Write Ups

Panda Tea Cata Panda Challenges

Motivational Message

I. HEX Analysis

This is by far the most fun I've had in a challenge at this CTF, given no more than a simple text file and a dream to unravel the choice, it leads you through a tear jerking journey, in an attempt to find the truth.

This text file isn't really a text file, it contains a lot of unrecgnozied symbols, and led me on a Goosechase, checking the magic bytes (the inital 4 bytes indicating the file type yielded, nothing), sure enough the entire fire was backwards. The magic 4 bytes were actually the bytes for the tail of the file, yet reversed, and sure enough the tailing 4 bytes were backwards, and supposed to be the header.

First thing on the agenda was converting the mot.txt to a hexdump, `hexdump mot.txt > hexdump` , this gives an ideal format, for reading the bytes that we will be flipping.

```
0040c40 3ff4 05cc 3a1d 9f74 fffb 7f80 2ff0 813f
0040c50 2420 8004 1290 4002 0948 2001 0424 9080
0040c60 1712 c5d3 ef07 3e77 cf95 668f dc3d 75c5
0040c70 dd9f 776d fe32 4ae6 9a29 e52e 3207 7265
0040c80 dc45 f6f8 fd6f 4f90 b266 7499 7ae6 7fe6
0040c90 977d f3ce ecbb 7595 b9aa ce9f 7366 ef99
0040ca0 e9ac feec 3bf3 cdfc dfff 5feb 37f3 32ff
0040cb0 d6cf cd97 ef7f 8efb ff64 43eb 8ff5 7ff8
0040cc0 1cc3 0ec9 2fb0 7f9b ff98 7d97 fe6b d77f
0040cd0 3f98 fde7 585f f73f 2ffc 7ff9 fec3 c77f
0040ce0 80ff fdff bfff fde7 ffee fddf fd27 3dff
0040cf0 f378 599f dbf3 9eaf 79bd abd7 3f57 38bc
0040d00 ebfd 75ae 6dd7 fb7b 9d5a e774 f93a fbe7
0040d10 38ce 8ee3 f33d 388e 9be3 24e9 4992 9224
0040d20 4c66 4c92 3299 cc64 cccc 99c9 9299 6666
0040d30 994c 6692 4c66 cccc 3399 6426 6492 24c9
0040d40 24c9 4992 9224 a35f 388e dfe3 e338 388e
0040d50 fbfb ffff fde3 44dd dd0f 5eec 5478 4441
0040d60 0049 0020 3b00 364f 0012 0000 0806 03ad
0040d70 0000 03e8 0000 4452 4948 000d 0000 1a0a
0040d80 0d0a 4e47 8950
0040d86
root@virtualizeddragon:~/Downloads/lab#
```

II. Python Scripting

Now personally, I'd rather write a python script to perform the flipping of bytes myself, rather than using a precompiled tool, supposedly `xdd` can perform this task, I did not have success with it .

```
hex = open('hexdump', 'r')
def reverse(lst):
  return [ele for ele in reversed(lst)]

newarray = []
output = []
for line in hex:
  newarray.append(line)

flip = [];

for row in newarray:
  rowData = row[8:]
  flip.append(rowData.replace('\n','').split(' '))
i = len(flip)-1

while i > 1:
  output.append(reverse(flip[i]))
  i = i - 1



f = open("output.png", "w")
for line in output:
  f.write(' '.join(line) + " ")
  print(' '.join(line))
```

To be fair this is very jenky code as I wrote it in steps, rather than algorithmically, yet it works enough for this CTF.

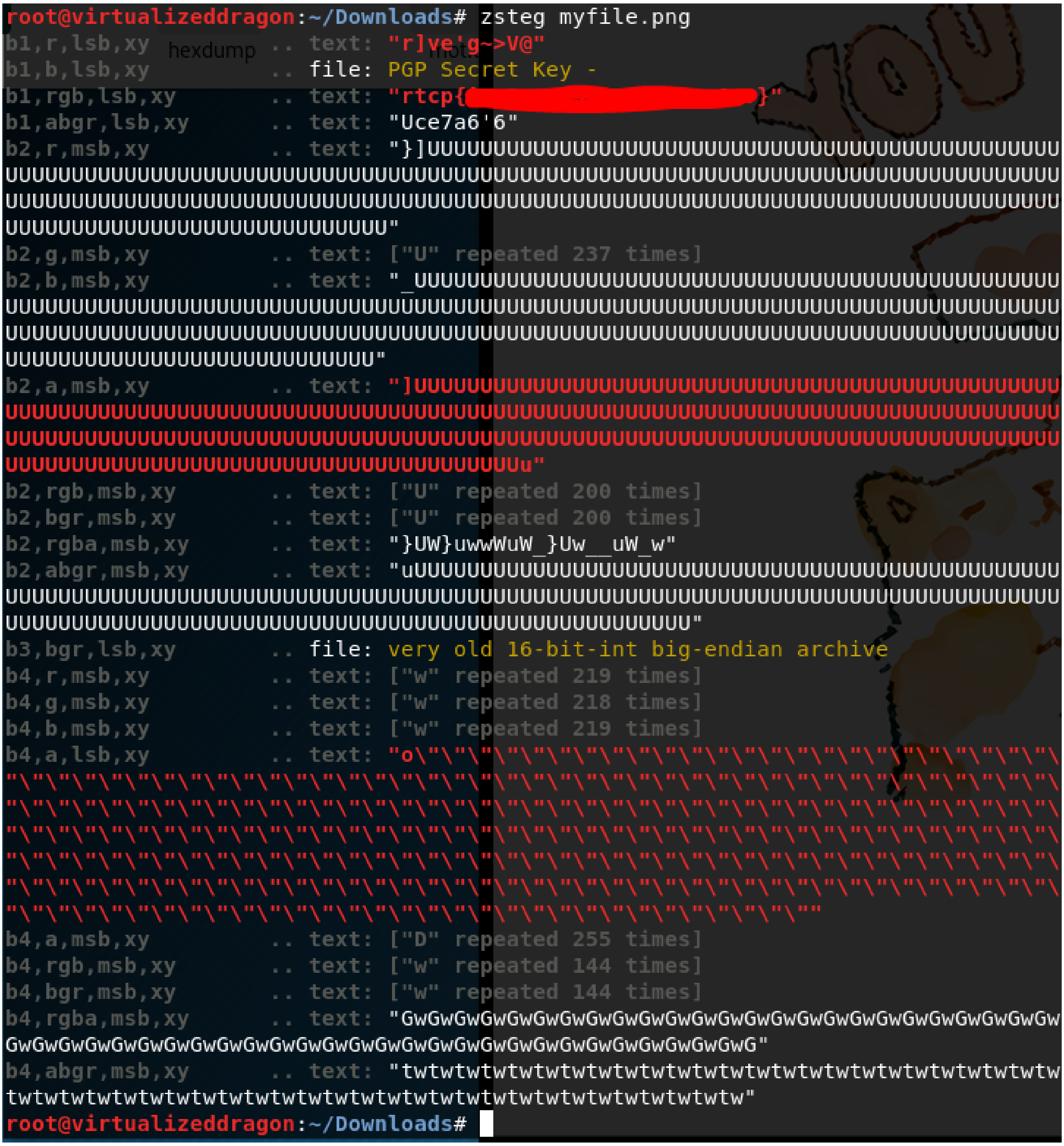
- First import hexdump.
- Second create a single long list of all lines in that dump.
- Third, remove memory addresses as shown in the first 8 bytes above.
- Fourth flip the lines, convert them to an array split by spaces
- Fifth iterate through the list backwards and reverse the list.
- Sixth pipe it to an output file, as it is hex and not an actual file yet.
  `python reverse.py > file`

```
6601 02ba cdd5 a16d 0820 8000 0208 2080
0002 0820 8000 02ce 0810 d09d 2935 1d45
0001 0410 4000 0104 1040 0001 0410 b059
8080 6e73 7568 1b02 0820 8000 0208 2080
0002 0820 8080 3302 0474 674a 4d47 1140
0001 0410 4000 0104 1040 0001 046c 1620
a0db 5c1d da86 0002 0820 8000 0208 2080
0002 0820 e08c 0001 dd99 52d3 5104 1040
0001 0410 4000 0104 1040 0001 9b05 08e8
3657 87b6 2180 0002 0820 8000 0208 2080
0002 0838 23e0 05f4 6949 ff9f a4ff df99
6ed3 5104 1040 0001 0410 4000 0104 1040
```

