

Writeup CTF GEMING 2023

HAHA HOHO AWIKWOK GG GEMING



Excy
Maskirovka
Kisanak

Powered by:



Table of Content

Table of Content

WEBEX

LoG1n

Flag = JCTF2023{s0m3_h34deR_&_c00k1e_4re_us3fu1_r1ght?}

Web of the Gods

Flag = JCTF2023{t4kAr4pUt0_P0p0ruN64_p1R1TOP4R0}

Vision

Flag = JCTF2023{s0_e4sy_w3b_3xPI0itation}

NWORDPASS

Flag = JCTF2023{h3y_k1ds_d0nt_b3_r4c1st}

REVENG

For You

Flag = JCTF2023{w3_just_engin33red_th1s_One_4_you}

BINEX

CRYPTO

Easy CBC

Flag = JCTF2023{n4rim0_in9_pAndum}

Rumah Sakit Akademik UGM

Flag = JCTF2023{d0nt_r3us3_y0ur_pr1m3s_4g41n_4nd_4g41n}

FORENSIC

Dinosaur

Flag = JCTF2023{the_st364n0s4uru5_likes_b10wf15h}

File Smuggling

Flag = JCTF2023{YOUGOTIT}

Spartan Ghosts

Flag = JCTF2023{dream_on_kratos}

MISCEL

Mega Sus

Flag = JCTF2023{rEALLysusXDD}

Strange Message

Flag = JCTF2023{0h_n0o0_u_g0t_m3:_(_c0ngr4t5!}

Feedback

Flag = JCTF{thanks_for_filling_this_feedback}

OSINT

WhereIsThis

Flag = JCTF2023{69FC+8V_TERBAN}

WEBEX

LoG1n

Challenge

26 Solves

×

LoG1n

300

Jota created a website but he forgot the password. However, he remembers that he can edit something from the client side so he can login in the admin area. Even so, he also remembered that to enter the admin area there is also one-way encryption that must be passed. Can you help him?

Author: BROP #9678

34.101.234.148:8499

Submit

Pada soal ini, diberikan sebuah website yang memiliki tampilan seperti berikut.

Login

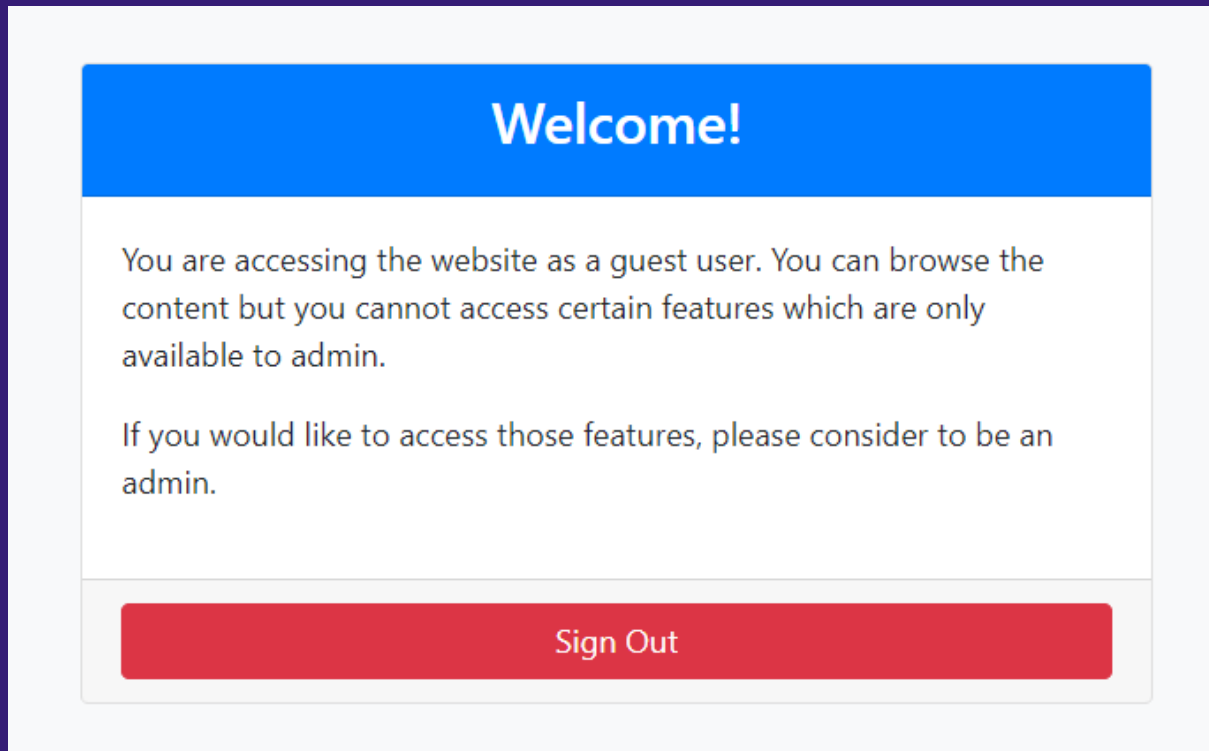
Username

Password

Login

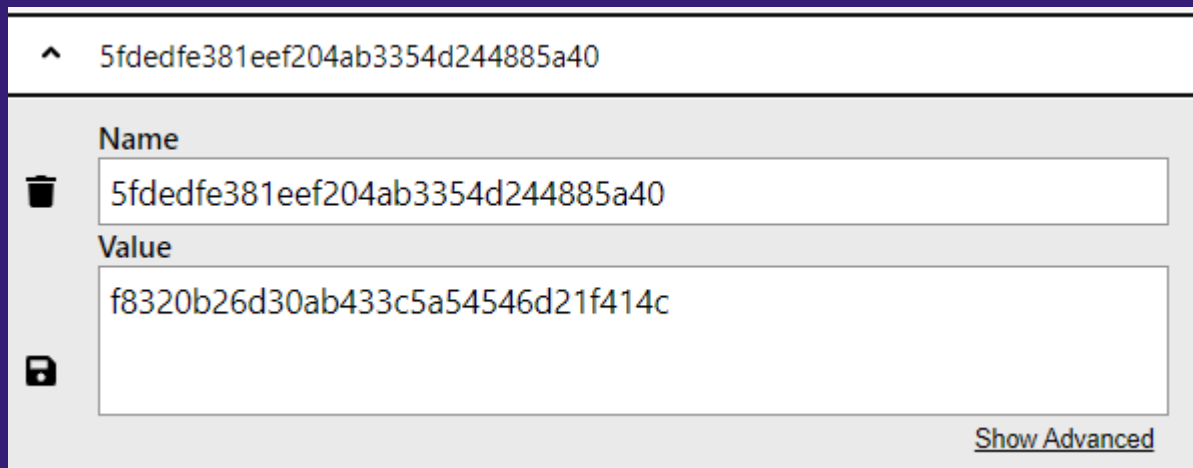
Continue as Guest

Kami tidak menemukan kerentanan apapun pada bagian form login sehingga kami memutuskan untuk melakukan testing pada fitur yang lain yakni fitur “continue as guest”. Jika kita pilih fitur tersebut, maka tampilannya akan menjadi seperti ini.



Menarik, kita sudah berhasil login. Namun, perlu diperhatikan bahwa kita berhasil login tidaklah sebagai admin melainkan sebagai guest biasa. Disini kami mencoba untuk melakukan privilege escalation dari guest menjadi admin.

Untuk melakukan hal tersebut, kita bisa terlebih dahulu melakukan pengecekan pada cookie yang digunakan.



HAHA HOHO AWIKWOK GG GEMING

Ternyata, terdapat sebuah cookie yang menurut kami memiliki nama cookie dan juga value yang unik. Kami menyadari bahwa penamaan tersebut bukanlah penamaan biasa melainkan sebuah hashing (terlihat dari panjang dan juga kombinasi karakter yang digunakan). Langsung saja kita coba pecahkan dulu menggunakan crackstation.

Enter up to 20 non-salted hashes, one per line:

5fdedfe381eef204ab3354d244885a40
f8320b26d30ab433c5a54546d21f414c

I'm not a robot

reCAPTCHA
Privacy - Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripemd160, whirlpool, MySQL 4.1+ (sha1 sha1_bin), QubesV3.1BackupDefaults

Hash	Type	Result
5fdedfe381eef204ab3354d244885a40	md5	isAdmin
f8320b26d30ab433c5a54546d21f414c	md5	False

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

Ternyata keduanya memiliki arti “isAdmin” dan “False”, dari sini kami langsung berpikiran untuk mencoba melakukan tampering pada value dari cookie tersebut menjadi “True”. Kita bisa menggunakan tools-tools online yang ada di internet seperti cyberchef atau semacamnya untuk melakukan hal tersebut.

Recipe

MD5

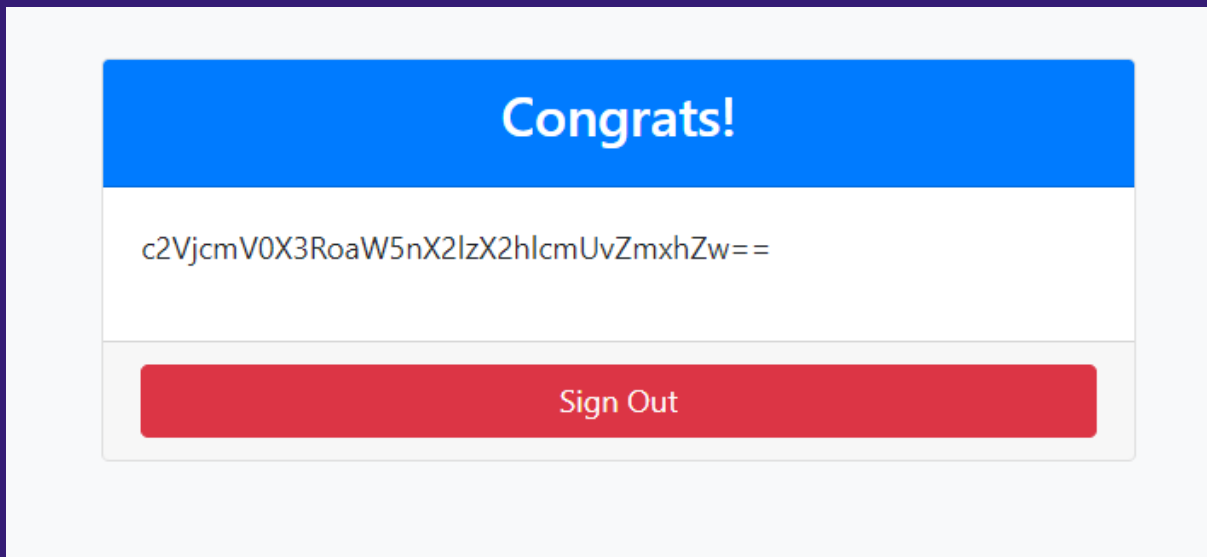
Input

True

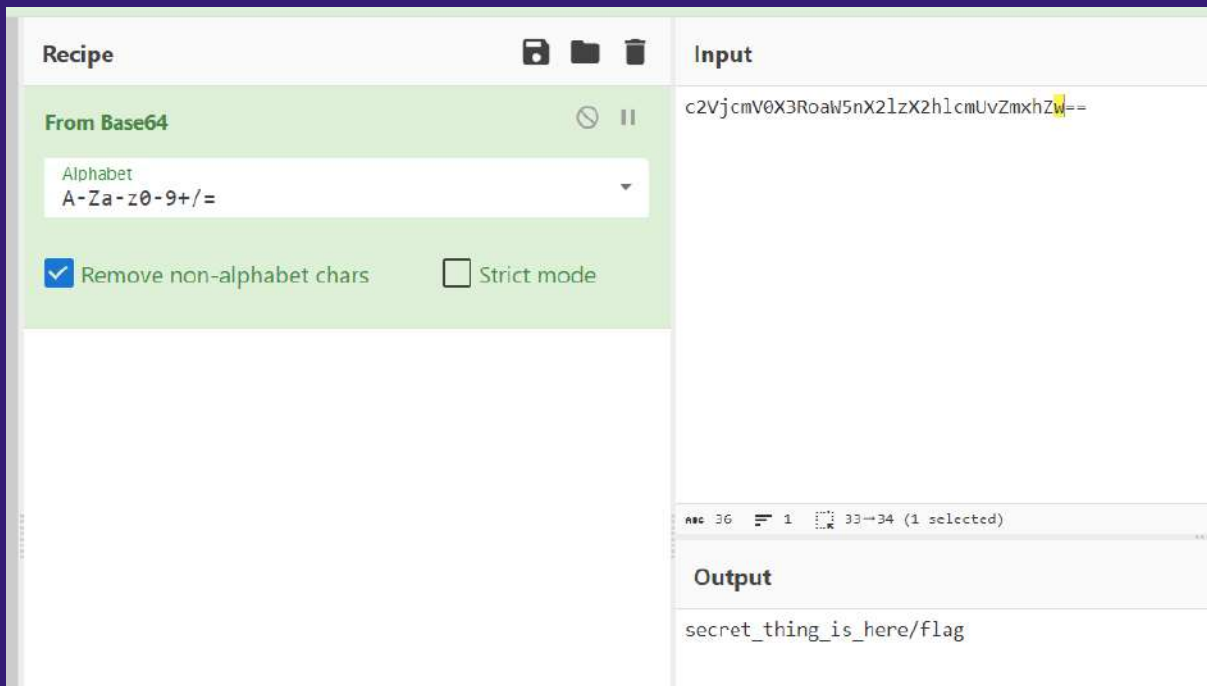
Output

f827cf462f62848df37c5e1e94a4da74

Setelah melakukan tampering, hasilnya ialah seperti ini.



