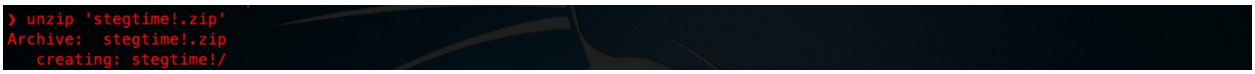
stegCTF writeup on Tryhackme

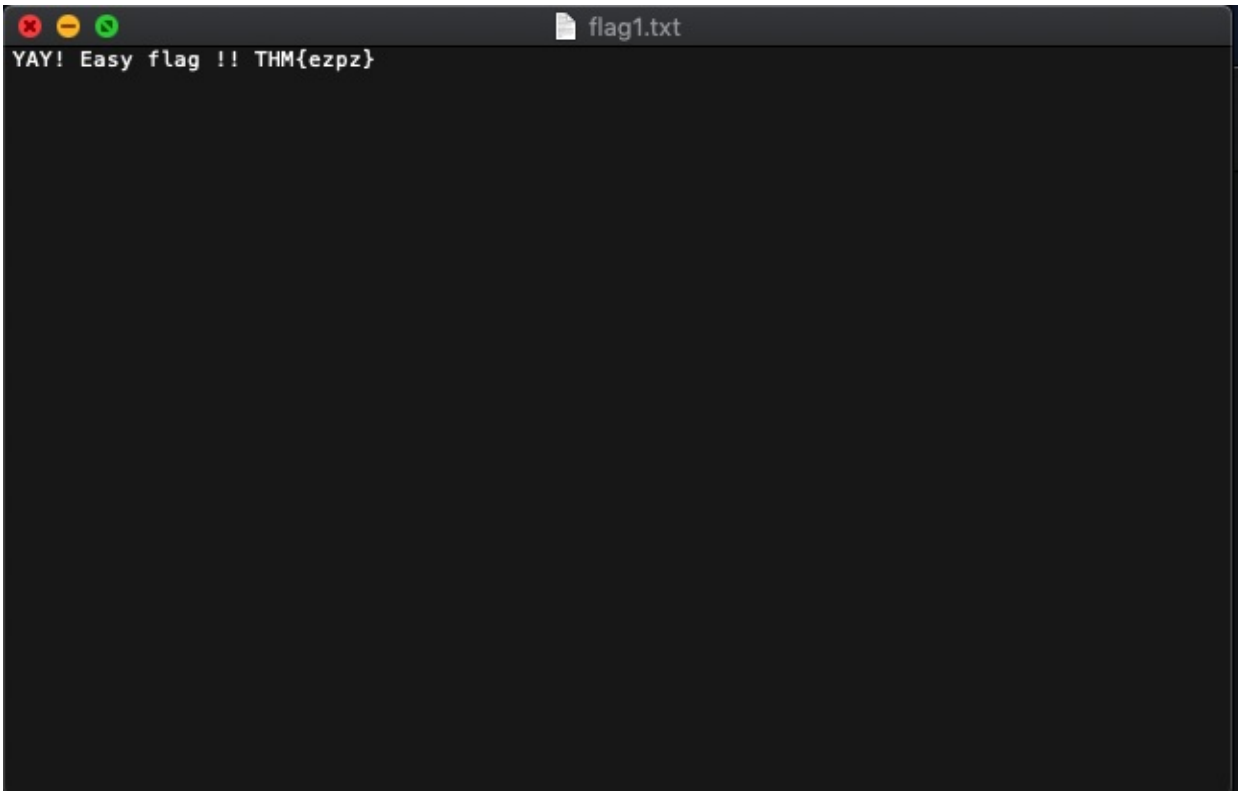
The following writeup is for the room stegCTF on tryhackme.

TASK 1: Download the zip file and unzip it.



Task 2 : Finding the flags!