Which should output a file similar to this, running it through an online hex to file converter shows you this beautiful image.

III. File Analysis

This PNG still contains the encrypted flag, it's not decipherable via strings, nor exiftool, nor binwalk, yet using `zsteg [filename.png]` returns the following:



Growls at the Chicken

Given a RSA public, private, and encoded message the task is clear. Decode, and proceed. After decoding the message, a url is retrieved this URL shows a website containing this GIF.

Showing this GIF, which I have to take a moment to absolutely honor the perfection of such a isometric masterpiece.

Nonetheless, some private codes are on the page. Too lazy to decrypt this manually I figured I could utilize Javascript in some way.

```
numbers = "9 20 30 15 16 5 14 19 30 27 29 8 20 13 12 28"
key = "abcdefghijklmnopqrstuvwxyz[]. "
numbers = numbers.split(" ")
nString=""

for(i=0;i<numbers.length;i++){
  nString= nString + key[numbers[i]-1]
}

console.log(nString)

```

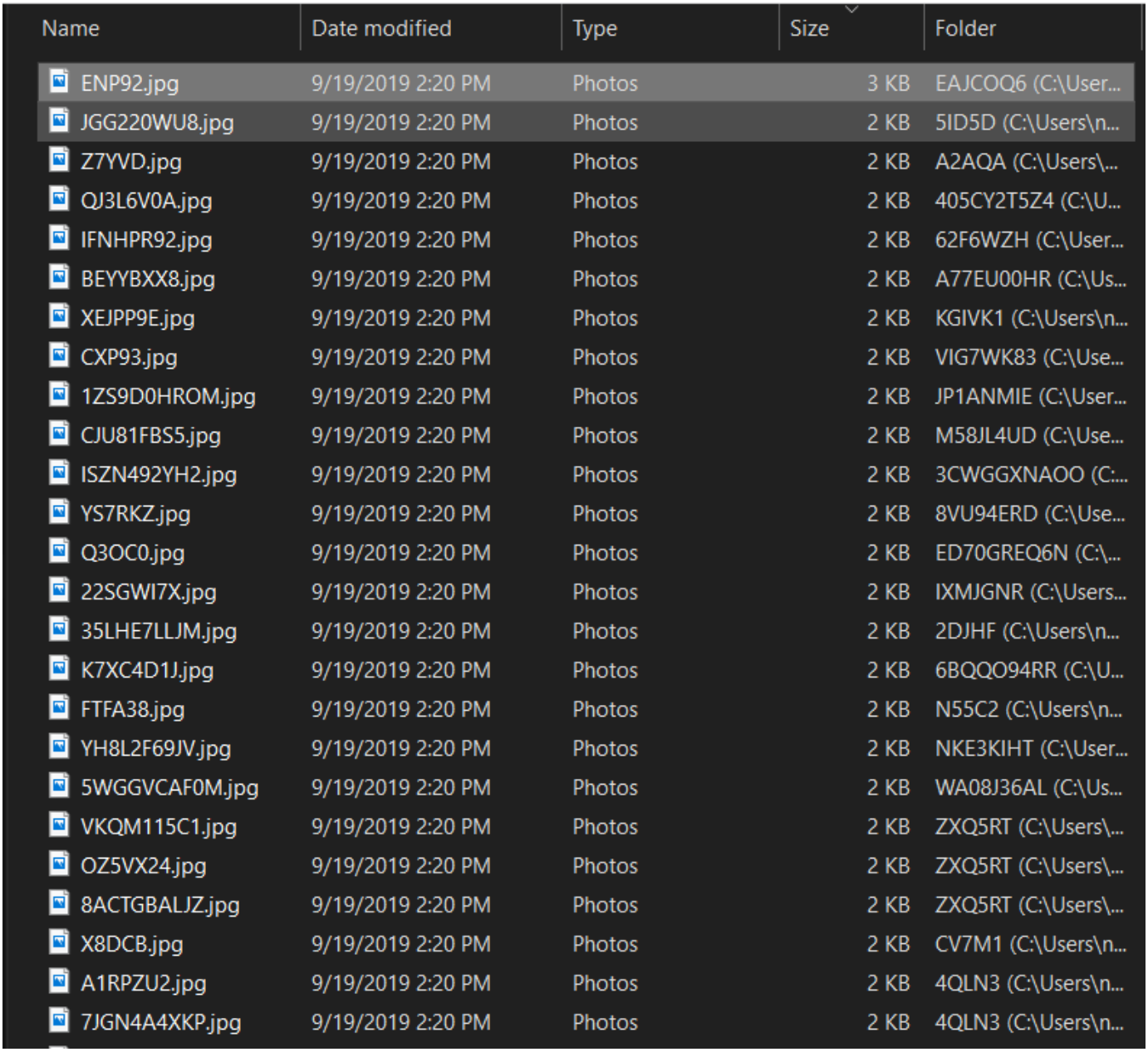
> it opens [.html]

This cryptic message is referring to the thing that is animated and “opening” in the gif, ***.html***, which is another webpage containing the key.

