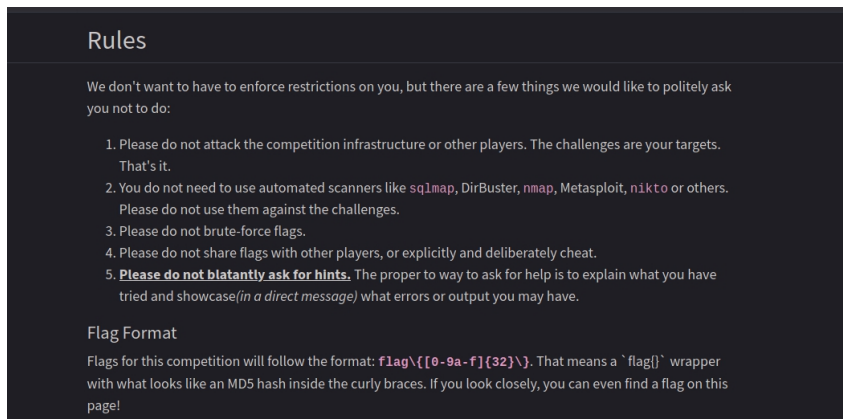


!@@! GrimmCon CTF - Writeup !@@!

(2020)

1. Read The Rules (Warm Up) :

When opened the 'Read the Rules' page it seemed to be an ordinary web page, Nothing Special!!

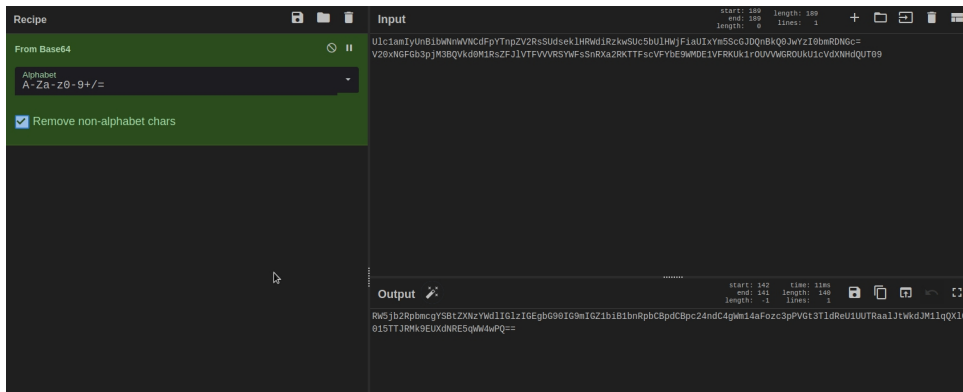


I decided to look the source code of the page and there it was, A Comment which had the valid flag 'flag{90bc54705794a62015369fd8e86e557b}'.

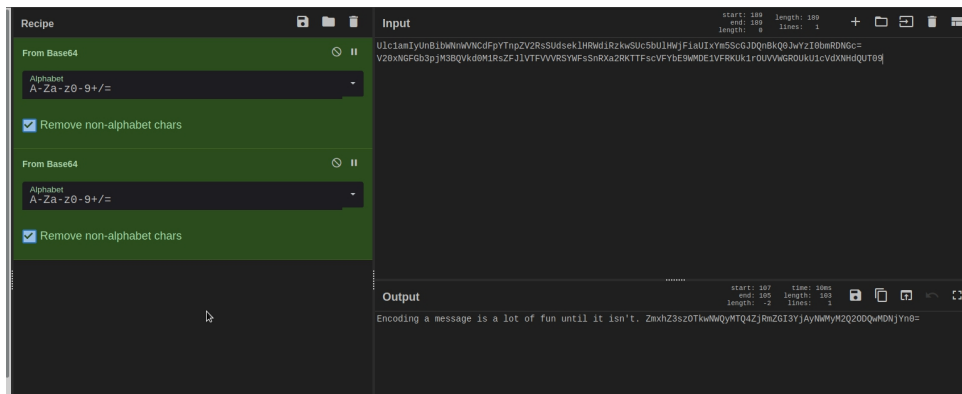
```
challenge authors as noted in the challenge description.
230 </p>
231 <h5 class="mb-2">Prizes</h5>
232 <p>
233 We are pleased to offer prizes to the winners of the GrimmCon 0x3 CTF!
234 </p>
235 <ul>
236 <li><b>1st Place</b> - Drone </li>
237 <li><b>2nd Place</b> - GRIMM iToaster </li>
238 <li><b>3rd Place</b> - GRIMM Backpack full of swag </li>
239 </div>
240 </section>
241 <!-- Thank you for reading the rules! Your flag is: -->
242 <!-- flag{90bc54705794a62015369fd8e86e557b} -->
243
244
245 </div>
246 <!-- /.content-wrapper -->
247
248 <!-- Main Footer -->
249 <footer class="main-footer">
250 <!-- To the right -->
251 <div class="text-center">
252 <strong>Hosted by <a href="https://ctf4hire.com">CTF4Hire</a></strong>
253 </div>
254 <!-- Default to the left -->
255 </footer>
256 </div>
257 <!-- ./wrapper -->
258
```