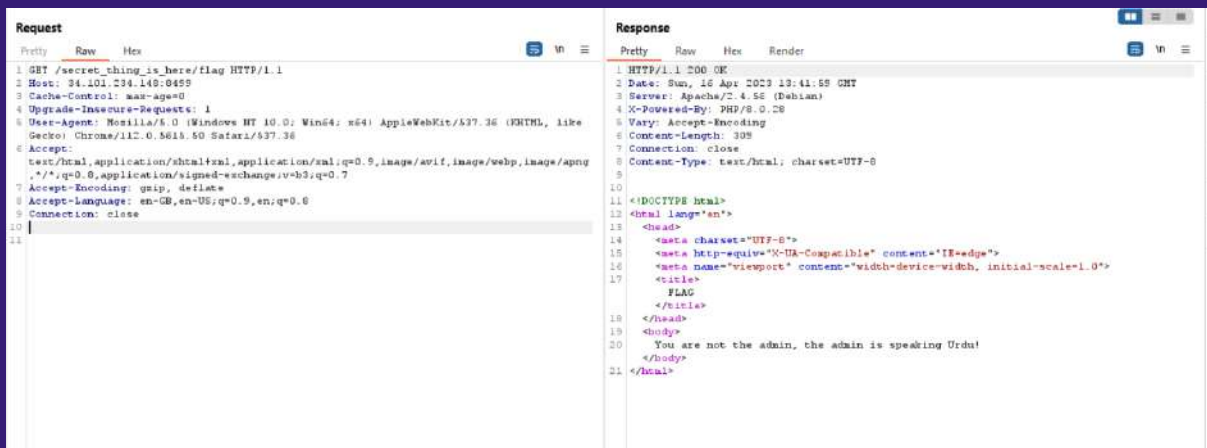
Kami telah berhasil menjadi admin dan diberikan sebuah string yang kami duga sebagai base64 encoded (terlihat dari adanya karakter sama dengan di akhir string). Jika kita coba decode, hasilnya adalah seperti ini.



Kami mendapatkan sebuah petunjuk untuk mengunjungi sebuah directory yakni "secret_thing_is_here/flag". Langsung saja kita coba untuk kunjungi dan berikut adalah tampilannya.



Terlihat bahwa disini diperlukan beberapa modifikasi pada request header agar bisa memenuhi syarat-syarat yang dibutuhkan. Untuk mempermudah hal tersebut, kami memutuskan untuk menggunakan burp suite.

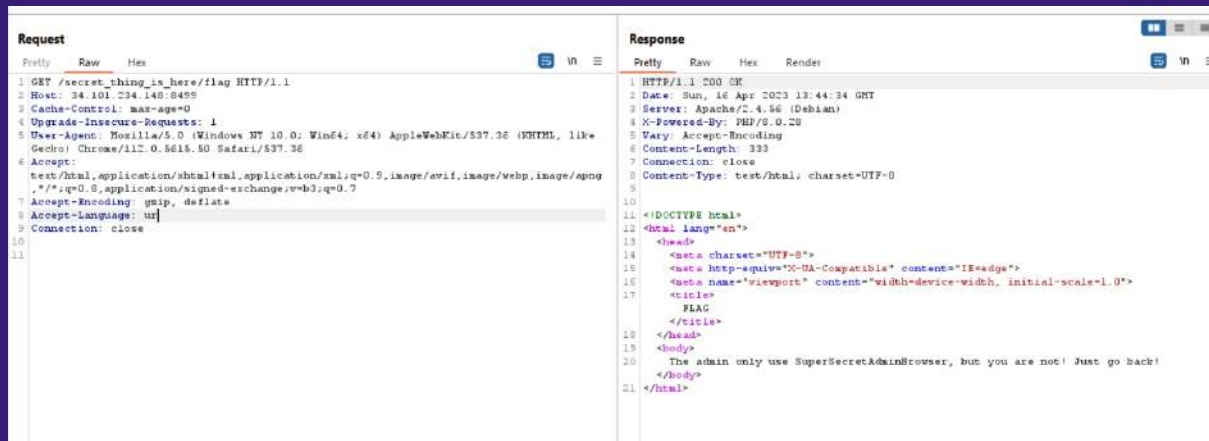


Dengan menggunakan burp suite, kita bisa langsung saja mengubah header yang diperlukan.

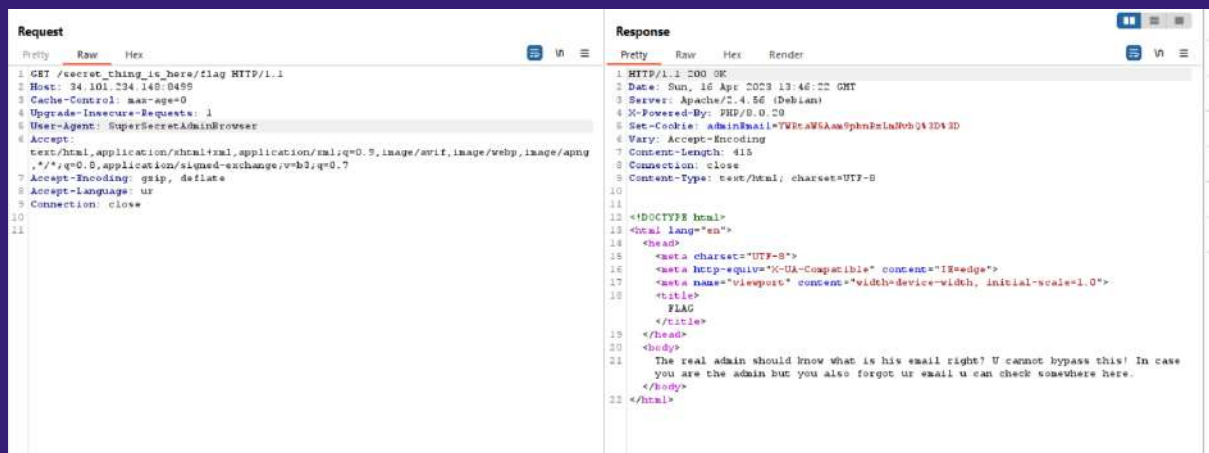
Pada syarat pertama, kita diminta untuk berbicara dalam bahasa Urdu. Dengan sedikit googling, maka bisa ditemukan bahwa kode untuk penggunaan bahasa Urdu adalah "ur" (<https://www.w3.org/International/ms-lang.html>).

Ukrainian	Ukrainian	0422; UKR	X	X	uk
Urdu	Urdu	0420; URD			ur

Dan header yang berfungsi untuk menampung informasi tersebut ialah header "Accept-Language" (<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Accept-Language>). Jadi, untuk memenuhi syarat pertama kita bisa menggunakan "Accept-Language: ur" sebagai header.



Terlihat bahwa respons sudah berubah, menandakan bahwa langkah sebelumnya sudah benar. Untuk syarat kedua, kita diminta untuk menggunakan browser bernama “SuperSecretAdminBrowser”. Pada syarat kedua, kita bisa mengganti value dari header “User Agent”. Header tersebut digunakan untuk menampung informasi terkait browser yang kita gunakan.



Lanjut ke syarat ketiga, kita diminta untuk juga memasukkan email pada header. Namun bagaimana cara kita mengetahui email yang benar? kita bisa memperhatikan pada response header bahwa terdapat header “Set-Cookie” dengan value “adminEmail”. Tentunya ini adalah email yang sedang kita cari-cari. Kita bisa melakukan decode pada value tersebut dan kita akan mendapatkan emailnya yaitu “admin@joints.com”.



Nah, setelah mendapatkan emailnya maka kita bisa memasang value tersebut pada header “From” yang mana digunakan untuk menginformasikan pengiriman email (<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/From>).

Setelah digunakan, maka responsnya akan berubah menjadi seperti ini.

```

Body Cookies (1) Headers (9) Test Results
Pretty Raw Preview Visualize HTML
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <title>FLAG</title>
9 </head>
10
11 <body>
12   U was tracked. Use untracked one to go in real admin area!</body>
13
14 </html>

```

Pada syarat ini, kita memerlukan sebuah header baru yakni “Do Not Track (DNT)” yang berfungsi untuk melakukan pemblokiran pada tracker (<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/DNT>). Jadi,

kita bisa memasang header baru yakni “DNT: 1” untuk memenuhi syarat terakhir ini.

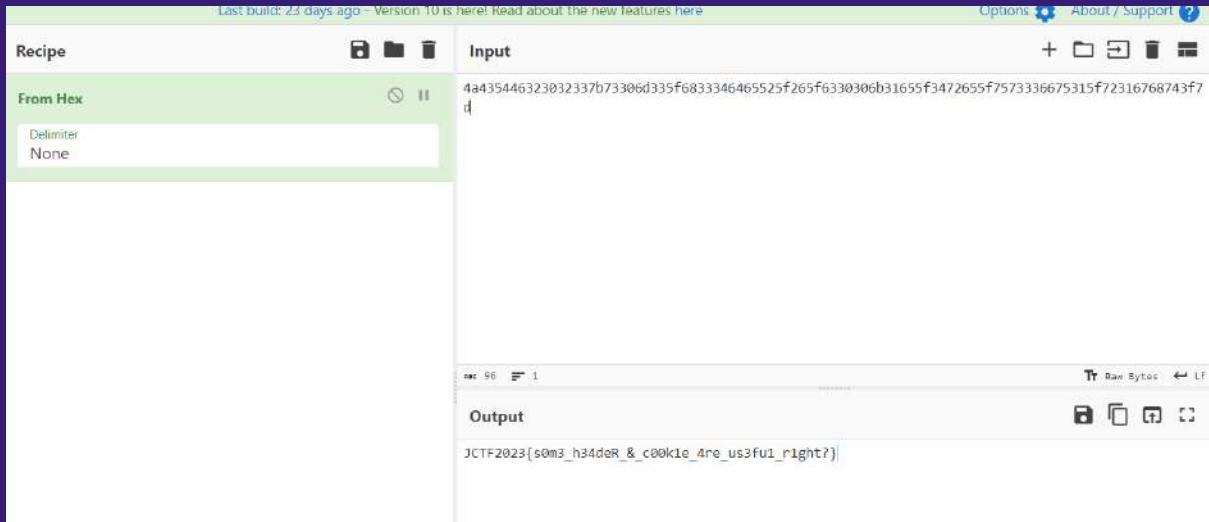
```

1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <title>FLAG</title>
9 </head>
10
11 <body>
12   <p>Congrats! Here's my Response to you! :</p>
13   <p>Welcome to Admin Area! My Real Admin!</p>
14 </body>
15
16 </html>
  
```

Terlihat bahwa sudah tidak ada syarat-syarat lagi yang berarti kita seharusnya sudah menyelesaikan challenge ini, namun tidak terlihat keberadaan flag. Setelah melakukan pengecekan kembali, ternyata terdapat response header yang unik yakni sebagai berikut.

KEY	VALUE
Date	Sun, 16 Apr 2023 14:06:25 GMT
Server	Apache/2.4.56 (Debian)
X-Powered-By	PHP/8.0.28
For-Admin-Only	4a435446323032337b73306d335f6833346465525f265f6330306b31655f3472655f7573336675315f72316768743f7d
Vary	Accept-Encoding
Content-Encoding	gzip
Content-Length	257
Keep-Alive	timeout=5, max=99
Connection	Keep-Alive
Content-Type	text/html; charset=UTF-8

Terlihat bahwa ada sebuah header “For-Admin-Only” dengan value yang juga cukup unik. Mari kita coba decode dengan menggunakan cyber chef.



Dan ternyata string tersebut merupakan flag yang diencode dengan menggunakan hex. Dengan begitu, maka challenge ini telah selesai.

Flag =

JCTF2023{s0m3_h34deR_&_c00k1e_4re_us3fu1_r1ght?}

Web of the Gods

Challenge 24 Solves X

Web of the Gods

300

The power of a god... One could only dream it.