TREE

Name	Size	Packed Size	Modified	Attributes	CRC	Encrypted	Method	Block	Folders	Files
P24AKQCA	412 046	0	2019-09-19...	D	CCD06F2E	-			5 429	257
Z9FVS4ZTM	393 720	0	2019-09-19...	D	518AB679	-			4 842	243
NUD67JGK	343 840	0	2019-09-19...	D	B5910079	-			4 455	215
GT9HA	341 958	0	2019-09-19...	D	CB5B235A	-			4 648	213
DCTT2Z2K3	268 480	0	2019-09-19...	D	E019E4B9	-			3 433	167
CV7M1	111 602	0	2019-09-19...	D	6858B757	-			1 373	70
GOY3BFAA	80 128	0	2019-09-19...	D	37026F4E	-			1 055	50
573CB5	78 574	0	2019-09-19...	D	F6C0B961	-			910	50
H028UQCY	56 578	0	2019-09-19...	D	544D36A2	-			834	35
G7EA2OH	14 796	0	2019-09-19...	D	A3903575	-			169	9
3JXDFIK	14 130	0	2019-09-19...	D	88065534	-			187	9
03VJ7	12 856	5 789	2019-09-19...	D	85267684	-			164	8
W6YAVH	9 642	0	2019-09-19...	D	63DCD8E3	-			193	6
ARMGRGU7C	3 214	0	2019-09-19...	D	21499DA1	-			4	2
P4D7TBQ	1 718	0	2019-09-19...	D	153BC986	-			26	1
UW085JWRA	1 496	0	2019-09-19...	D	0C0DD41B	-			25	1
VRYWVT	1 496	0	2019-09-19...	D	0C0DD41B	-			34	1
6JSS1X	0	0	2019-09-19...	D	00000000	-			0	0
BS2AS1WFN	0	0	2019-09-19...	D	00000000	-			27	0
FV247GJ8F	0	0	2019-09-19...	D	00000000	-			0	0
IRGTPG	0	0	2019-09-19...	D	00000000	-			0	0
L3JXCSZ	0	0	2019-09-19...	D	00000000	-			5	0
MYIVDIW8MR	0	0	2019-09-19...	D	00000000	-			4	0
ORV76	0	0	2019-09-19...	D	00000000	-			0	0
TNXNVXLCAV	0	0	2019-09-19...	D	00000000	-			0	0
XOCGL	0	0	2019-09-19...	D	00000000	-			4	0

Tree was a fun challenge, it was literally a file tree, one filled with several images, and sure enough you needed to find the correct one. At first I took the approach that Linux command line would easily sift through this tree and eliminate any folders, unsure of the specific command-line I needed, I tried `find` with some parameters, eventually I gave up and decided to finish it later. I was staring at the tree, in Windows in 7-zip, and remembered the search bar in Windows Explorer, typed in "`.jpg`" and sure enough, there were all the files. I finished this challenge by sorting by size to easily find the photos that were different, as there were many duplicates, with varying file names, and I was preparing to copy all these files to a folder, and use Python to delete the ones with same hashes, but thankfully it didn't resort to that.