2. Triple (Easy) :

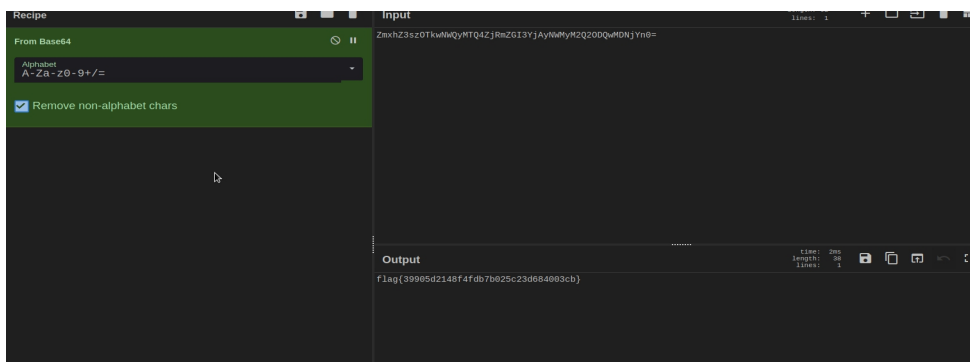
First look at the code "Ulc1amlyUnBibWNNWVNCdFp" suggested that it is base64 encoded. I jumped to CyberChef (SwissKnife for Ecrption/Decryption - "<https://gchq.github.io/CyberChef/> ") and decoded it with base64 recipe.



But it didn't gave the flag. Then I recalled the name 'TRIPLE' which gave me idea to decrypt it 2 more time. Second Decryption gave a sweet message.



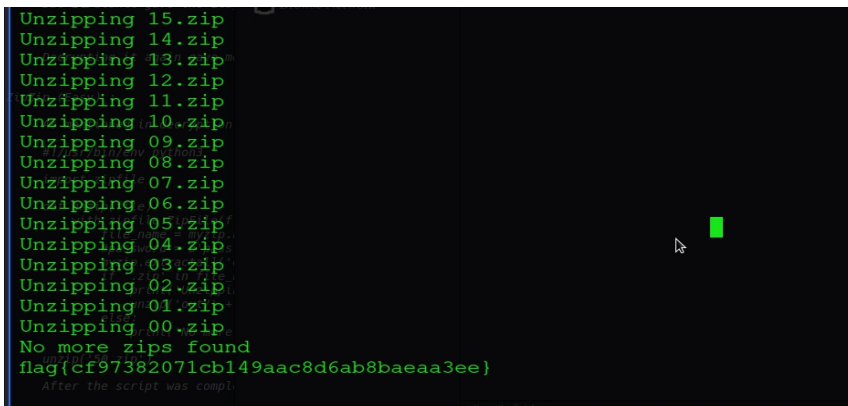
Decrypting it again gave me a valid flag 'flag{39905d2148f4fdb7b025c23d684003cb}'.



3. ZipZip (Easy) :

As mentioned in description file was password encoded multi-ziped folder which had flag in it some where. So Following the description I began unzipping file but it was taking lot of time so I google script to do it quickly and found this python code

```
#!/usr/bin/env python3
import zipfile
def unzip(file):
    with zipfile.ZipFile(file) as myzip:
        file_name = myzip.namelist()[0]
        #password = b'pass'
        myzip.extractall('out', pwd= b'pass')
        if '.zip' in file_name :
            print('Unzipping', file_name)
            unzip('out/' + file_name)
        else:
            print('No more zips found')
unzip('50.zip')
```