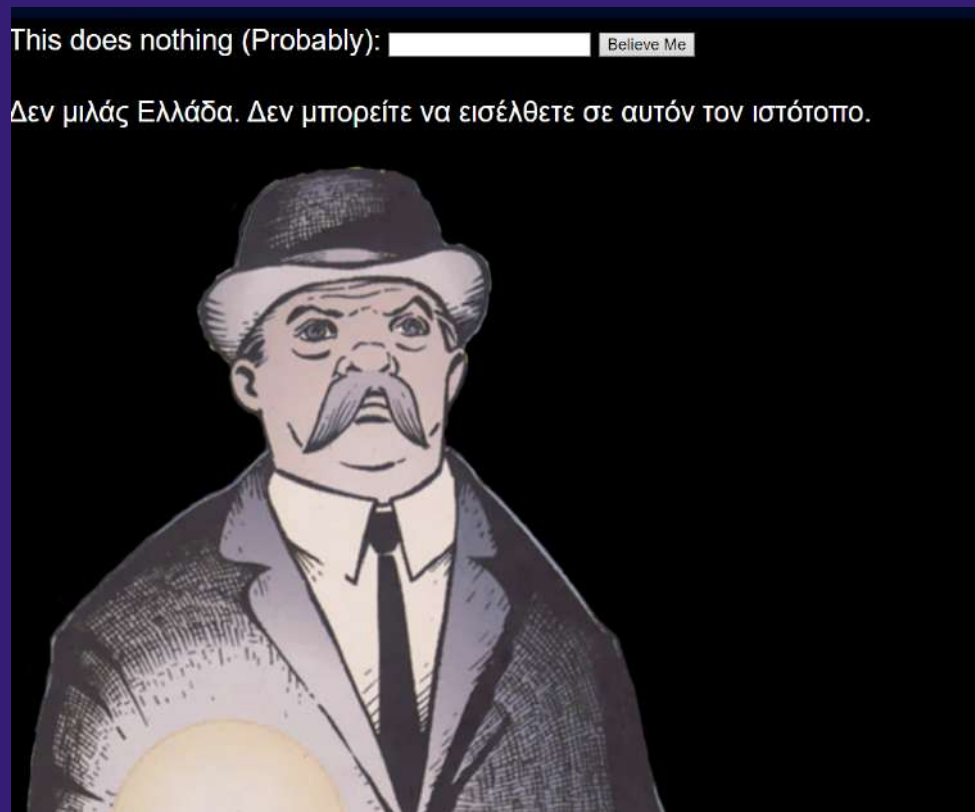
Author: Giga - Infinicus#6867

34.101.234.148:8069

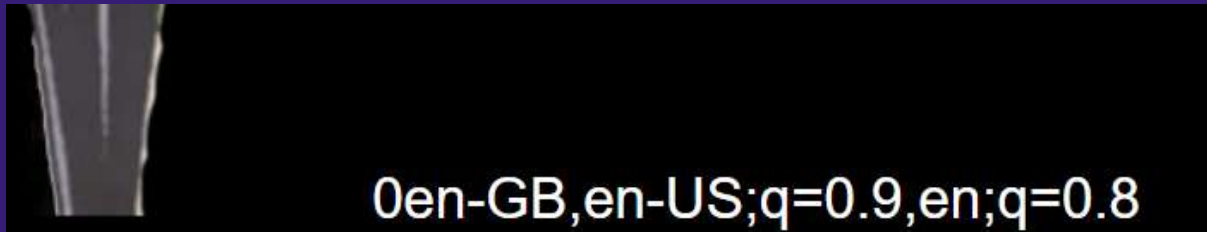
► View Hint

Submit

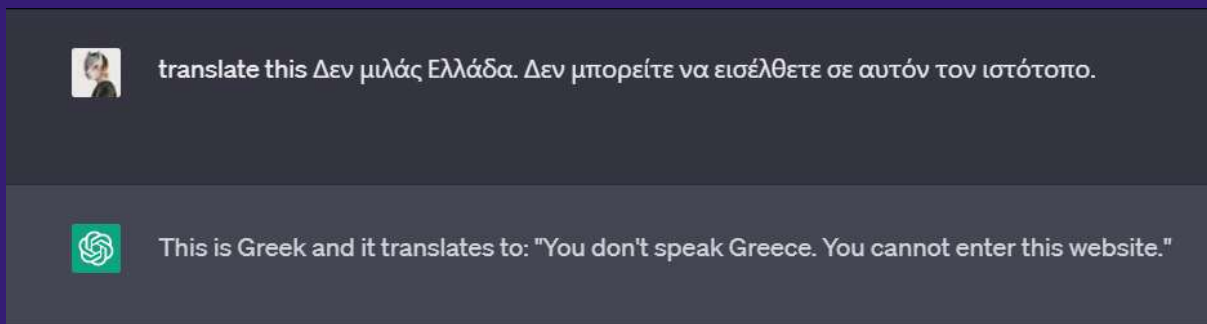
Pada soal ini, diberikan tampilan website yaitu sebagai berikut.



Website yang hanya berisi 1 buat input box yang mana akan selalu memiliki hasil yang sama yaitu alert. Disini, kami berpikir bahwa tentu saja diperlukan pendekatan lain untuk bisa mendapatkan flag. Jika diperhatikan lebih lanjut lagi, terdapat juga string berikut pada bagian bawah website.



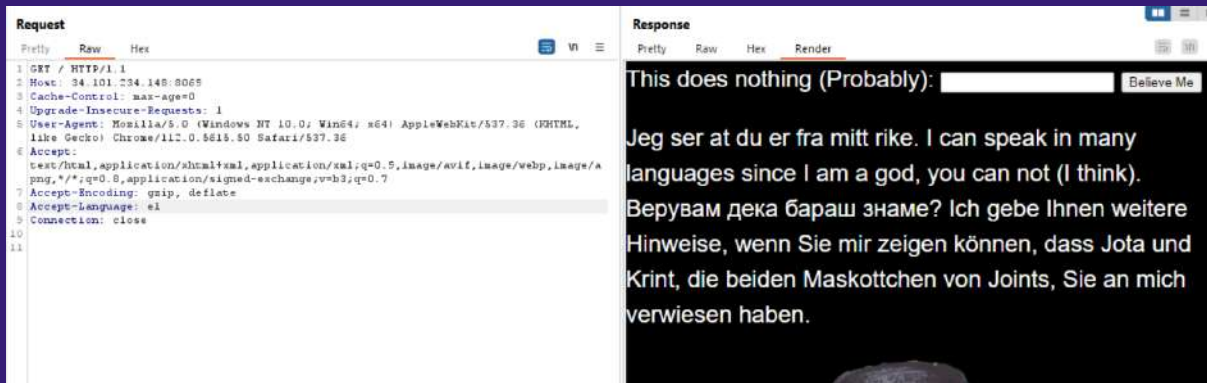
Ini mengarah pada value dari sebuah header "Accept-Language", namun apa hubungannya dengan challenge kali ini? Setelah beberapa waktu berpikir, akhirnya kami mencoba untuk mentranslate string yang ada bagian atas website tersebut. Disini kami menggunakan bantuan chatgpt agar bisa mengerti makna dari kata-kata tersebut dengan mudah.



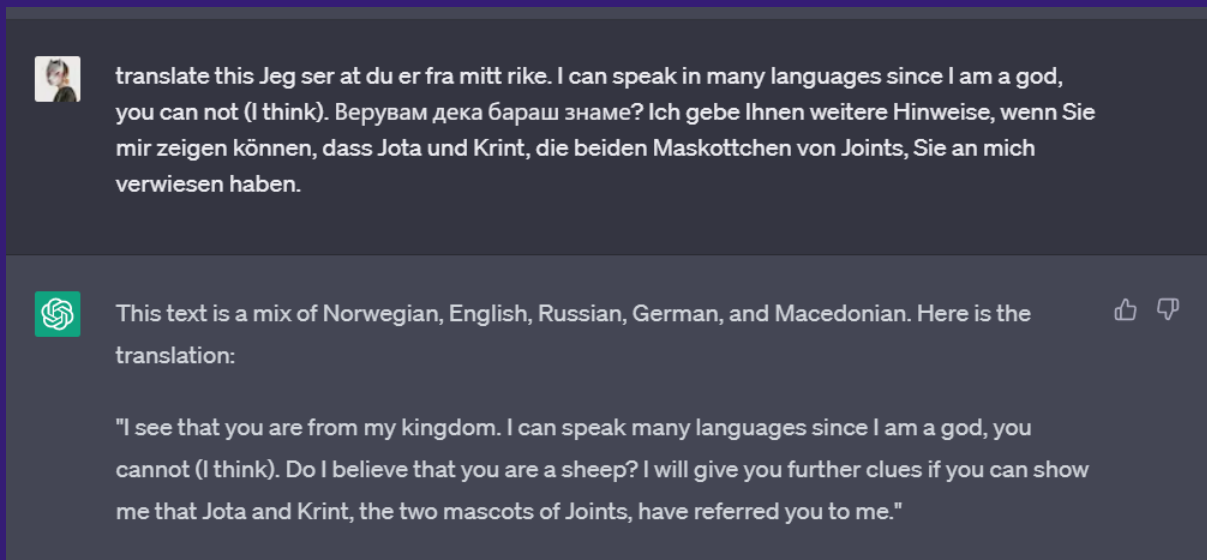
Ternyata artinya adalah seperti di atas, disini kami menyadari bahwa kami perlu melakukan penyamaan header seperti pada challenge sebelumnya. Untuk memenuhi syarat ini, kita tinggal mencari value "Accept-Language" untuk Greece.

Greek	Greek	0408; ELL	x	x	el	
-------	-------	-----------	---	---	----	--

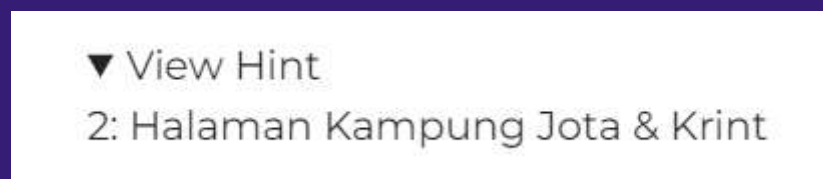
Ternyata valuenya adalah "el", dengan begitu kita bisa memasang header "Accept-Language: el". Setelah memasang header tersebut, kita akan mendapatkan response seperti ini.



Kembali kami menggunakan bantuan chatgpt untuk mempermudah translate.



Ternyata ini ada hubungannya dengan “Referer” header, namun value seperti apa yang harus dimasukkan? Setelah beberapa kali mencoba, kami pun sempat stuck untuk menemukan value yang tepat. Untungnya, terdapat hint yang berbunyi demikian.



Untuk syarat kedua, ada hubungannya dengan “Halaman Kampung” dari Jota & Krint. Disini kami cukup yakin “Halaman Kampung” yang dimaksud ialah sebuah website atau “Home Page”. Home page dari Jota & Krint tentunya adalah website dari Joints itu sendiri. Dengan begitu, kita bisa memasang header “Referer: <https://www.jointsugm.id/>” untuk memenuhi syarat kedua.

```
<input type='submit' value='Believe Me' onclick='popup()'>
</form>
<p>Bonvenon! Jota an Krint si meng gutt Frënn. Aia ka hae ma kahi huna, 'a'ole au makemake e 'ike kekahi i kēia.
मुझे बस यह सुनिश्चित करने की आवश्यकता है कि कोई भी आपका अनुसरण न करे। Tiedät kuinka todistaa se.</p>
<img src='Mount-Olympus/Presence.png' alt='Presence'>2e1</body>
```

Hasil responsenya adalah seperti di atas, kembali kita diberikan sebuah instruksi baru lagi. Kami kembali melakukan translate untuk memahami instruksi selanjutnya.



translate this Bonvenon! Jota an Krint si meng gutt Frënn. Aia ka hae ma kahi huna, 'a'ole au makemake e 'ike kekahi i kēia.

मुझे बस यह सुनिश्चित करने की आवश्यकता है कि कोई भी आपका अनुसरण न करे। Tiedät kuinka todistaa se.



This text is a mix of Esperanto, Marshallese, Hawaiian, Hindi, and Finnish. Here is the translation:

"Welcome! Jota and Krint are my good friends. They have something to tell me, but I don't want anyone else to know about it. You know how to prove it."

Sekarang kita diminta agar tidak diketahui oleh siapapun, tentu saja hal ini berhubungan dengan header "DNT" kembali. Dengan begitu, kita bisa memasang header "DNT: 1" untuk memenuhi syarat ini.

```
<input type='submit' value='Believe Me' onclick='popup()'>
</form>
<p>رائع ، الآن أعرف أنني أستطيع أن أثق بك. Die vlag word in die lêer geplaas 'Domain-of-Gods/secrpt.js'. Ma tha
thu air tighinn cho fada seo, tuigidh tu agus lorg thu a' bhratach. Boa sorte, você pode ganhar este jogo.
</p><img src='Mount-Olympus/Presence.png' alt='Presence'>3e1</body>
</html>
```

Kembali kita diberikan instruksi baru dengan campuran bahasa lagi. Dan dengan cara yang sama kami melakukan translate.



translate this رائع ، الآن أعرف أنني أستطيع أن أثق بك. Die vlag word in die lêer geplaas 'Domain-of-Gods/secrpt.js'. Ma tha

thu air tighinn cho fada seo, tuigidh tu agus lorg thu a' bhratach. Boa sorte, você pode ganhar este jogo.