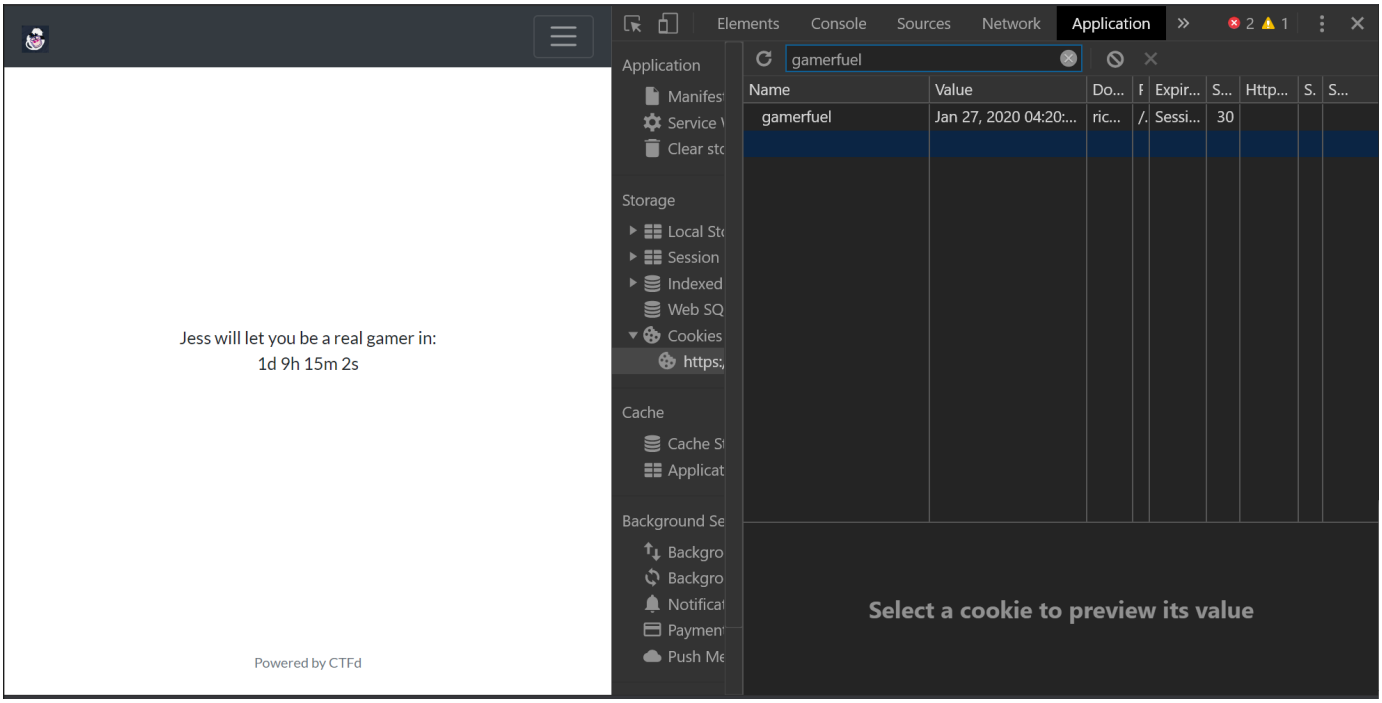
Name	Date modified	Type	Size	Folder
ENP92.jpg	9/19/2019 2:20 PM	Photos	3 KB	EAJCOQ6 (C:\User...
JGG220WU8.jpg	9/19/2019 2:20 PM	Photos	2 KB	5ID5D (C:\Users\n...
Z7YVD.jpg	9/19/2019 2:20 PM	Photos	2 KB	A2AQA (C:\Users\...
QJ3L6V0A.jpg	9/19/2019 2:20 PM	Photos	2 KB	405CY2T5Z4 (C:\U...
IFNHPR92.jpg	9/19/2019 2:20 PM	Photos	2 KB	62F6WZH (C:\User...
BEYYBXX8.jpg	9/19/2019 2:20 PM	Photos	2 KB	A77EU00HR (C:\Us...
XEJPP9E.jpg	9/19/2019 2:20 PM	Photos	2 KB	KGIVK1 (C:\Users\n...
CXP93.jpg	9/19/2019 2:20 PM	Photos	2 KB	VIG7WK83 (C:\Use...
1ZS9D0HROM.jpg	9/19/2019 2:20 PM	Photos	2 KB	JP1ANMIE (C:\User...
CJU81FBS5.jpg	9/19/2019 2:20 PM	Photos	2 KB	M58JL4UD (C:\Use...
ISZN492YH2.jpg	9/19/2019 2:20 PM	Photos	2 KB	3CWGGXNAOO (C:...
YS7RKZ.jpg	9/19/2019 2:20 PM	Photos	2 KB	8VU94ERD (C:\Use...
Q3OC0.jpg	9/19/2019 2:20 PM	Photos	2 KB	ED70GREQ6N (C:\...
22SGWI7X.jpg	9/19/2019 2:20 PM	Photos	2 KB	IXMJGNR (C:\Users...
35LHE7LLJM.jpg	9/19/2019 2:20 PM	Photos	2 KB	2DJHF (C:\Users\n...
K7XC4D1J.jpg	9/19/2019 2:20 PM	Photos	2 KB	6BQQO94RR (C:\U...
FTFA38.jpg	9/19/2019 2:20 PM	Photos	2 KB	N55C2 (C:\Users\n...
YH8L2F69JV.jpg	9/19/2019 2:20 PM	Photos	2 KB	NKE3KIHT (C:\User...
5WGGVCAF0M.jpg	9/19/2019 2:20 PM	Photos	2 KB	WA08J36AL (C:\Us...
VKQM115C1.jpg	9/19/2019 2:20 PM	Photos	2 KB	ZXQ5RT (C:\Users\...
OZ5VX24.jpg	9/19/2019 2:20 PM	Photos	2 KB	ZXQ5RT (C:\Users\...
8ACTGBALJZ.jpg	9/19/2019 2:20 PM	Photos	2 KB	ZXQ5RT (C:\Users\...
X8DCB.jpg	9/19/2019 2:20 PM	Photos	2 KB	CV7M1 (C:\Users\n...
A1RPZU2.jpg	9/19/2019 2:20 PM	Photos	2 KB	4QLN3 (C:\Users\n...
7JGN4A4XKP.jpg	9/19/2019 2:20 PM	Photos	2 KB	4QLN3 (C:\Users\n...

Robots

```
User-agent: *
Disallow:
/robot-nurses
/flag
```

After accessing /robots.txt it returns this file. The obvious solution would be to check `/flag`, and unfortunately that page returns a beautiful video of the one and only Rick Astley, going back and checking `/robot-nurses` gives a page, containing an actual flag.

No Sleep



No Sleep was a challenge that brought you to a page that you were unable to access until 4 days after you open (which is after the challenge is over). Needless to say, cookie manipulation is one of my specialties, decrementing the date by 2-3 days, did the trick, and flashed the flag at me.