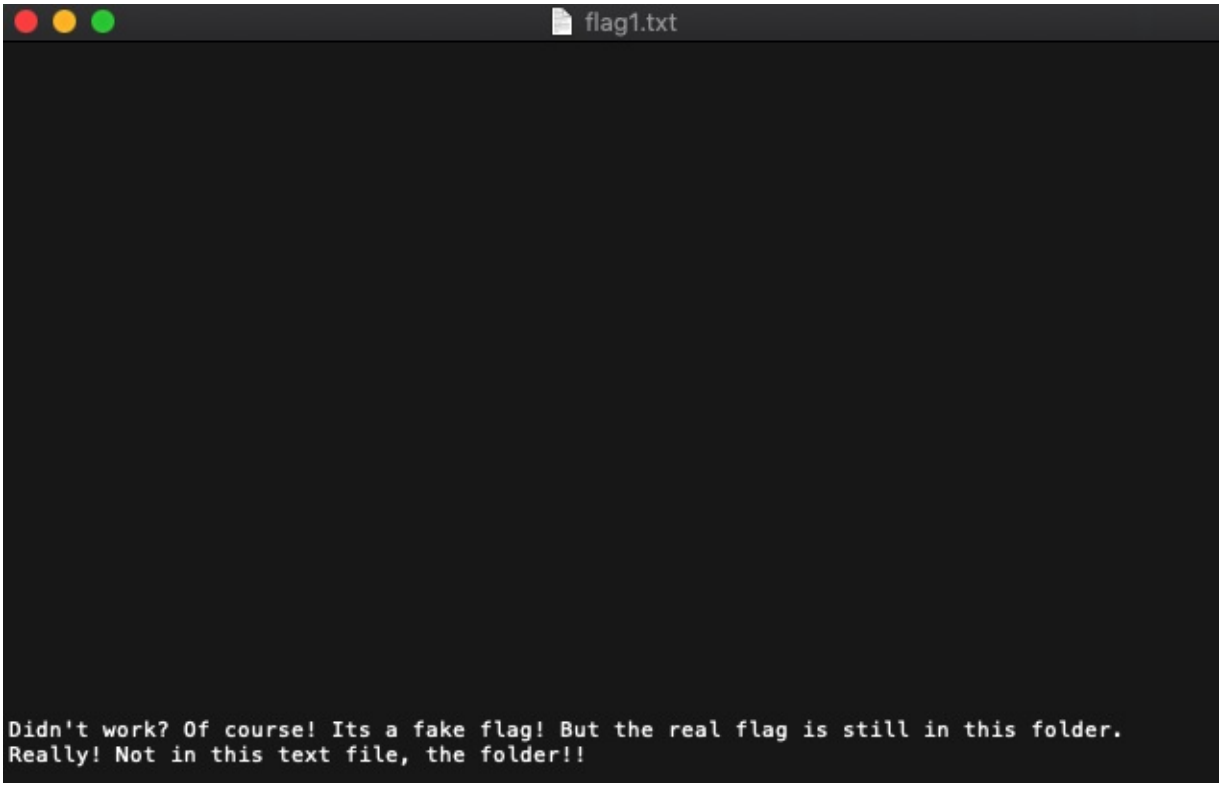
Flag1 :- Lets investigate the extracted file . It is imperative that to find flag1 we have to check flag1.txt. Open flag1.txt.