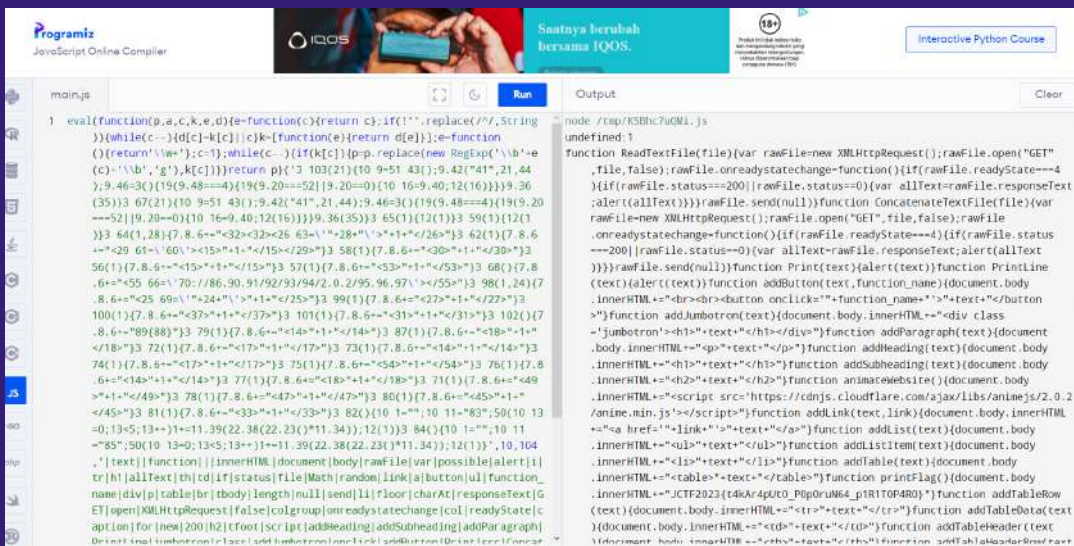
This text is a mix of Arabic, Afrikaans, Scots Gaelic, and Portuguese. Here is the translation:

"Great, now I know I can trust you. The flag is placed in the file 'Domain-of-Gods/script.js'. But if you've come this far, you'll understand and find the flag. Good luck, you can win this game."

Hasilnya adalah seperti demikian, sekarang kita tahu bahwa terdapat path “Domain-of-Gods/script.js”. Langsung saja kita kunjungi directory tersebut.

[illegible]

Ternyata kita diberikan sebuah script js yang keliatannya rumit dan cukup panjang. Disini, kami langsung berpikir untuk menjalankan script js tersebut dengan menggunakan online compiler dan melihat hasilnya. Setelah dijalankan, hasilnya adalah sebagai berikut.



Sekarang hasilnya lebih mudah dibaca dan strukturnya terlihat seperti sebuah HTML source code lengkap dengan js function. Tanpa perlu dijalankan, kita bisa melihat bahwa flag dari challenge ini sudah terlihat.

```
L+="



" }function printFlag(L+="

|                                           |
|-------------------------------------------|
| JCTF2023{t4kAr4pUt0_P0p0ruN64_p1R1T0P4R0} |
|-------------------------------------------|

") {document.body.innerHTML += "| JCTF2023{t4kAr4pUt0_P0p0ruN64_p1R1T0P4R0} |
" }
```

Dengan begitu, maka challenge ini telah selesai.

Flag = JCTF2023{t4kAr4pUt0_P0p0ruN64_p1R1T0P4R0}

Vision

Challenge 67 Solves X

Vision

100

Jota visited a strange website that asked him to enter a secret code, can you help Jota to explore the website ?

Author: Jears #8964

34.101.234.148:8239

Submit

Pada challenge ini, diberikan tampilan website yaitu sebagai berikut.



Terlihat terdapat sebuah input box yang meminta kita untuk memasukkan sebuah code. Namun, code apa yang perlu kita masukkan? Untuk menjawab hal ini, kita bisa melihat source code HTML dari website tersebut.

```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Web Exploitation Challenge</title>
8     <link rel="stylesheet" href="views/style.css">
9   </head>
10  <body>
11    <h1>Hello, Welcome to my website</h1>
12    <h2>What's the secret code ?</h2> <br> <br>
13    <input type="text" placeholder="Enter the code" id="userInput"> <br> <br>
14    <button type="submit" class="btn" onclick="inputCode()">Submit</button>
15    <h1 id="message"></h1>
16    
17    <div class="popup" id="popup">
18      <h2>Congratulation</h2>
19       <br><br>
20      <a href="/webchallSecret"> Next </a>
21    </div>
22    <style>
23      body{background-color: orange; text-align: center; color: green;}
24    </style>
25    <script>
26      let popup = document.getElementById("popup");
27      function inputCode() {
28        let input= document.getElementById("userInput").value;
29        let message = document.querySelector("#message")
30        if(input == "mantapujiwa"){
31          popup.classList.add("showPopup");
32          message.innerHTML = "Your code is right!";
33        }
34        else{
35          message.innerHTML = "Your code is wrong!";
36        }
37      }
38    </script>
39  </body>
40 </html>

```

Terlihat dari source code JS script yang ada, bahwa code yang benar adalah “mantapujiwa”. Kita bisa langsung saja memasukkan code tersebut dan tampilan website akan berubah menjadi seperti ini.



Terlihat kosong. Kembali kita bisa melihat pada source code HTML nya untuk menganalisa apa yang sebenarnya terjadi.

```

line wrap
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Hidden Flag</title>
8     <link rel="stylesheet" href="views/style2.css">
9
10  </head>
11  <body>
12    <h1>Can you make me visible ?</h1>
13    <div class="popup" id="popup">
14      <h2>Congratulation</h2>
15      <div class="row">
16        <div class="column">
17          
18          
19          
20          
21          
22        </div>
23        <div class="column">
24          
25          
26          
27          
28          
29        </div>
30        <div class="column">
31          
32          
33          
34          
35          
36        </div>

```

Terlihat bahwa sebenarnya flag sudah ada dan tersusun dalam bentuk gambar, namun kenapa tidak terlihat? Untuk menjawab pertanyaan ini, kita bisa melihat pada source code CSS yang digunakan.

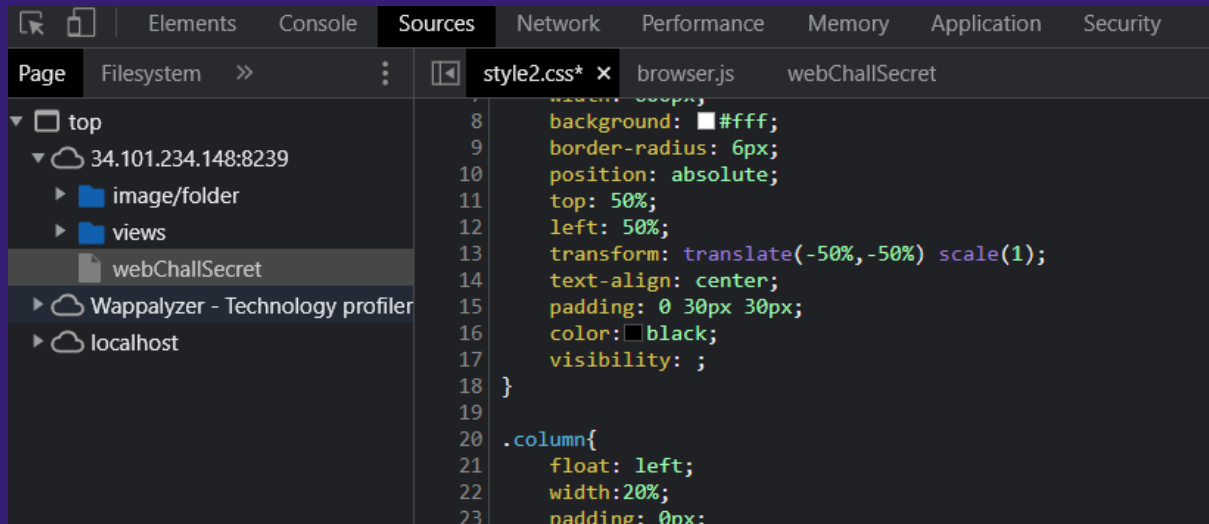
```
*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
.popup{
  width: 600px;
  background: #fff;
  border-radius: 6px;
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%,-50%) scale(1);
  text-align: center;
  padding: 0 30px 30px;
  color:black;
  visibility: hidden;
}

.column{
  float: left;
  width:20%;
  padding: 0px;
}

.row::after {
  content: "";
  display: table;
  clear: both;
}

}
```

Ternyata, hal inilah yang menyebabkan flag tidak terlihat (adanya penggunaan attribute hidden pada class milik flag). Namun, dikarenakan ini hanyalah di front end saja maka kita bisa langsung mendelete saja value dari attribute tersebut ketika CSS di load pada webpage. Hal ini dapat dengan mudah dilakukan lewat inspect source via browser.



Dengan menghilangkan value tersebut, maka flag akan terlihat.



Flag = JCTF2023{s0_e4sy_w3b_3xPI0itation}

NWORDPASS

Challenge 4 Solves X

NWORDPASS

718

This company is giving away the N Word Pass, I kinda don't like this company and I hear that they got something more precious than this pass. Can you help me get it? Someone say that they might host multiple services.

author: Lurifos

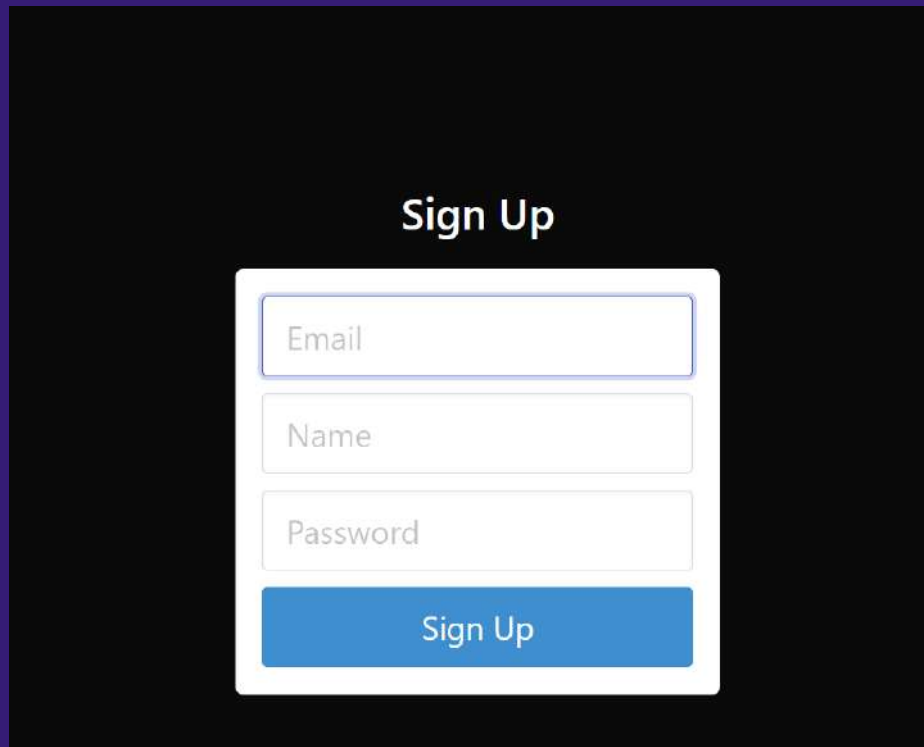
34.101.234.148:8171

Flag Submit

Pada challenge ini, diberikan website dengan tampilan sebagai berikut.

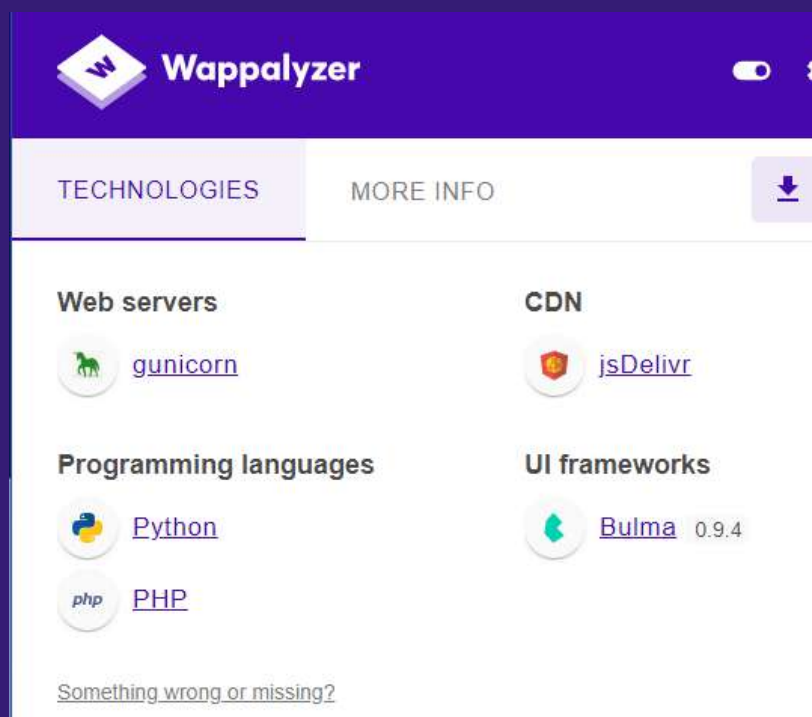


Dikarenakan disini tidak diberikan source code, maka kami langsung saja mencoba beberapa fitur-fitur seperti fitur Login dan juga Sign Up. Pada fitur Login, tidak ditemukan celah SQLi ataupun semacamnya. Sehingga kami langsung saja mencoba untuk Sign Up.