Phishing for Flags

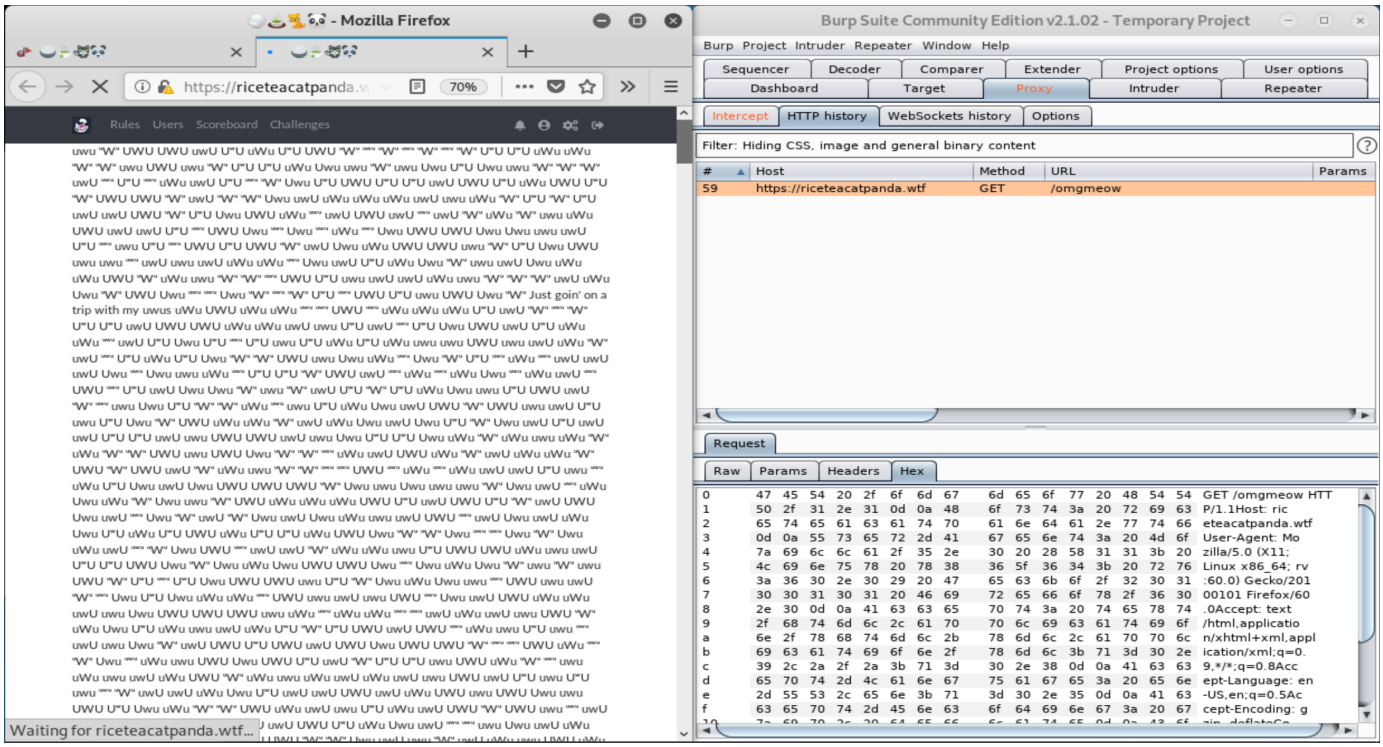
This was a zip of emails that you had to analyze to discover the key. It was pretty basic, look for something unusual, and as the hint says, “some are more interesting than others.” Needless to say a file named “GIVE ME BACK MY EYE HOLES”, caught my eye , which is hardly surprising given it’s literally the strangest thing I’ve ever read.

```
MIME-Version: 1.0
Date: Mon, 18 Nov 2019 10:31:11 -0800
Message-ID: <CAHbBcduwrSDvJ2DWCg8XiH31+rkoLgEEQuEpxG-2=vMsjd0uyA@mail.gmail.
Subject: GIVE ME BACK MY EYEHOLES
From: Eyehole Man <eyeholeman@holez.org>
To: Jerry Smith <jerrysmith@james.com>
Content-Type: multipart/alternative; boundary="000000000000f5b3a00597a3260d"

--000000000000f5b3a00597a3260d
Content-Type: text/plain; charset="UTF-8"

I'm the eyeho-hole man! I'm the only one who's allowed to have eyeholes!

Get up on out of here with my Eyeholes!

GIVE ME BACK M-MY EYEHOLES
<https://riceteacatpanda.wtf/phishingemail>

--000000000000f5b3a00597a3260d
Content-Type: text/html; charset="UTF-8"
Content-Transfer-Encoding: quoted-printable

<div dir=3D"ltr"><div>I&#39;m the eyeho-hole man! I&#39;m the only one who&=
#39;s allowed to have eyeholes!=C2=A0<br></div><div><br></div><div>Get up o=
n out of here with my Eyeholes!<br></div><div><br></div><div>GIVE ME BACK M=
-MY <a href=3D"https://riceteacatpanda.wtf/phishingemail">EYEHOLES</a></div=
></div>
```

Regardless, it has an interesting link on there, which happens to correlate with the challenge, so might as well give that one a try.

Uwu?

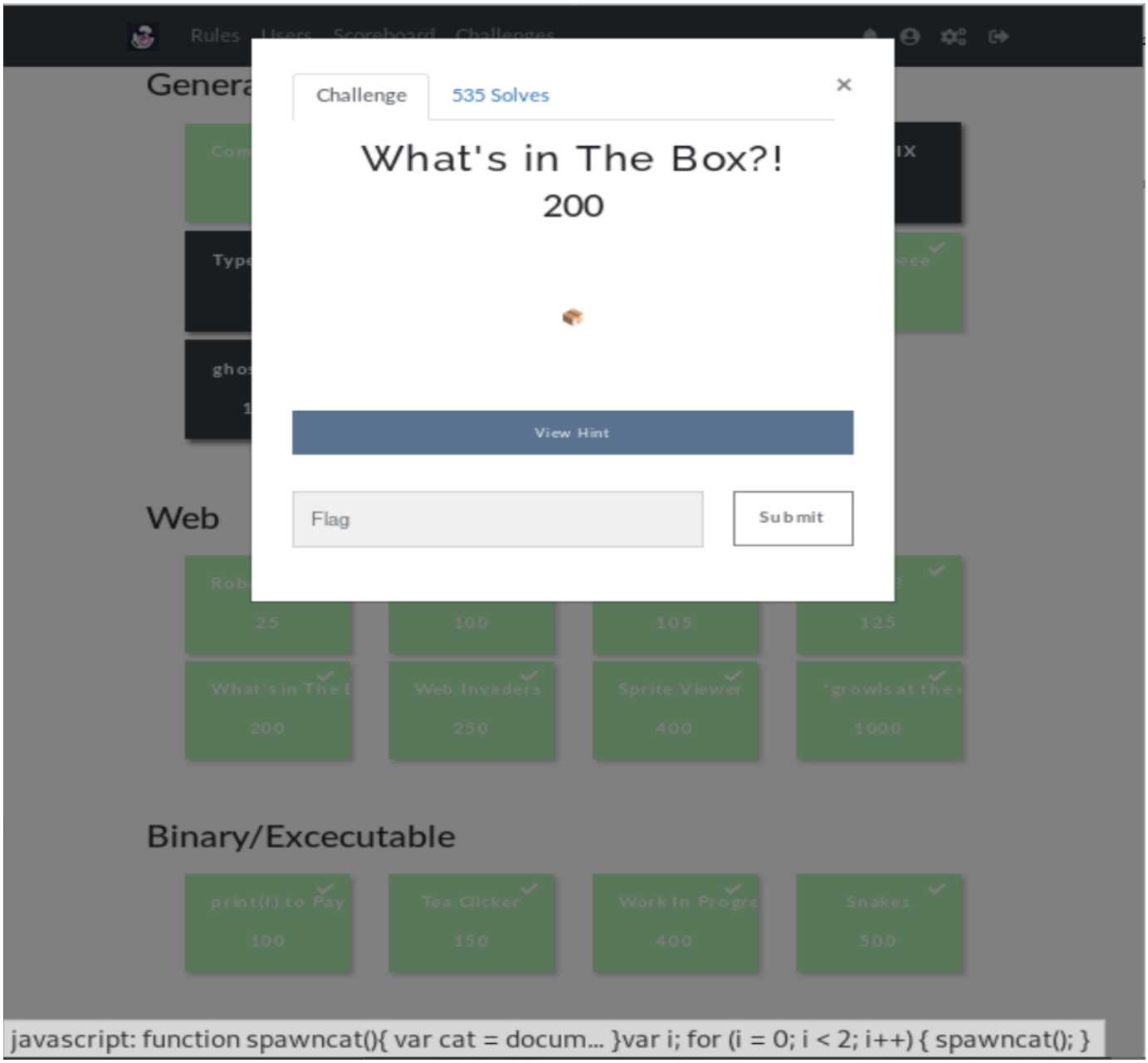
This challenge took me much much much longer than I thought it should, probably because I wasn't looking at the URL. Essentially it was a series of eight silent redirects. Fiveentire pages filled with no other than UwUs.