Well that was easy! Well...the flag does not work, no biggie! We scroll down on

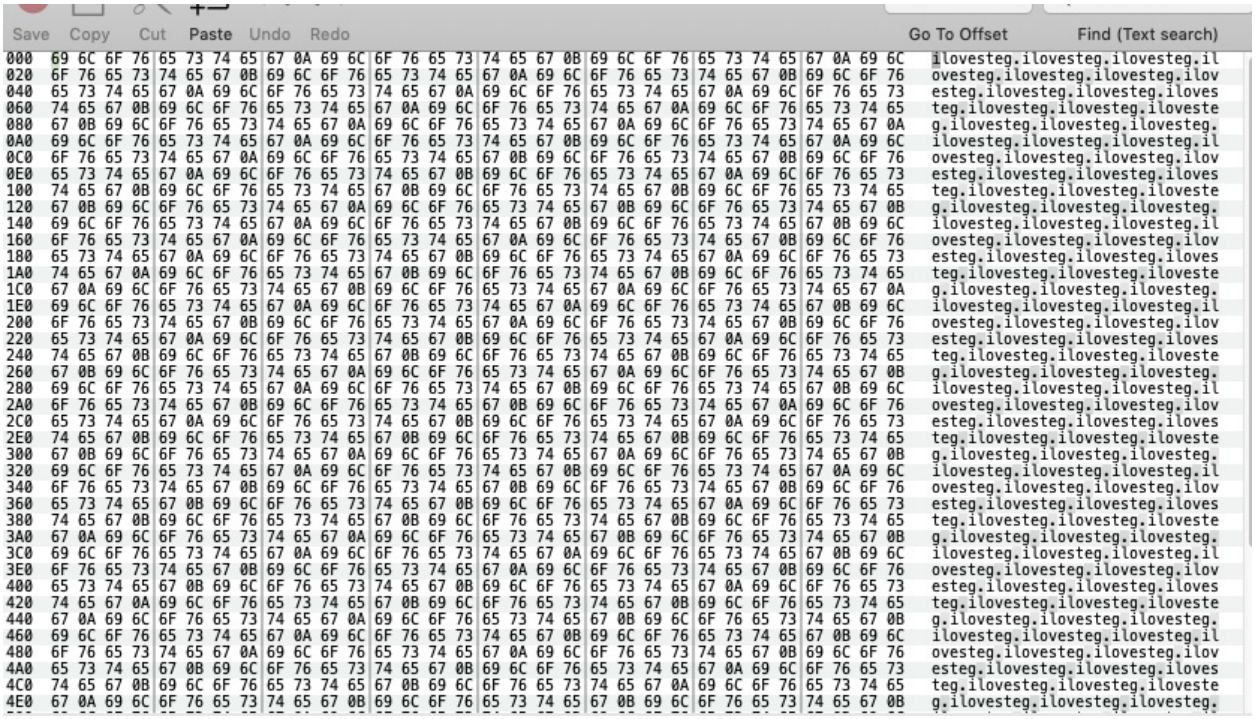
the text file and

find this.



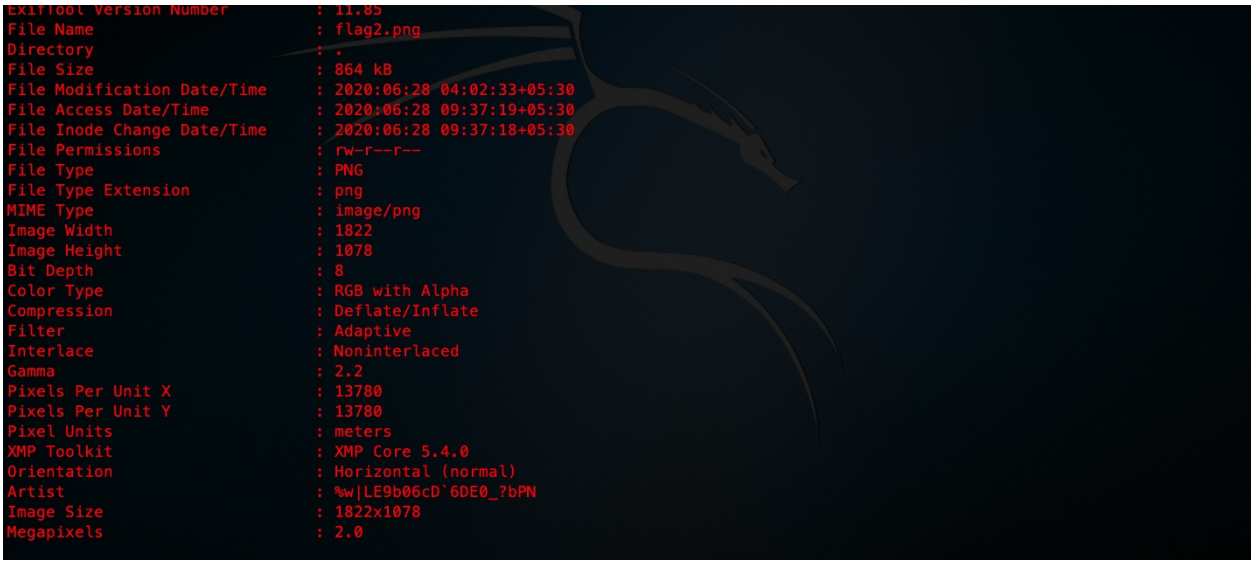
Now it says the flag is in same folder. Lets check the hidden directories.

We find the hidden directory! We find that it contains a text file with ‘ilovesteg’ repeated over and over. On opening the file in a hex editor, we that the newlines are actually different for some.



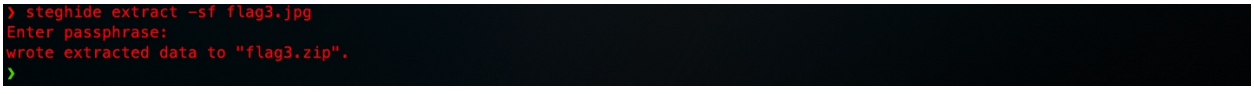
Lets copy the hexdump, and remove the characters except the newlines(0a and 0b). This looks like a binary arrangement. On replacing 0a with 0 and 0b with 1 we get the binary. Decode this binary to text to get the flag!

Flag2: We are given a flag2.png. Lets check out the image's metadata.(since its an image of a meta knight from kirby's adventure!) We use exiftool for this purpose.



We notice that there in the artist section we have some characters under the artist section. Rot47 on them to get the flag!

Flag3: We have the flag3.jpg given to us. On extracting the data without password, we get a new zip file!