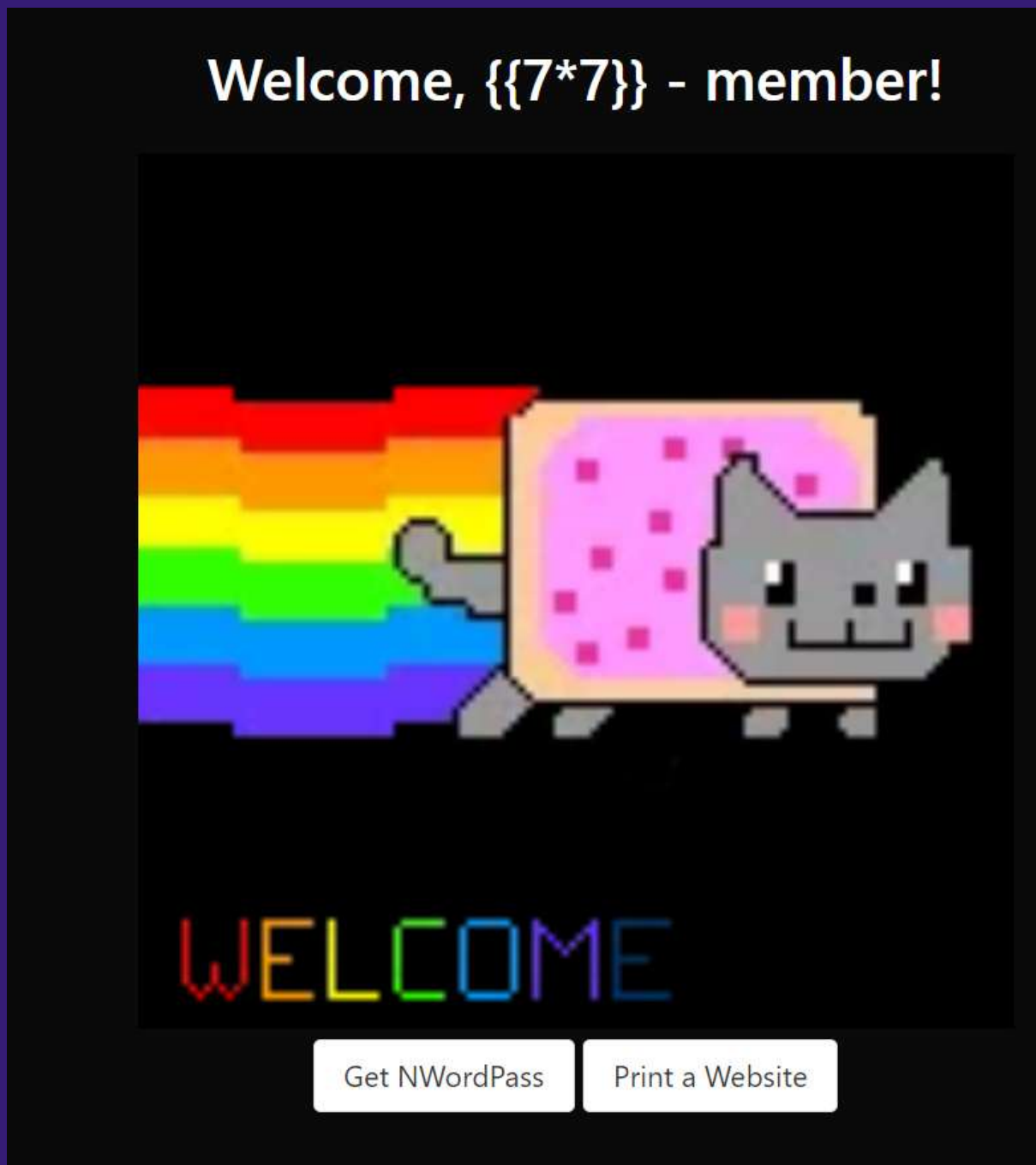


A screenshot of a 'Sign Up' form. The form is centered on a black background. It consists of a white rounded rectangle containing three input fields: 'Email', 'Name', and 'Password'. Below these fields is a blue button with the text 'Sign Up' in white. The text 'Sign Up' is also displayed above the form in a large, white, sans-serif font.

Disini kami mencoba untuk Sign Up dengan memasukkan sebuah payload yakni SSTI. Hal ini kami lakukan dikarenakan untuk mempersingkat waktu dalam testing. Adapun, alasan kami memasukkan payload SSTI ialah dikarenakan salah satu backend yang digunakan adalah python (biasanya python website menggunakan template engine yang vulnerable terhadap SSTI).



Setelah login, kami mendapatkan tampilan dashboard sebagai berikut.



Terdapat 2 buah fitur, yaitu "Get NWordPass" dan "Print a Website". Fitur "Get NwordPass" tidak bisa digunakan dikarenakan kami bukanlah admin. Kemudian, pada fitur yang satunya lagi tampilannya adalah sebagai berikut.

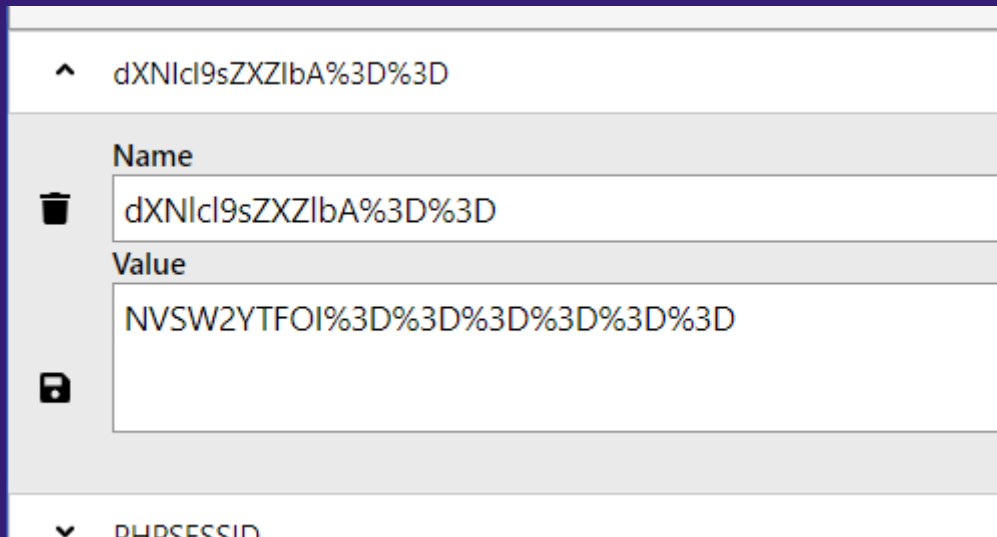
Thank you for using our service!

URL

Print

Kami menyadari bahwa ini seperti sebuah service untuk melakukan convert website content ke PDF. Kami sempat mencoba untuk melakukan file read namun tidak berhasil dan kami juga belum tahu mau melakukan read ke file apa. Oleh karena itu, kami memutuskan untuk menunda testing pada fitur ini dan mundur kembali ke dashboard.

Disini kami mencoba untuk melakukan privilege escalation menjadi admin. Kami mencoba untuk melakukan pengecekan pada cookie dan ternyata benar saja.



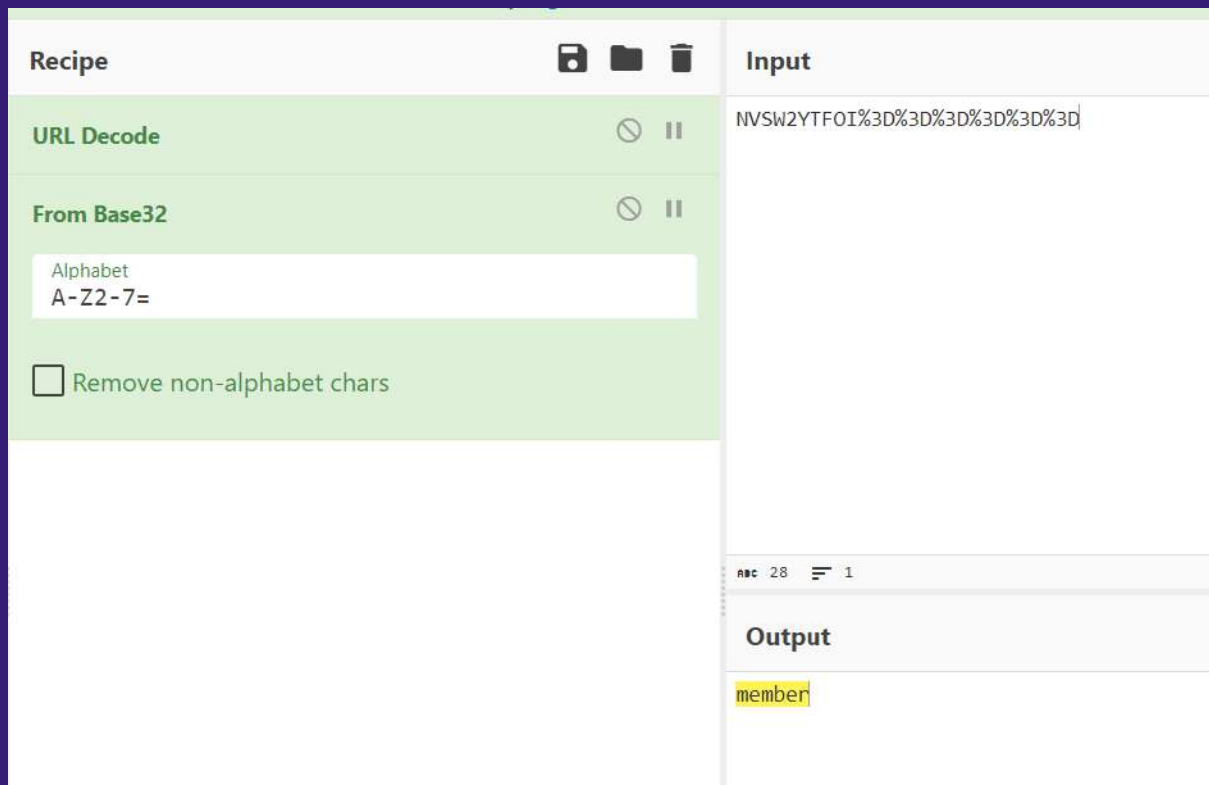
^ dXNIcl9sZXZlbA%3D%3D

Name
dXNIcl9sZXZlbA%3D%3D

Value
NVSW2YTF0I%3D%3D%3D%3D%3D%3D

PHPSESSID

Lagi-lagi terdapat cookie dengan nama dan value yang tidak biasa. Kita bisa langsung decode dengan cyber chef.



