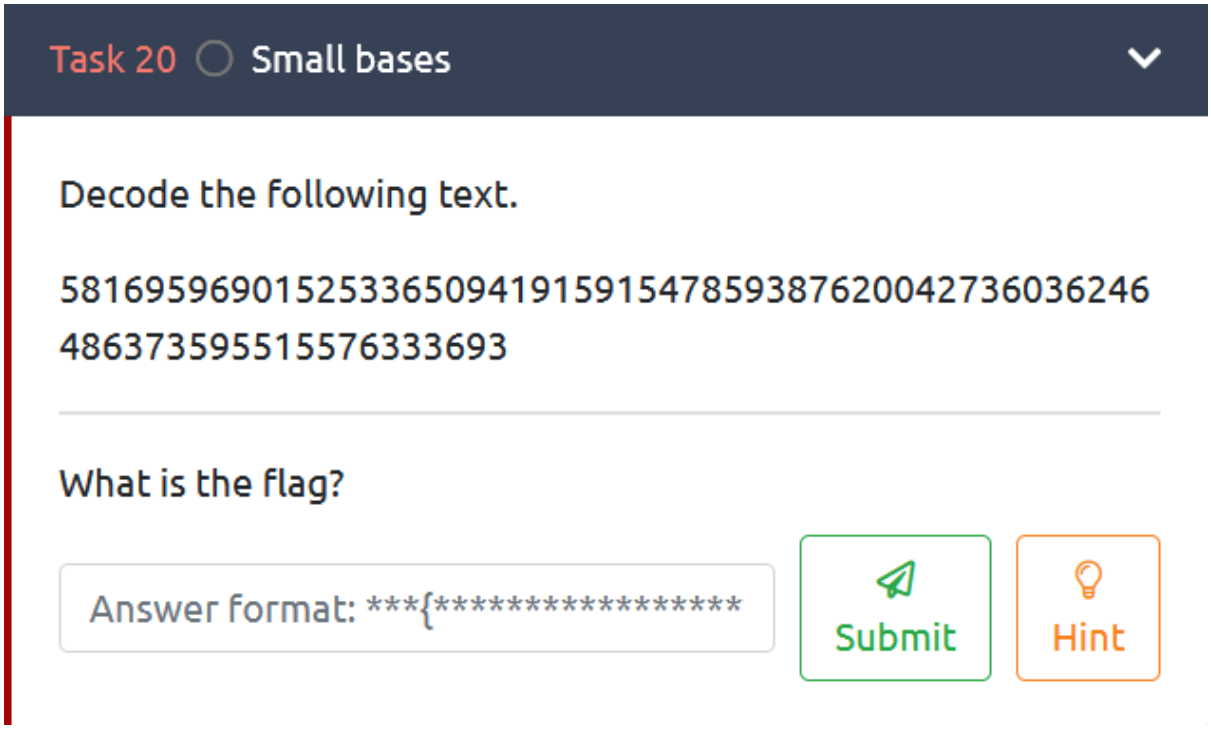
But now the problem begins because while decoding it we require a key.

So there is a strange thing that we can notice is all flags are in the format of THM{……} but in this task flag format is TRYHACKME{……} that's why we used it as a key and got another encoded value.