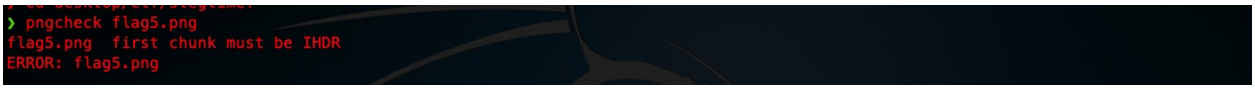
On trying to open we find that the file is password protected. Lets run strings on the zip file to obtain data.

Unzipping the zip file with the password, we see a text file filled with '+' and ','. This is an esolang called reversefuck. On interpreting we get something similar to binary. This is a yet another esolang called binaryfuck. Interpreting we get the flag. To interpret the esolangs use this .

Flag4: For flag4 we are given an image of a keyboard as flag4.jpeg. Steghide isn't able torecover any file from the image. Even bruteforcing with stegcracker returns nothing. We then use another steganography tool for jpegs called jsteg.

These numbers are actually keyboard coordinate cipher. Decoding and making it correct according to the flag format, we have our flag!

Flag5: We are given a flag5.png which seems to be corrupted. A quick check in a hex editor shows us that the image has wrong header. On fixing the header, we are still unable to open the image. Lets check the image using pngcheck.

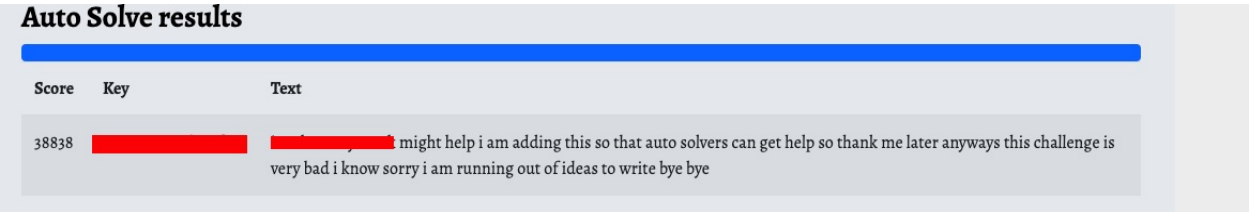


Lets change the ADHD chunk to IHDR. We are still unable to open the file. On checking with pngcheck again,

A quick look at libpng suggests that the chunk DRAT is actually IDAT. On changing the chunk we are finally able to open the image. Steghide can't be used here as its a png file. So we search for alternatives. We find stegolsb. Now we extract data from the image.

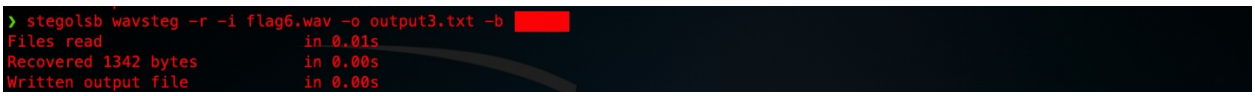
Now we have a text file which contains what looks like chess moves.

On observing closely we see that the image has a hidden pastebin link. On following the link we are presented with a vigenere cipher and a base58 encoded text. Decoding the base58 text we get hint to the key size. Finding an online auto-solver for vigenere takes no time. Setting the key size accordingly we obtain the cracked cipher and a hint to what to do next.



Following on to the site, we find a chess steganography decoder! On decoding without blunders, we get the flag!

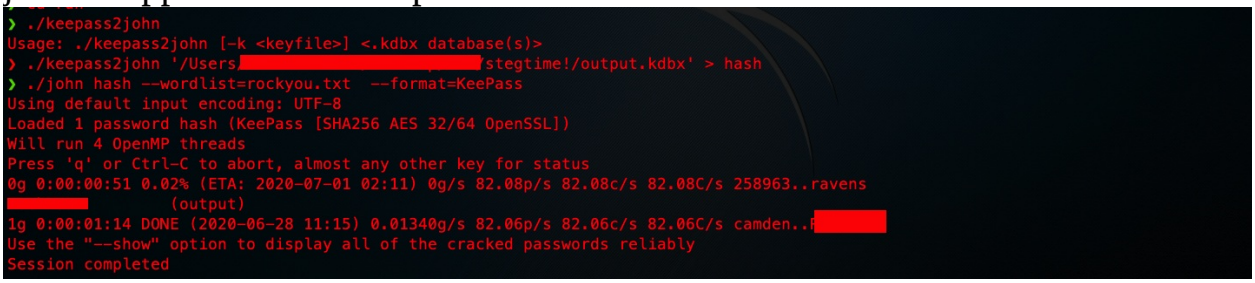
Flag6: For flag we have a wav file. Audio for a change! In the stegolsb tool we just downloaded we find a special subtool called wavsteg. But extracting data demands the number of bytes to be extracted. On hearing the audio file we have our answer! Extracting with given no. bytes we get a file.