Recipe

URL Decode

From Base32

Alphabet
A-Z2-7=

☐ Remove non-alphabet chars

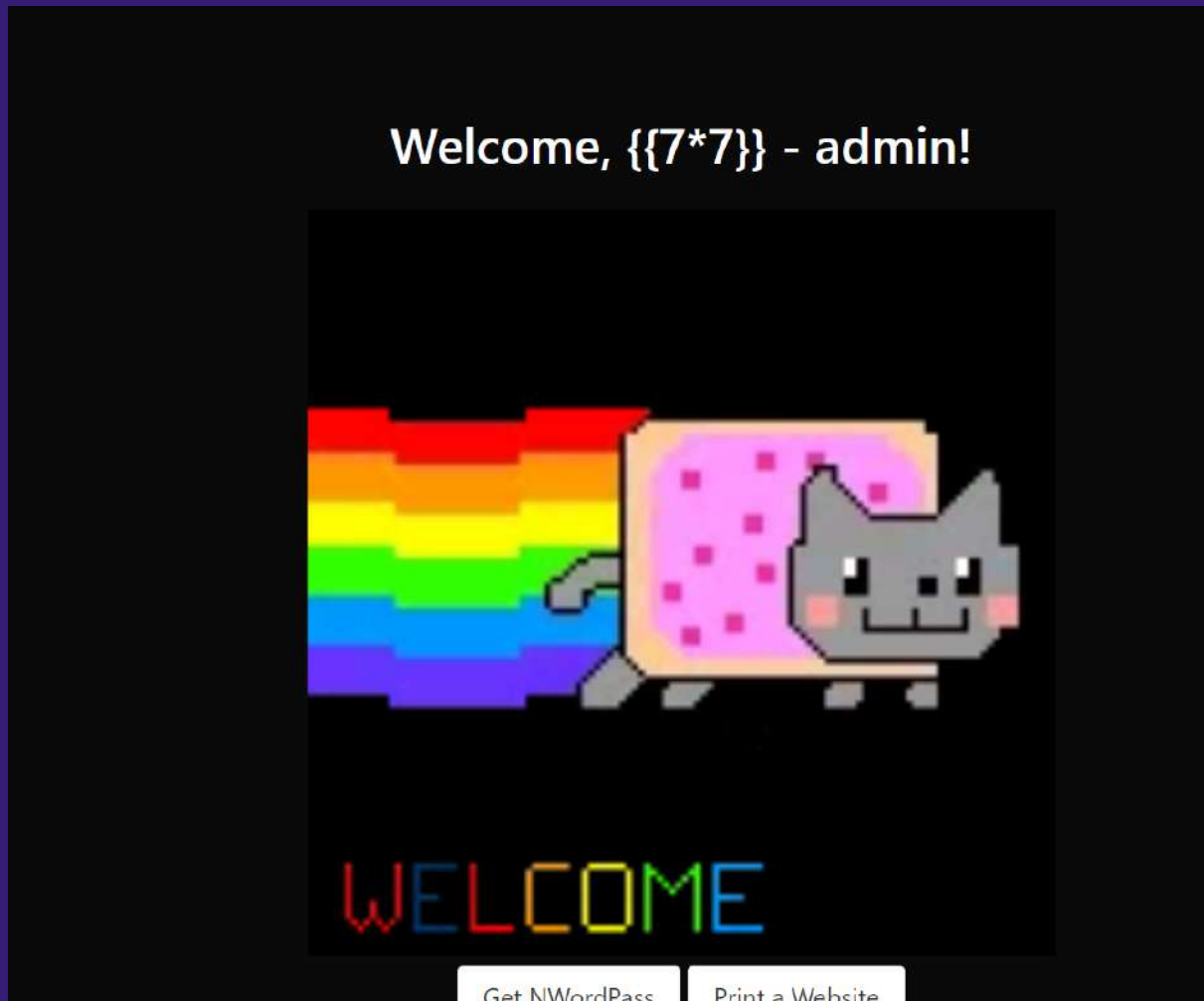
Input

NVSW2YTF0I%3D%3D%3D%3D%3D%3D

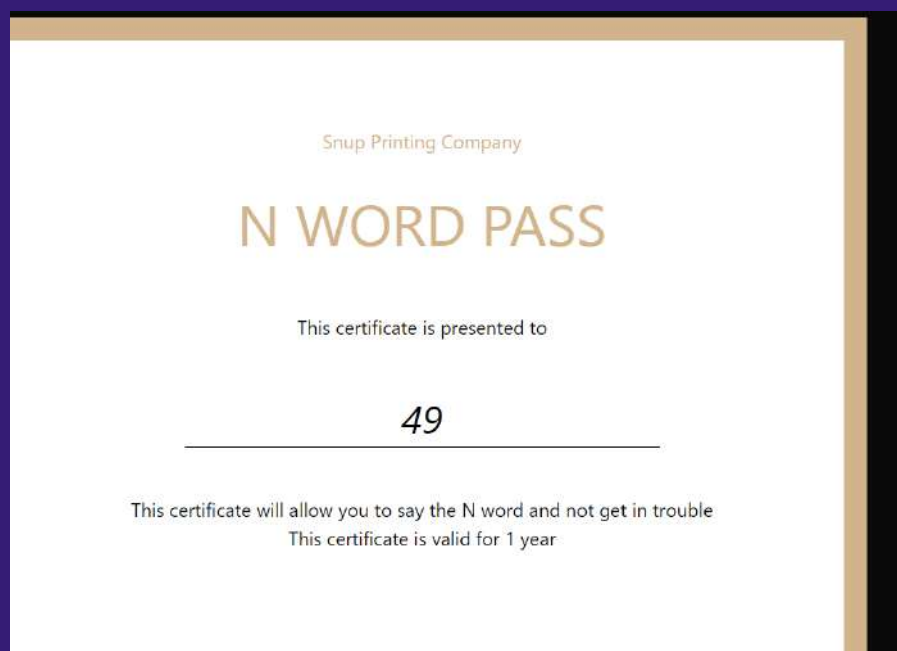
Output

member

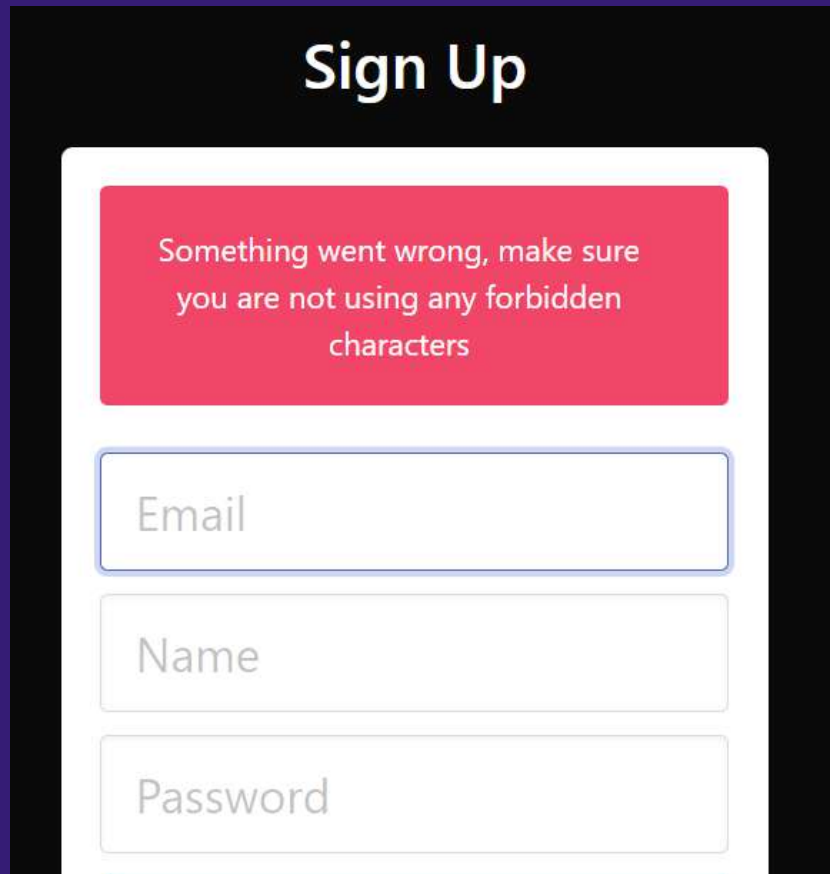
Terlihat bahwa value nya memiliki arti “member”, dengan begitu kita bisa coba untuk tampering value nya menjadi admin. Setelah ditamper, maka hasilnya menjadi seperti ini.



Voila! Kita sudah menjadi admin. Karena kita sudah menjadi admin, mari kita cek fitur yang sebelumnya tidak bisa kita akses.



Benar saja, payload SSTI pada username yang sebelumnya kami gunakan dirender menjadi “49” yang berarti website ini vulnerable terhadap SSTI. Dari sini, kami berusaha untuk melakukan RCE dengan menggunakan berbagai macam payload. Namun, sepertinya terdapat blacklist yang cukup ketat.



The image shows a 'Sign Up' form on a dark background. The form is white and contains a red error message box at the top. Below the error message are three input fields labeled 'Email', 'Name', and 'Password'.

Sign Up

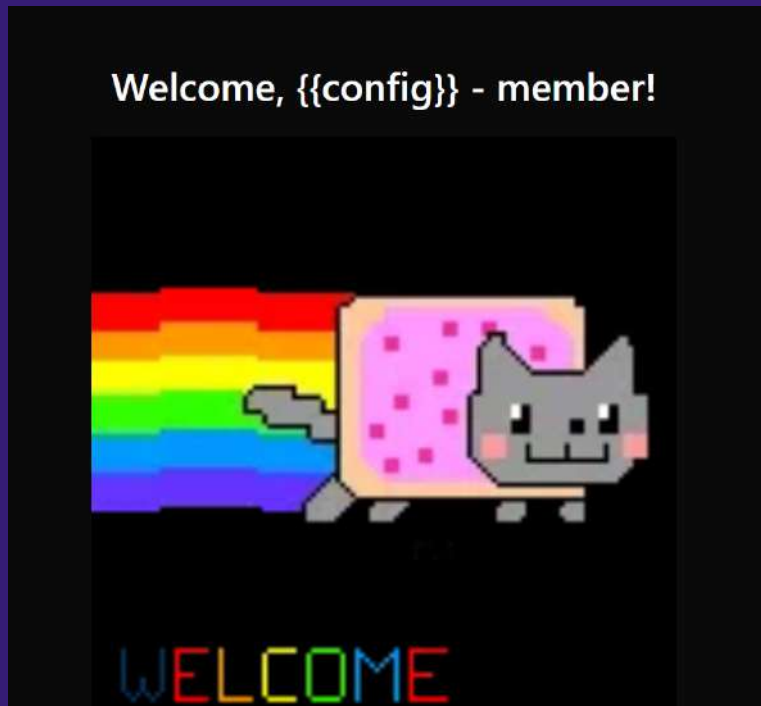
Something went wrong, make sure you are not using any forbidden characters

Email

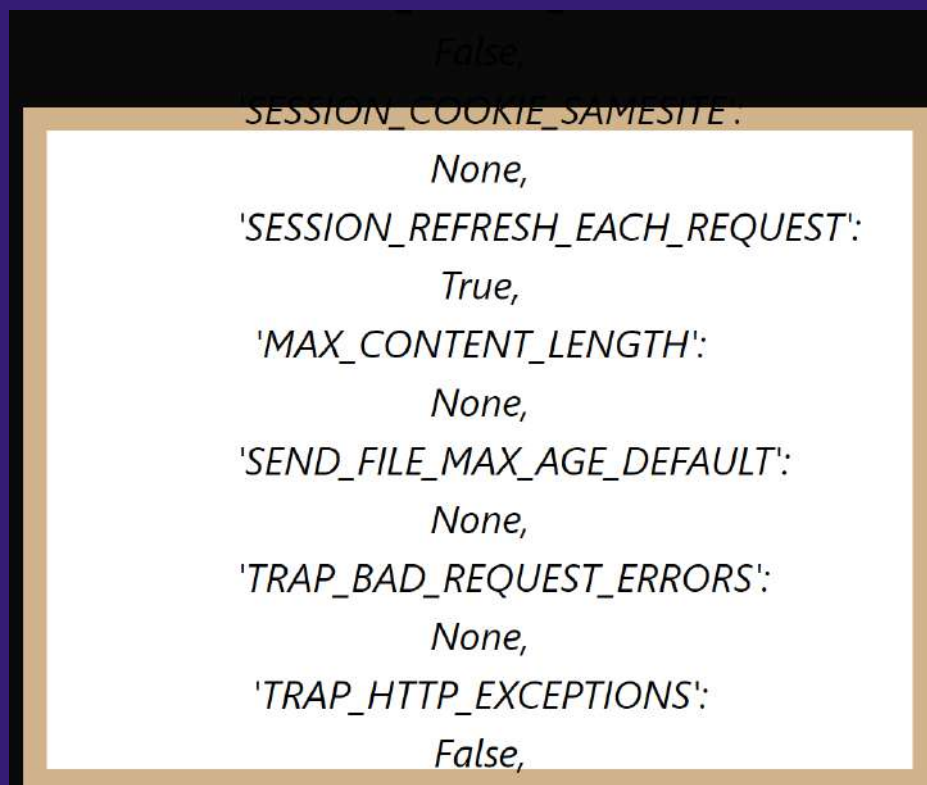
Name

Password

Kami mencoba untuk berpikir kembali dan mencoba untuk menggunakan payload yang sederhana dengan tujuan untuk enumerasi saja.



Kita ulangi langkah yang sama seperti pada langkah sebelumnya dan hasilnya ternyata seperti ini.



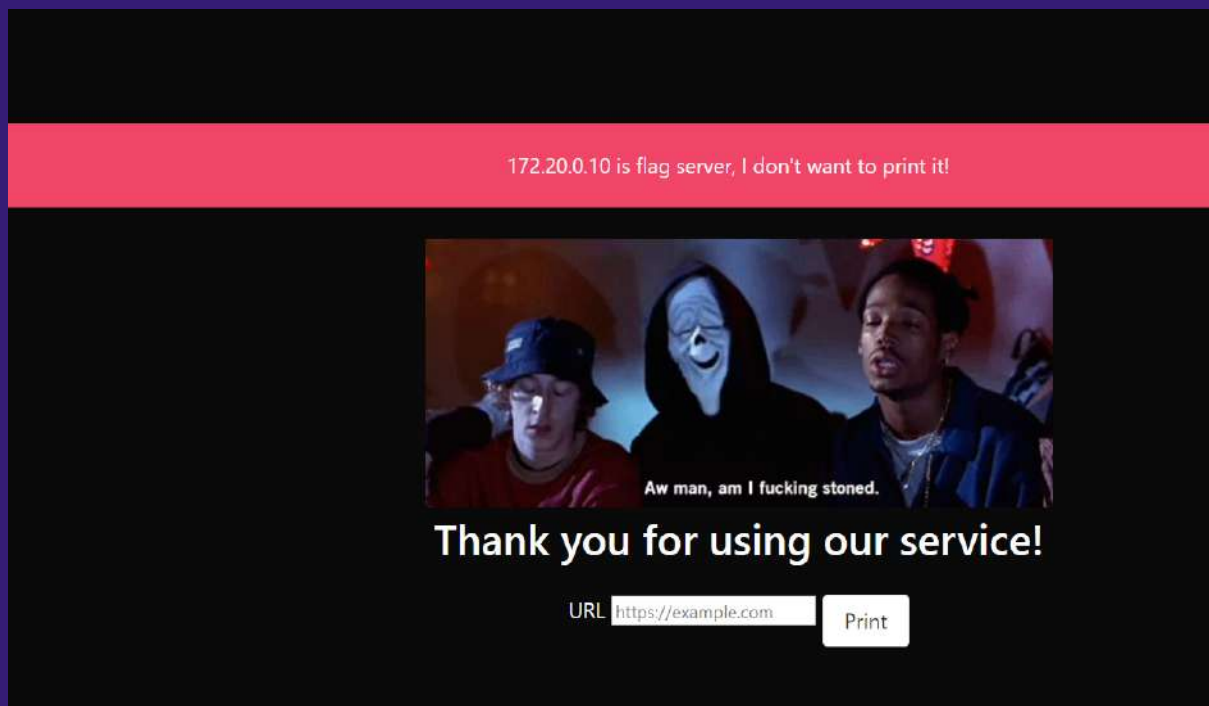
Ternyata payload tersebut berhasil, setelah dirapikan maka outputnya adalah seperti ini.

```
Output
<config {'ENV': 'production', 'DEBUG': False, 'TESTING': False, 'PROPAGATE_EXCEPTIONS': None, 'SECRET_KEY': 'randomstringtr4stme', 'PERMANENT_SESSION_LIFETIME': datetime.timedelta(days=31),
'USE_X_SENDFILE': False, 'SERVER_NAME': None, 'APPLICATION_ROOT': '/', 'SESSION_COOKIE_NAME': 'session', 'SESSION_COOKIE_DOMAIN': False, 'SESSION_COOKIE_PATH': None,
'SESSION_COOKIE_HTTPONLY': True, 'SESSION_COOKIE_SECURE': False, 'SESSION_COOKIE_SAMESITE': None, 'SESSION_REFRESH_EACH_REQUEST': True, 'MAX_CONTENT_LENGTH': None,
'SEND_FILE_MAX_AGE_DEFAULT': None, 'TRAP_BAD_REQUEST_ERRORS': None, 'TRAP_HTTP_EXCEPTIONS': False, 'EXPLAIN_TEMPLATE_LOADING': False, 'PREFERRED_URL_SCHEME': 'http', 'JSON_AS_ASCII': None,
'JSON_SORT_KEYS': None, 'JSONIFY_PRETTYPRINT_REGULAR': None, 'JSONIFY_MIMETYPE': None, 'TEMPLATES_AUTO_RELOAD': None, 'MAX_COOKIE_SIZE': 4093, 'SQLALCHEMY_DATABASE_URI':
'sqlite:///db.sqlite', 'FLAG_URL': 'http://172.20.0.10:1234/flag6386236835', 'SQLALCHEMY_ENGINE_OPTIONS': {}, 'SQLALCHEMY_ECHO': False, 'SQLALCHEMY_BINDS': {}, 'SQLALCHEMY_RECORD_QUERIES':
False, 'SQLALCHEMY_TRACK_MODIFICATIONS': False}>
```