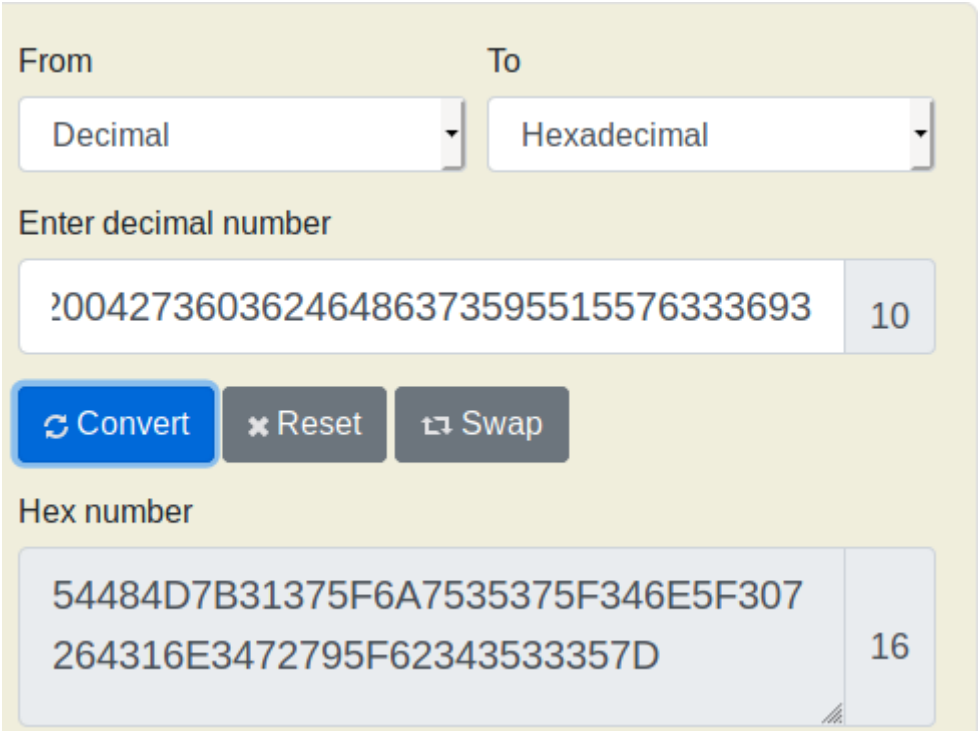
It looks like we found another constant THMTHMTHM. Now we can either attempt to decrypt consecutively using the same key that is TRYHACKME or we can also use THMTHMTHM as a key to decode MYKAHODTQ{RVG YVGGK FAL WXF}

But we got our 19th flag by using THMTHMTHM as a key.