#	Host	Method	URL	Params
59	https://riceteacatpanda.wtf	GET	/omgmeow	
62	https://riceteacatpanda.wtf	GET	/pandaaaaaaa	
64	https://riceteacatpanda.wtf	GET	/you-better-wash-your-rice	
73	https://riceteacatpanda.wtf	GET	/footprint	
75	https://riceteacatpanda.wtf	GET	/uwustorage	

By analysing this traffic, and the intercepted HTML pages, I was able to find the key in one of the pages listed above.

What's in the Box?!



When hovering over the box, you can clearly see some Javascript onclick action, eagerly awaiting your input. Using inspect element, and an online Javascript prettifier I was able to convert it back to.

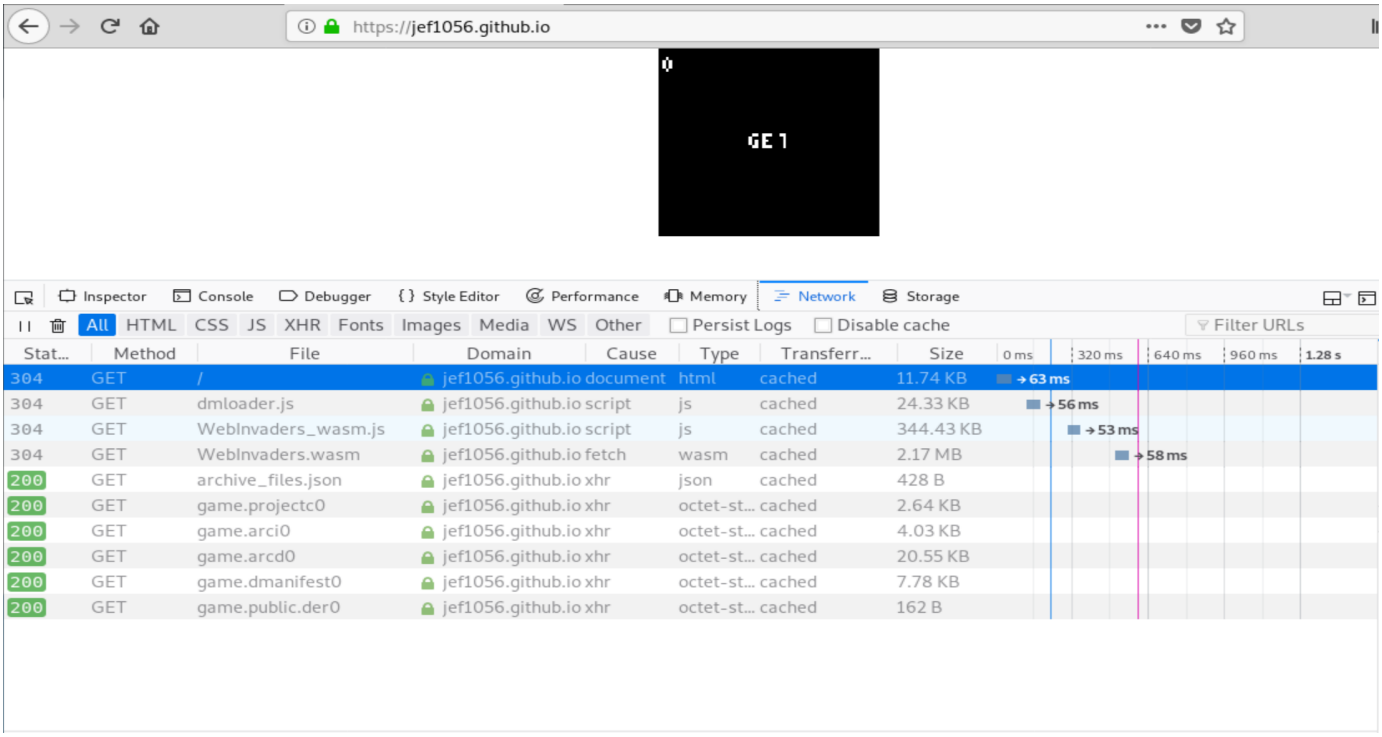
```
function spawncat() {
    var cat = document.createElement("IMG");
    var cat_right = "
    var cat_left = "
    var cat_idle = "
    cat.setAttribute("src", cat_idle);
    var x = Math.random() * (window.innerWidth - 64);
    var y = Math.random() * (document.documentElement.scrollHeight - 64);
    var dx = 0;
    var dy = 0;
    var onGround = true; /*rtcp{*/
    var ticks = 0;
    var jump_t = 20;
    var move_t = 180;
    cat.style.cssFloat = "left";
    cat.style.position = "absolute";
    cat.style.width = "64px";
    cat.style.height = "64px";
    cat.style.left = 0;
    cat.style.up = 0;
    cat.style.zIndex = 100000000;
```

Which shows the beginning of the flag, within a comment, by concatenating further comments in this file, I was able to stringify the flag.