Dan ternyata terdapat sebuah URL rahasia yang kemungkinan besar berisi flag.

```
, 'JSONIFY_PRETTYPRINT_REGULAR': None, 'JSONIFY_MIMETYPE': None,
'FLAG_URL': 'http://172.20.0.10:1234/flag6386236835',
'SQLALCHEMY_TRACK_MODIFICATIONS': False}
```

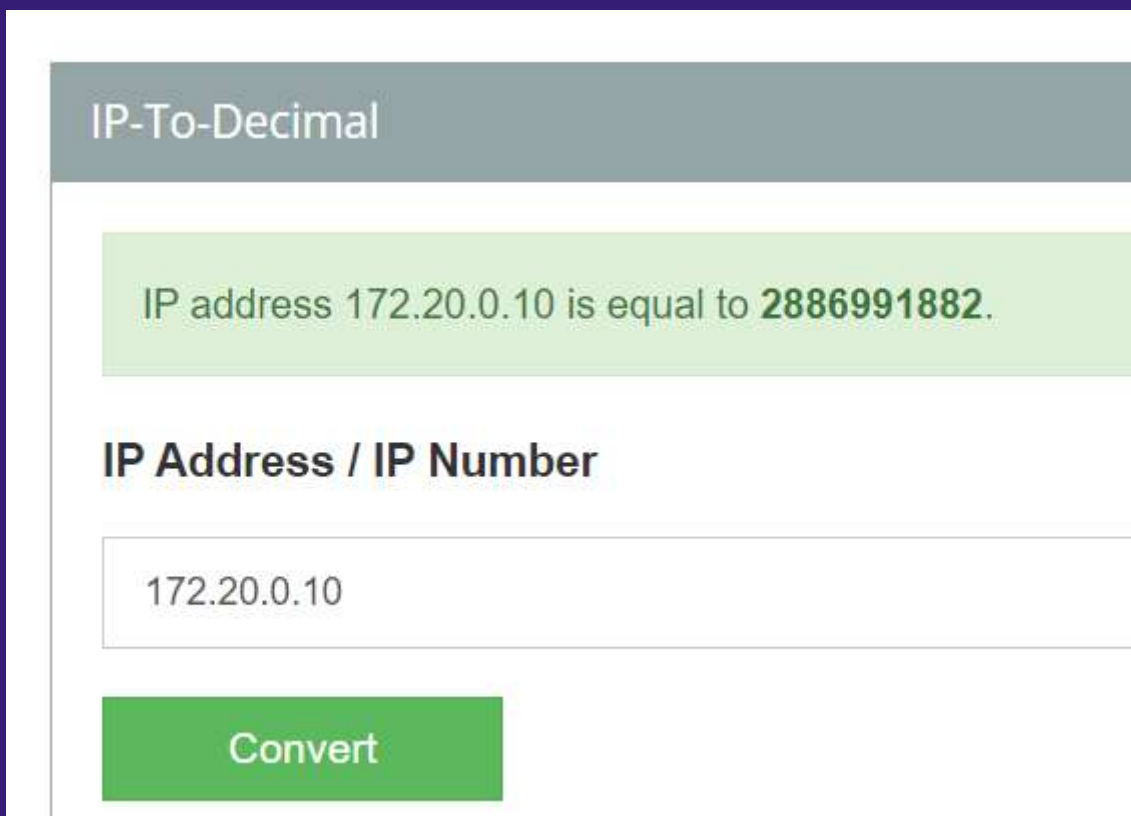
Dari sini kami menyadari bahwa tidak diperlukan SSTI to RCE, kita hanya perlu memanfaatkan fitur “Print a Website” untuk bisa mengakses URL internal tersebut (semacam SSRF to leak internal asset). Kami mencoba dan hasilnya adalah sebagai berikut.



Ternyata terdapat blacklist sehingga kita tidak bisa begitu saja langsung mengakses website tersebut. Setelah beberapa waktu mencari berbagai cara untuk bypass blacklist, kami menemukan referensi yang menarik yaitu <https://abdilahrf.github.io/cheatsheet/ssrf-openredirect-cheatsheet> yang mengajarkan kami untuk membypass dengan menggunakan IP conversion ke decimal.

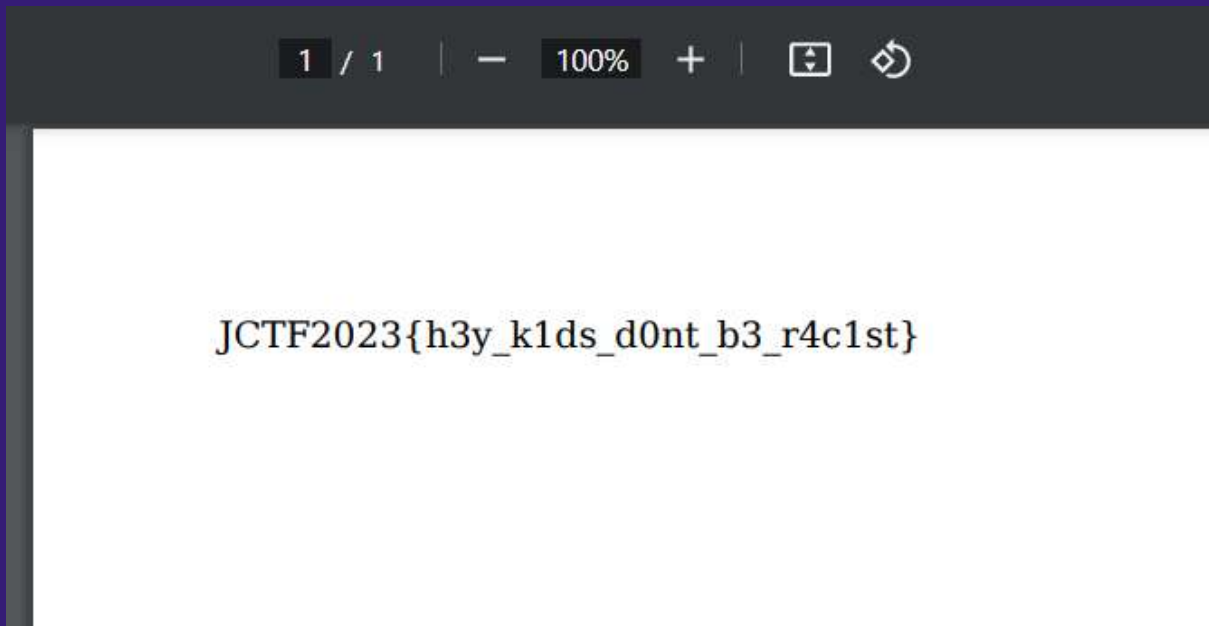
```
http://[0.0.0.0.111.127.0.0.1]  
With decimal IP location, really useful if dots are blacklisted  
http://0177.0.0.1/ --> (127.0.0.1)  
http://2130706433/ --> (127.0.0.1)  
http://3232235521/ --> (192.168.0.1)  
http://3232235777/ --> (192.168.1.1)  
With malformed URLs, useful when port is blacklisted
```

Kami mencoba untuk mencari tools untuk mengkonversi IP ke decimal dan kami mendapatkan tool ini <https://www.ipaddressguide.com/ip>. Kita langsung coba saja.



The screenshot shows a web interface for an "IP-To-Decimal" converter. At the top, the title "IP-To-Decimal" is displayed in a grey header. Below this, a green box contains the text "IP address 172.20.0.10 is equal to 2886991882." Underneath, the section "IP Address / IP Number" is shown. A text input field contains the IP address "172.20.0.10". At the bottom, there is a green button labeled "Convert".

Ternyata hasilnya adalah "2886991882", dirapikan sedikit dengan URL yang seharusnya maka akan menjadi "<http://2886991882:1234/flag6386236835>". Payload kita sudah siap, kita bisa langsung saja menggunakannya.

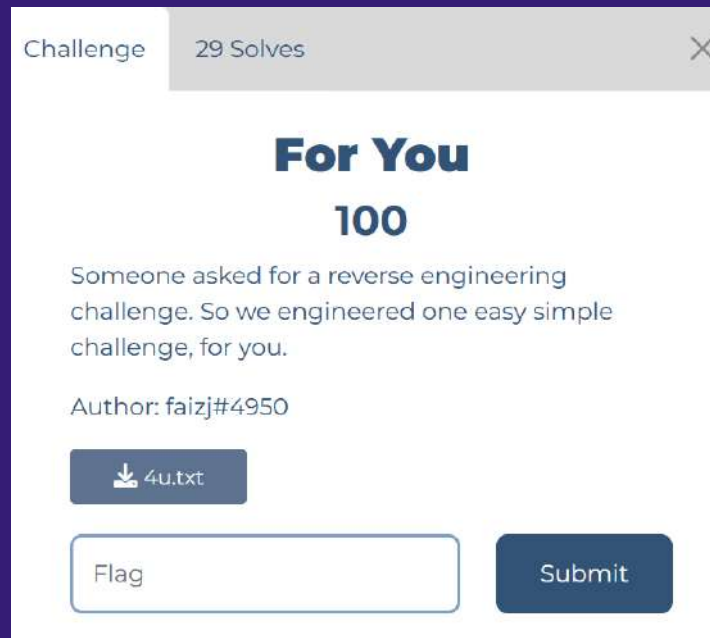


Ternyata bypass kami berhasil, akhirnya kami berhasil mendapatkan akses internal dan konten yang ada di dalamnya dengan memanfaatkan celah SSRF yang ada pada fitur tersebut. Dengan begitu, maka challenge ini telah selesai.

Flag = JCTF2023{h3y_k1ds_d0nt_b3_r4c1st}

REVENG

For You



Kita diberikan sebuah file txt yang isinya sebagai berikut:

```

1      0 LOAD_CONST          0 (0)
      2 LOAD_CONST          1 (None)
      4 IMPORT_NAME          0 (sys)
      6 STORE_NAME          0 (sys)

3      8 BUILD_LIST           0
      9 LOAD_CONST          2 (('2', '_', 'e', 'n', 'u', 's', '3', '3',
'n', 'n', 'T', 'C', '_', '_', '2', '0', 'r', 't', 'g', '1', '0', '_', 'j', 'h', 's',
'w', '{', '4', 'e', 'u', '3', 'y', '}', '_', '3', 'F', 'o', 'd', '_', 'e', 'j', 'i',
't'))
     12 LIST_EXTEND          1
     14 STORE_NAME          1 (s)

5     16 LOAD_CONST          3 (')
     18 LOAD_METHOD          2 (join)
     20 LOAD_NAME           1 (s)
     22 LOAD_CONST          1 (None)
     24 LOAD_CONST          1 (None)
     26 LOAD_CONST          4 (-1)
     28 BUILD_SLICE         3
     30 BINARY_SUBSCR
     32 CALL_METHOD          1
     34 STORE_NAME          1 (s)

7     36 LOAD_NAME           0 (sys)
     38 LOAD_ATTR            3 (stdout)
     40 LOAD_METHOD          4 (write)

```

```

42 LOAD_NAME          1 (s)
44 LOAD_CONST         5 (20)
46 BINARY_SUBSCR
48 CALL_METHOD        1
50 POP_TOP

8      52 LOAD_NAME          0 (sys)
      54 LOAD_ATTR          3 (stdout)
      56 LOAD_METHOD        4 (write)
      58 LOAD_NAME          1 (s)
      60 LOAD_CONST         6 (31)
      62 BINARY_SUBSCR
      64 CALL_METHOD        1
      66 POP_TOP

9      68 LOAD_NAME          0 (sys)
      70 LOAD_ATTR          3 (stdout)
      72 LOAD_METHOD        4 (write)
      74 LOAD_NAME          1 (s)
      76 LOAD_CONST         7 (32)
      78 BINARY_SUBSCR
      80 CALL_METHOD        1
      82 POP_TOP

```

...