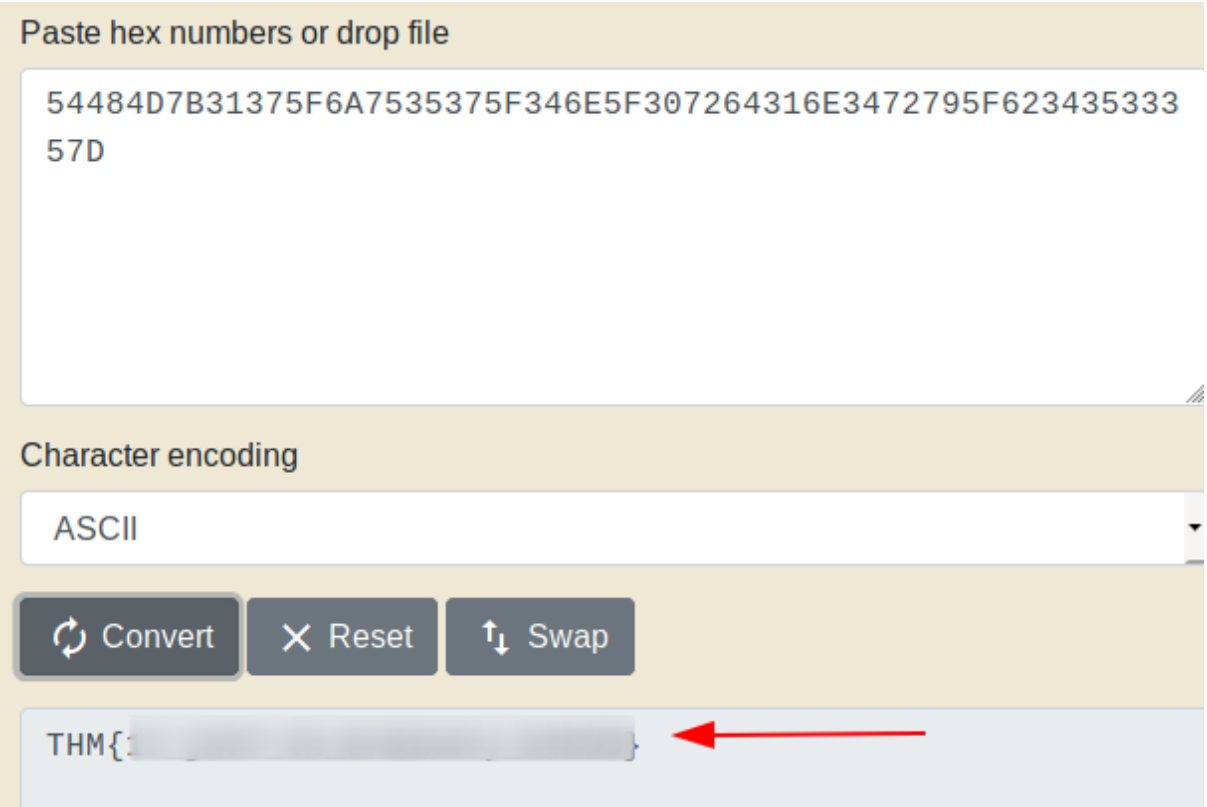
[TASK 20] Small bases



This task was also of cryptography. In this task, the hints say **(dec-> hex -> ASCII)** we have to simply follow these tips that are converting decimal to hex first