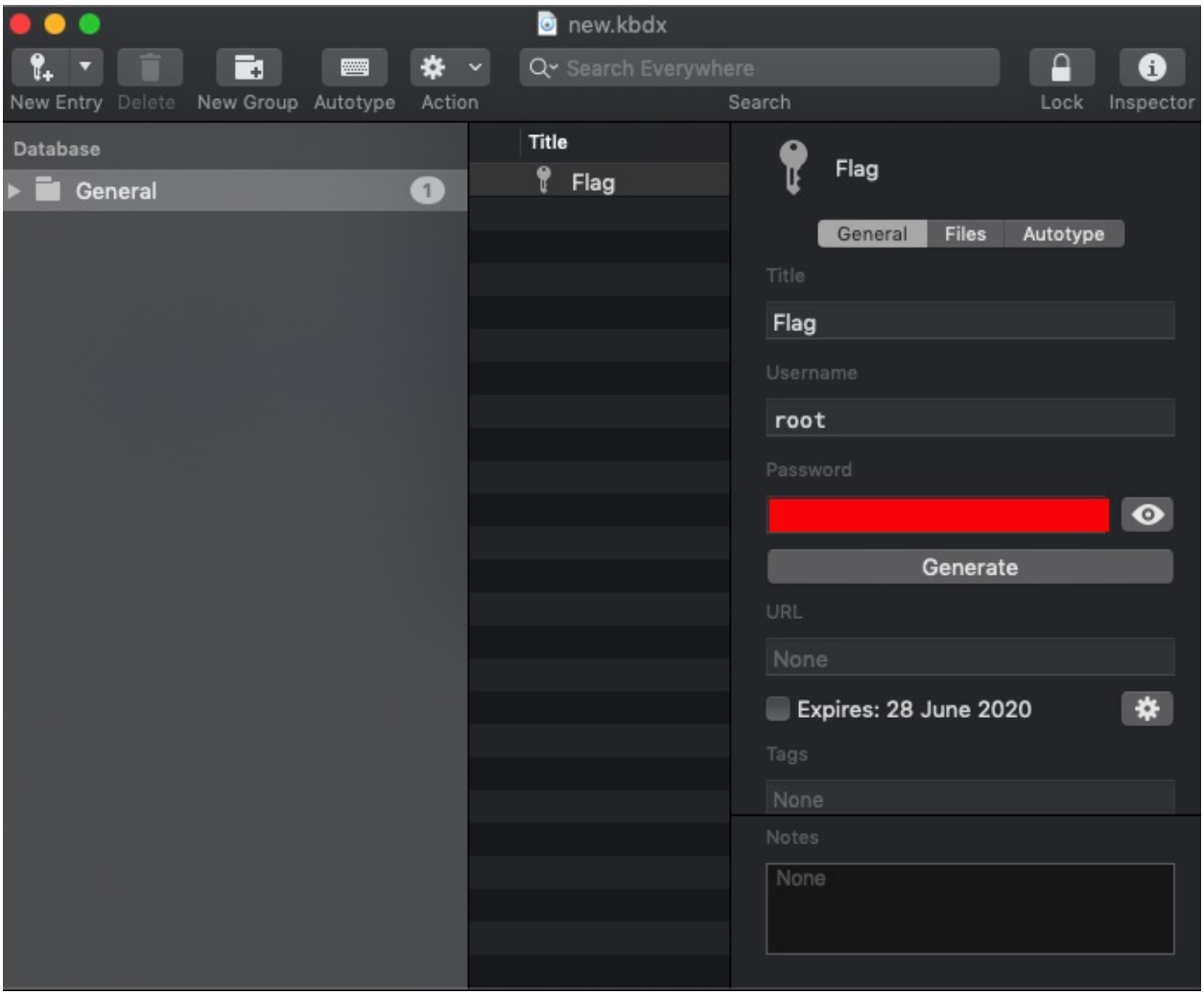
Now on checking the filetype with the file command, we see,

On googling we find that its file extension is .kbdx. Now on further researching we find many applications to open a keepass file. On trying to open with any of

the application we are prompted for a password. Now since we do not know the password, we have to bruteforce! Converting to its hash and bruteforcing with johntheripper we have the password!



Use the password to open the file and find the flag!



And there we have it! We have solved all the 6 challenges!