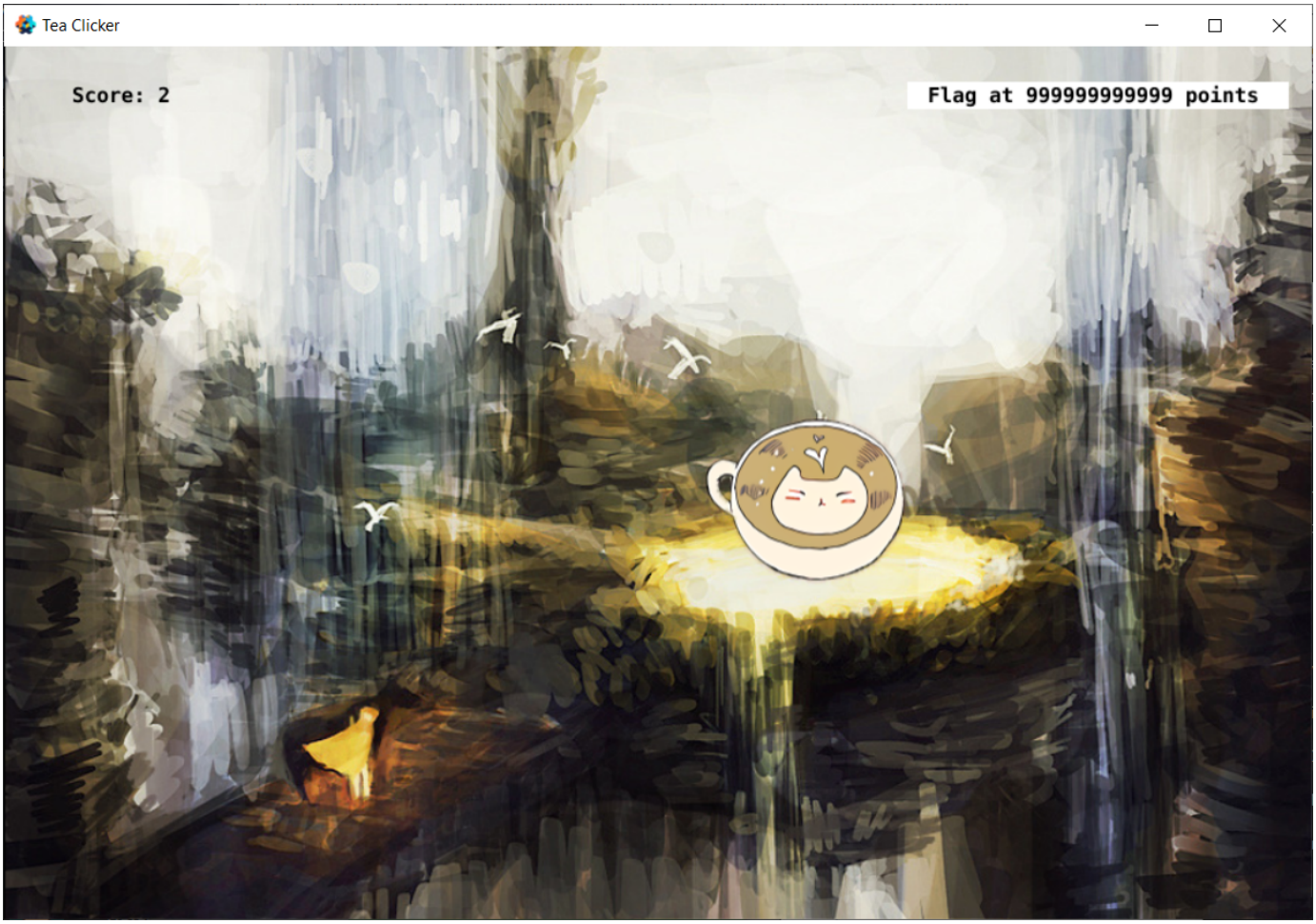
Web Invaders

```
<iframe src="https://jef1056.github.io/" height="300" width="300">
  #document
    <!DOCTYPE html>
    <html lang="en">…</html> ev
</iframe>
```

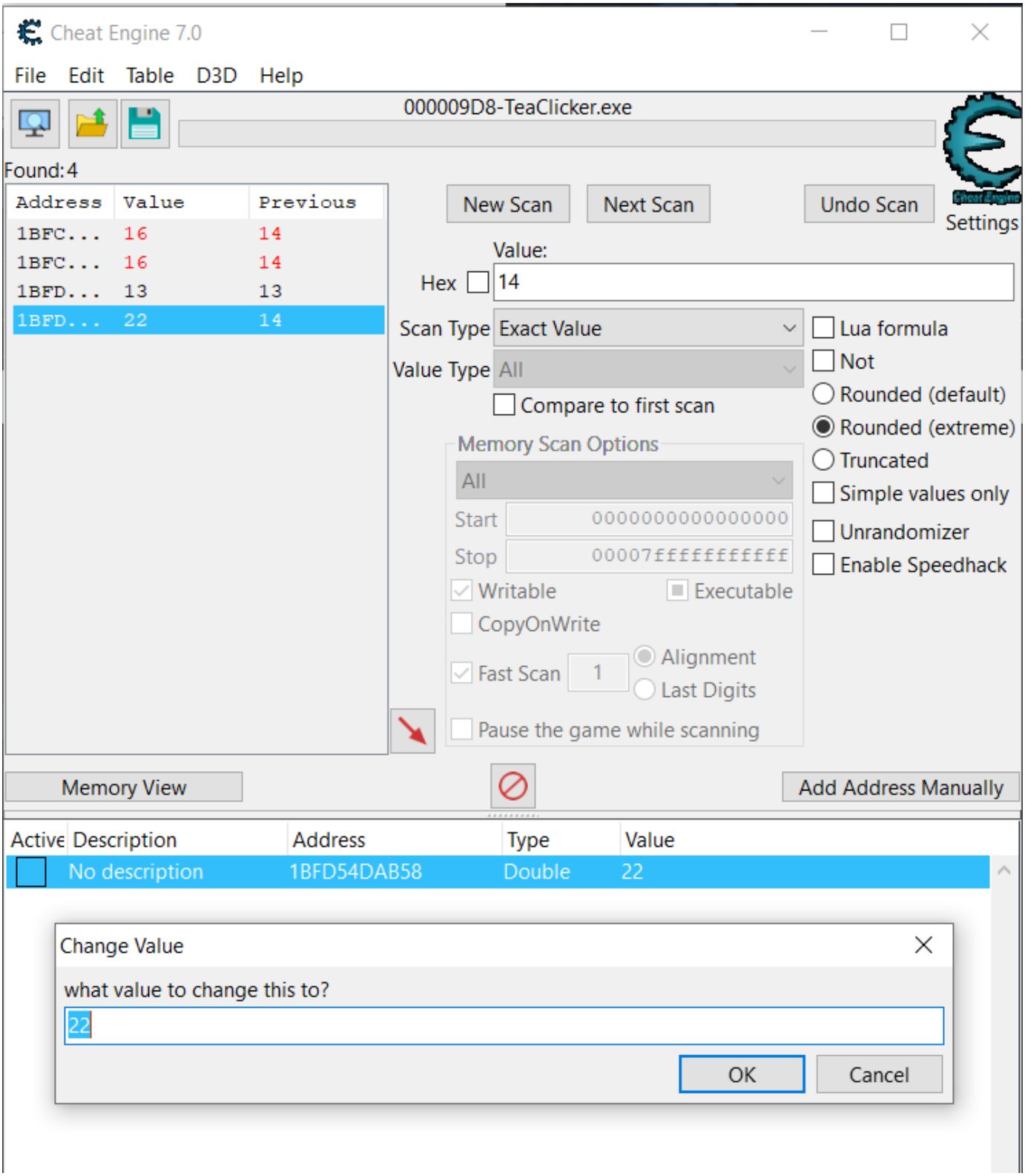
A JavaScript space invaders game was embedded to an iframe, by gathering it's originating link, and analyzing the source of the game, you'll find some interesting files, briefly analyzing them allows you to rather quickly find the flag.