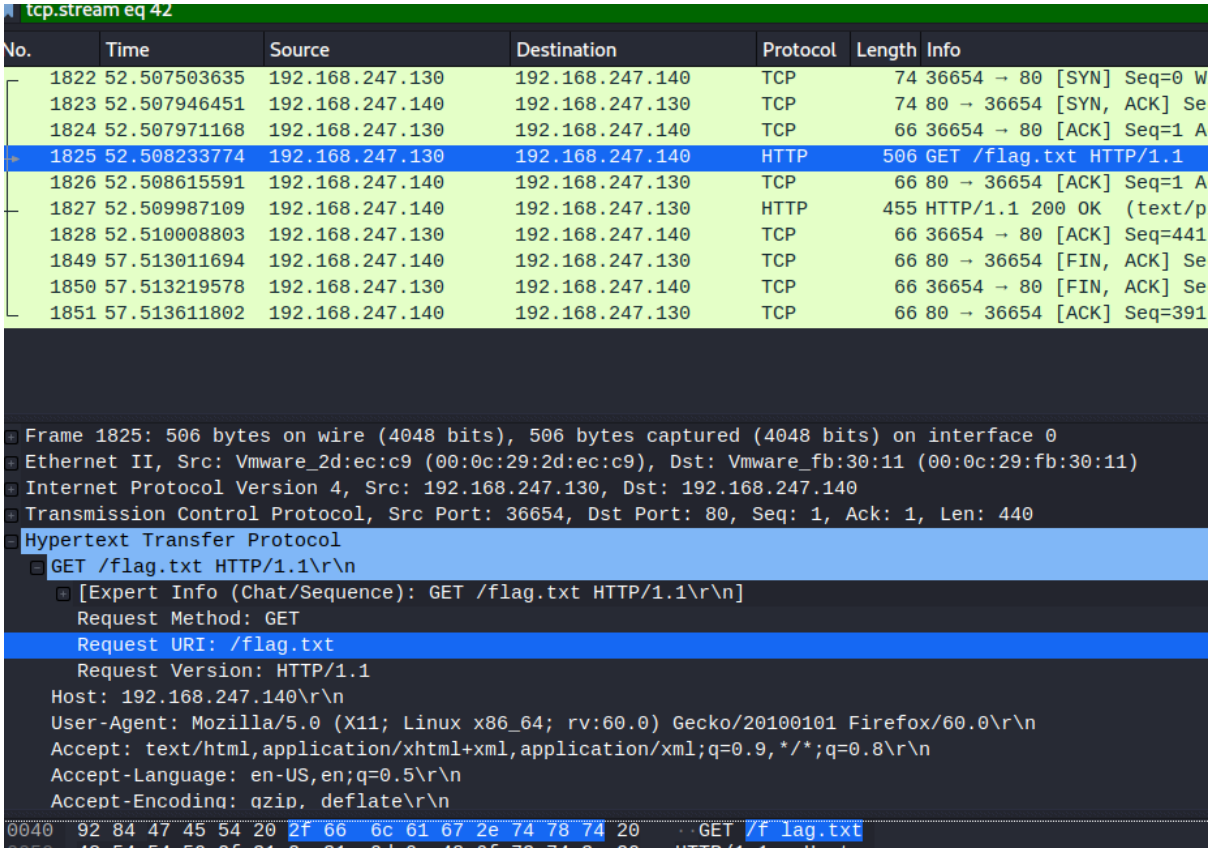
Then Hex to ASCII



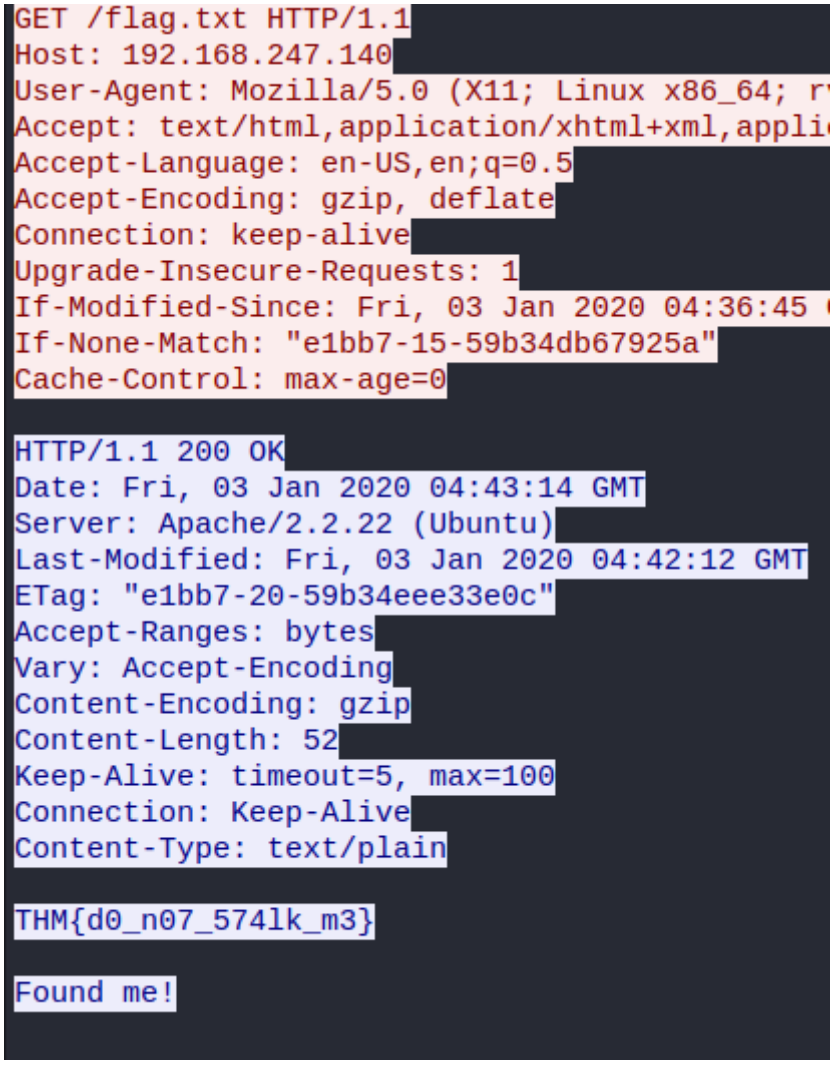
And hence we got our 20th flag

[TASK 21] Read the packet

In this task when we download the file it's a pcapfile. So, it cleared that it's a Wireshark file



And as the task name suggests we have to read any packet which is in any packet so after applying many filters and searching we get to the request over which flag.txt has been sent so we have to just open up that file with the help of stream.



And here we go we secured our last flag.

MISSION ACCOMPLISHED!!

Conclusion

Hence, one can make use of these commands as a cybersecurity professional to assess vulnerabilities on systems and keep these systems away from threat.

References

- https://www.hackingarticles.in/ctf-collection-vol-1-tryhackme-walkthrough/
- https://tryhackme.com/room/ctfcollectionvol1