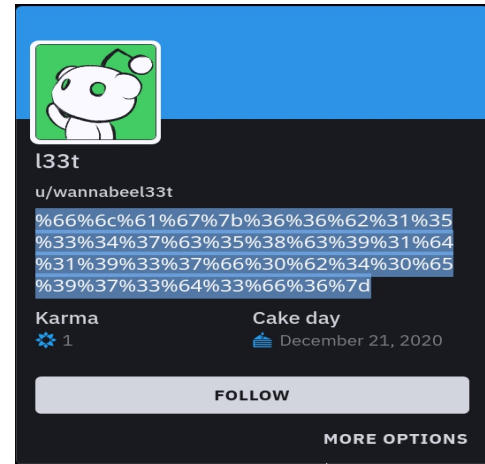
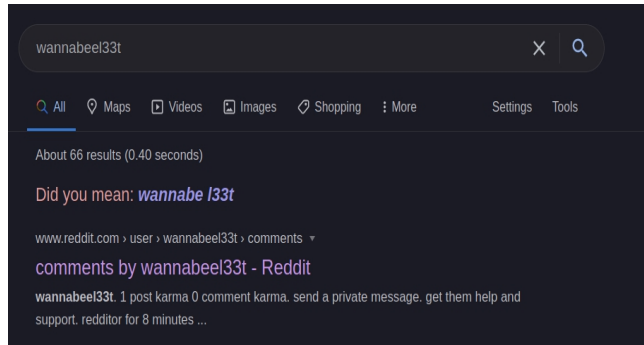


```
Unzipping 15.zip
Unzipping 14.zip
Unzipping 13.zip
Unzipping 12.zip
Unzipping 11.zip
Unzipping 10.zip
Unzipping 09.zip
Unzipping 08.zip
Unzipping 07.zip
Unzipping 06.zip
Unzipping 05.zip
Unzipping 04.zip
Unzipping 03.zip
Unzipping 02.zip
Unzipping 01.zip
Unzipping 00.zip
No more zips found
flag{cf97382071cb149aac8d6ab8baeaa3ee}
After the script was compl.
```

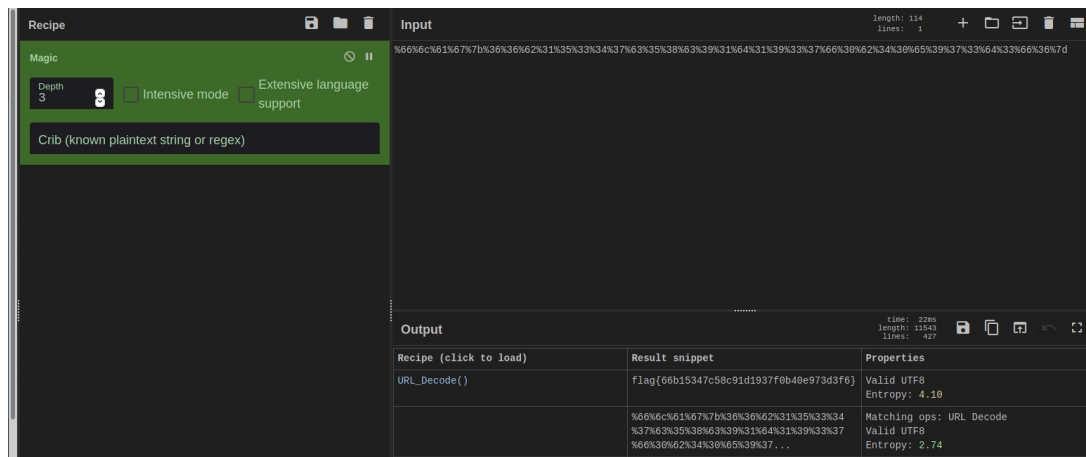
‘# cd’ to the folder which had zip file and run the script using the command ‘# python3 zipzip.py && ./out/flag.txt’. After the script was complete there was a file named flag.txt which had the valid flag ‘flag{cf97382071cb149aac8d6ab8baeaa3ee}’.

4. wannabeel33t (OSINT):

As the type of task suggest OSINT or Information Gathering. I googled the name and found the first link a to reddit account. The Description of the account had some sort of encrypted msg.



I jumped to CyberChef (SwissKnife for Ecrption/Decryption - "<https://gchq.github.io/CyberChef/>") and decoded it with Magic recipe. And it was decoded by URL_decoder and gave me the flag 'flag{66b15347c58c91d1937f0b40e973d3f6}'.



5. Memorandum (Forensics):

Download the zip file and extract the files. It is found that it contains only single binary file. Lets dive in and search for the required flag pattern in the binary.

```
flag s %y, supplied_bin %y
O_TEXT/O_BINARY set in flag s %y
flag s
flag \{e701f9290e2cd553be981461f8ea08e5\}\lang9\fl\par
flag
flag no
flag \{e701f9290e2cd553be981461f8ea08e5\}\lang9\fl\par
flag \{e701f9290e2cd553be981461f8ea08e5\}\lang9\fl\par
__exitflag
__wflag s
^Z
zsh: suspended strings memorandum.bin | grep --color=auto fla
```

use the ‘#strings memorandum.bin | grep flag’ string search command to get output and search for flag. Finally, found the valid flag ‘flag {e701f9290e2cd553be981461f8ea08e5}’.