Tea Clicker



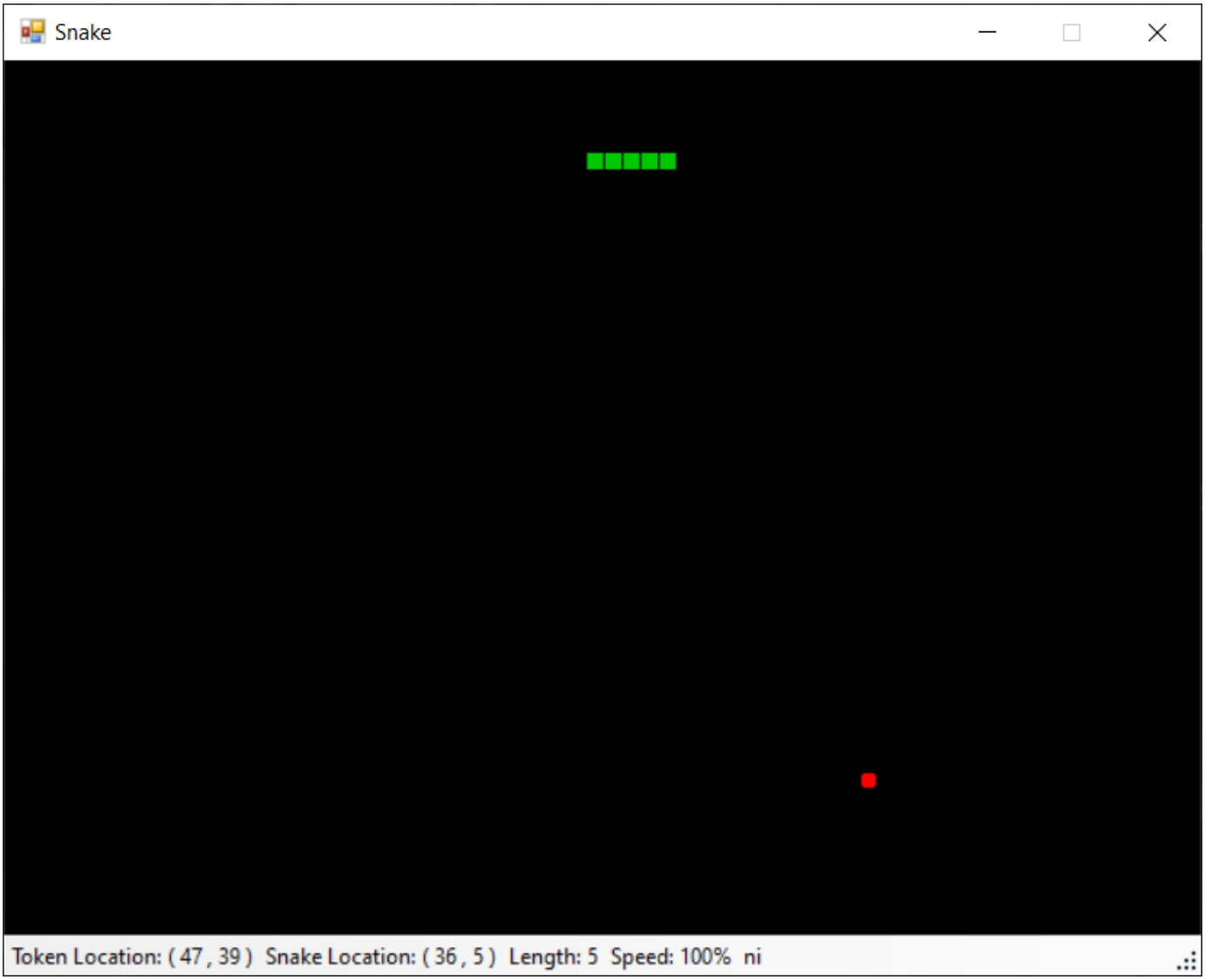
The classic clicker game Tea clicker, unfortunately I do not want to click that much, and thankfully we don't have to after a few memory scans.



Work in Progress

By glitching onto this block, we can scan for a change in memory (like the previous challenge), and then by falling we can scan for a decreased value, this takes a couple scans but eventually gives us 200 values, by trial and error we find the one that could be used to lock our character at any Y-coordinate, by manipulating this value out of expected bounds, a flag is revealed.

Snakes



The theme is the same as the last two challenges, scan your length and then when you're down to a single result, increment it expontentially until a flag reveals. Unfortunately I have found this to cause an error and crash the program, but they didn't make `{alt + prnt screen}` screenshots for nothing. Open up MSPaint.exe, or whatever and then paste your screenshot in, so that you can easily, type it in, without rushing yourself, given it's insane convoluted LEET-code.

BTS-Crazed

A beautifully crafted song, that is so good that you actually forget it's a challenge, with that being said. It unfortunately is a challenge, so listen to it maybe once, twice, or five times, boot up your hexeditor, or Kali system, `strings save.mp3 | grep "rtcp{"` , and it should return something pretty immediately.

Allergic College Application

The hint for this states that the original document was written in Mandarin, with that being said it's obvious the text encoding is wrong, and so the only solution is to open it in a software that will automatically choose an encoder, in this case Notepad ++ was the only tool I could find that did the job justice.

BASmati ricE 64

This image of a nice bowl of rice, contains some secret data, and given it's in the forensics category, it probably uses a command that incorporates Steg, or exifTool. In this case running steghide `extract -sf [filename]` , returns a interesting text file, `cat [filename]` returns some characters but not all of them, opening the file in GEdit or something similar to where all the characters are visible allows it to be decoded from Base64, which reveals the true flag.