Apabila dilihat dari sintaks-sintaks yang digunakan, file ini berisi section-section python bytecode. Untuk menyelesaikannya, kami menggunakan *approach* merekonstruksi kodingannya di python asli dengan module dis.dis.

```

solve.py > ...
1  import dis
2
3  def myfunc():
4      print('hallo world')
5
6
7  dis.dis(myfunc)
8

```

PROBLEMS	OUTPUT	TERMINAL	DEBUG CONSOLE
PS D:\ctf\joints\2023\for you> python -u "d:\ctf\joints\2023\for you\solve.py"			
4	0 LOAD_GLOBAL	0 (print)	
	2 LOAD_CONST	1 ('hallo world')	
	4 CALL_FUNCTION	1	
	6 POP_TOP		
	8 LOAD_CONST	0 (None)	
	10 RETURN_VALUE		

Cara merekonstruksinya adalah dengan melihat opcode yang tercantum di bagian kiri (LOAD_GLOBAL, LOAD_CONST, dll.), menghubungkan dengan value di kanan (0, None, sys, dll.) dan mencocokkannya dengan syntax python apa yang memiliki fungsi seperti itu. Contohnya instruksi 1 mengandung IMPORT_NAME dan diikuti dengan “sys”, artinya ada code **import sys**.

Kemudian di section 3 terdapat serangkaian value ascii dan diikuti dengan instruksi BUILD_LIST, STORE_NAME, artinya ada deklarasi list **s** yang berisi value ascii tersebut. Kemudian di section 5 list tersebut **s** tadi dipanggil, ada instruksi BINARY_SUBSCR yang biasa untuk indexing, ditambah index -1. Artinya yang paling masuk akal adalah list tersebut di-reverse urutannya (**s = s[::-1]**)

The screenshot shows a Python IDE with a file named `solve.py` and its disassembly output in the terminal.

solve.py

```

1  import dis
2
3  def myfunc():
4      import sys
5      s = ('2', '_', 'e', 'n', 'u', 's', '3', '3', 'n', 'n', 'T', 'C', '_', '_', '2', '0', 'n', 't', 'g',
6          '1', '0', '_', 'j', 'h', 's', 'w', '{', '4', 'e', 'u', '3', 'y', '}', '_', '3', 'F', 'o', 'd',
7          ' ', '_', 'e', 'j', 'i', 't')
8      s = ''.join(s)[::-1]
9
10 dis.dis(myfunc)
11 # myfunc()
12

```

Terminal Output (Disassembly):

```

PS D:\ctf\joints\2023\for you> python -u "d:\ctf\joints\2023\for you\solve.py"
4      0 LOAD_CONST          1 (0)
      2 LOAD_CONST          0 (None)
      4 IMPORT_NAME          0 (sys)
      6 STORE_FAST          0 (sys)

5      8 LOAD_CONST          2 (('2', '_', 'e', 'n', 'u', 's', '3', '3', 'n', 'n', 'T', 'C', '_', '_', '2', '0', 'n', 't', 'g',
2'  2', '0', 'n', 't', 'g', '1', '0', '_', 'j', 'h', 's', 'w', '{', '4', 'e', 'u', '3', 'y', '}', '_', '3', 'F', 'o', 'd',
', ' ', '_', 'e', 'j', 'i', 't'))
      10 STORE_FAST          1 (s)

6     12 LOAD_CONST          3 (')
      14 LOAD_METHOD          1 (join)
      16 LOAD_FAST             1 (s)
      18 CALL_METHOD           1
      20 LOAD_CONST          0 (None)
      22 LOAD_CONST          0 (None)
      24 LOAD_CONST          4 (-1)
      26 BUILD_SLICE           3
      28 BINARY_SUBSCR
      30 STORE_FAST          1 (s)

```

Kemudian di Section 7 dan seterusnya memiliki instruksi yang saling serupa, dimana ada index tertentu yang diambil dari list **s** tadi kemudian di `sys.stdout.write`.

Dari informasi yang kita dapatkan tadi, kami menyimpulkan bahwa list s tadi merupakan flag yang discrambled dan direverse, kemudian stdout per index tadi menunjukkan urutan karakter flag aslinya. Maka dari itu, kita bisa menyelesaikannya dengan script sederhana berikut:

```
import dis

def myfunc():
    import sys
    s = ('2', '_', 'e', 'n', 'u', 's', '3', '3', 'n', 'n', 'T', 'C', '_', '_','2', '0', 'r', 't', 'g', '1', '0', '_', 'J', 'h', 's', 'w', '{', '4', 'e', 'u', '3', 'y', '}', '_', '3', 'F', 'o', 'd', '_', 'e', 'j', 'i', 't')
    s = ''.join(s[::-1])

    c = [20,31,32,7,28,22,28,8,16,17,8,4,2,13,18,0,4,3,33,24,1,33,8,8,26,3,5,4,0,19,23,18,4,22,33,3,4,15,4,11,6,13,10]

    for i in c:
        print(s[i], end="")

# dis.dis(myfunc)
myfunc()
```

Output =

```
PS D:\ctf\joints\2023\for you> python -u "d:\ctf\joints\2023\for you\solve.py"
JCTF2023{w3_just_engin33red_th1s_one_4_you}
PS D:\ctf\joints\2023\for you>
```

Flag = JCTF2023{w3_just_engin33red_th1s_One_4_you}

HAHA HOHO AWIKWOK GG GEMING

BINEX



CRYPTO

Easy CBC

Challenge
47 Solves

Easy CBC

100

Whoa, do you know that you can encrypt an image and make it like nonsense? anyway, recently I heard about this AES-CBC encryption and I try to use it to encrypt an image.

author: Arif ('saj#6550)

<https://drive.google.com/drive/folders/1os7my96amOGnp2jmdCusYolOH53lV3Vh?usp=sharing>

Flag
Submit

Summary

Kita diberikan file one_time_password.txt yang berisi:

```
# !pip install certifi==2021.10.8
# !pip install cffi==1.15.0
# !pip install cryptography==36.0.2
# !pip install Pillow==9.0.1
# !pip install wincertstore==0.2
import os
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.backends import default_backend
import PIL.Image as Image

class CBCEncryption:
    def __init__(self, key, iv):
        self.cipher = Cipher(algorithms.AES(key), modes.CBC(iv),
backend=default_backend())
        self.encryptor = self.cipher.encryptor()

    def encrypt(self, image):
        return self.encryptor.update(image)

    def finalize_encrypt(self):
        return self.encryptor.finalize()
```

```

def EncryptImage(encryption, image, output):
    output = output + '.bmp'
    image = Image.open(image)
    image.save('temp.bmp')
    with open('temp.bmp', 'rb') as reader:
        with open(output, 'wb') as writer:
            image_data = reader.read()
            header, body = image_data[:54], image_data[54:]
            body += b'\x35' * (16 - (len(body) % 16))
            body = encryption.encrypt(body) + encryption.finalize_encrypt()
            writer.write(header + body)
            writer.close()
            reader.close()
    os.remove('temp.bmp')

def main():
    key = b'JOINTSCTF2023'
    key = key.ljust(32, b'\x35')

    iv = key[:16]
    iv = bytearray(iv)
    for i in range(16):
        iv[i] = iv[i] ^ 0x35
    iv = bytes(iv)

    AesCbc = CBCEncryption(key, iv)
    EncryptImage(encryption=AesCbc, image='flag.jpg', output='out')

if __name__ == '__main__':
    main()

```

Code ini akan memproses sebuah image dengan cara mengencryptnya menggunakan AES CBC dengan module **cryptography.hazmat**. Akan tetapi, bagian yang diencrypt hanyalah body nya saja (index 54 hingga akhir), bukan bagian header. Objektif kita adalah untuk merecover file **flag.jpg** yang sudah diencrypt menjadi **out.bmp**

Solution

Untungnya, key dan iv yang digunakan untuk mengencrypt diberitahu dan sifatnya statik sehingga kita tinggal membuat code decryptornya saja. Untuk membuat decryptornya, kita bisa merefer ke dokumentasi

`class cryptography.hazmat.primitives.ciphers.Cipher(algorithm, mode)` [\[source\]](#)

Cipher objects combine an algorithm such as `AES` with a mode like `CBC` or `CTR`. A simple example of encrypting and then decrypting content with AES is:

```
>>> import os
>>> from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
>>> key = os.urandom(32)
>>> iv = os.urandom(16)
>>> cipher = Cipher(algorithms.AES(key), modes.CBC(iv))
>>> encryptor = cipher.encryptor()
>>> ct = encryptor.update(b"a secret message") + encryptor.finalize()
>>> decryptor = cipher.decryptor()
>>> decryptor.update(ct) + decryptor.finalize()
b'a secret message'
```

Parameters:

- **algorithm** – A `CipherAlgorithm` instance such as those described [below](#).
- **mode** – A `Mode` instance such as those described [below](#).

Raises: `cryptography.exceptions.UnsupportedAlgorithm` – This is raised if the provided `algorithm` is unsupported.

Sehingga solvernya adalah seperti berikut:

```
# !pip install certifi==2021.10.8
# !pip install cffi==1.15.0
# !pip install cryptography==36.0.2
# !pip install Pillow==9.0.1
# !pip install wincertstore==0.2
import os
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.backends import default_backend
import PIL.Image as Image

class CBCEncryption:
    def __init__(self, key, iv):
        self.cipher = Cipher(algorithms.AES(key), modes.CBC(iv),
backend=default_backend())
        self.encryptor = self.cipher.encryptor()
        self.decryptor = self.cipher.decryptor()

    def encrypt(self, image):
        return self.encryptor.update(image)

    def finalize_encrypt(self):
        return self.encryptor.finalize()

    def decrypt(self, image):
        return self.decryptor.update(image)

    def finalize_decrypt(self):
```



```

        return self.decryptor.finalize()

def EncryptImage(encryption, image, output):
    output = output + '.png'
    image = Image.open(image)
    image.save('temp.png')
    with open('temp.png', 'rb') as reader:
        with open(output, 'wb') as writer:
            image_data = reader.read()
            header, body = image_data[:54], image_data[54:]
            body += b'\x35' * (16 - (len(body) % 16))
            print(body[54:60])
            body = encryption.encrypt(body) + encryption.finalize_encrypt()
            print(body[54:60])
            writer.write(header + body)
            writer.close()
            reader.close()
    os.remove('temp.png')

def DecryptImage(encryption, image, output):
    image = Image.open(image)
    with open('out.bmp', 'rb') as reader:
        with open('result.jpg', 'wb') as writer:
            image_data = reader.read()
            header, body = image_data[:54], image_data[54:]
            body = encryption.decrypt(body) + encryption.finalize_decrypt()
            writer.write(header + body)
            writer.close()
            reader.close()

def main():
    key = b'JOINTSCTF2023'
    key = key.ljust(32, b'\x35')
    print(key)
    iv = key[:16]
    iv = bytearray(iv)
    for i in range(16):
        iv[i] = iv[i] ^ 0x35
    iv = bytes(iv)
    print(iv)
    AesCbc = CBCEncryption(key, iv)
    DecryptImage(encryption=AesCbc, image='test.jpg', output='outcoba')

if __name__ == '__main__':
    main()

```

```
$ python3 solve.py  
b'JOINTSCTF202355555555555555555555'  
b'\x7fz|\{afvas\x07\x05\x07\x06\x00\x00\x00'
```



Flag = JCTF2023{n4rim0_in9_pAndum}


Rumah Sakit Akademik UGM

Challenge
15 Solves

Rumah Sakit Akademik UGM

152

Motto RSA UGM adalah "Friendly and Caring Hospital", merupakan komitmen mewujudkan rumah sakit yang benar-benar nyaman, sejuk, penuh keramahan dalam pelayanan, dan menghadirkan nuansa yang menunjang kesembuhan pasien. Layaknya rumah sakit lain, rumah sakit ini memiliki kapasitas ~~prima~~ pasien yang terbatas. Akan tetapi, rumah sakit akan berusaha semaksimal mungkin untuk memberikan kapasitas yang banyak guna menangani keadaan darurat seperti pandemi dan hal darurat lainnya.



Summary

Untuk soal ini, kami diberikan soal RSA yang diketahui p , q , cipher, dan e nya dengan nilai seperti ini

```

n: 203383993427357335284883156595733536612645021482108763525506366401606175676147
20676211740243261369072562649458706039877325974791302609240527473013872722597762
18135303666037752417509198891603025469907896585557793864192713591502663176537748
90530301714686506598673911290308530852433524465049883064446117835975274098552819
18563323211076644380201429498883029285625519870509616205840827085882720857004167
84701862117857228095744093556610446417523465917557756352761412746797634625664245
27431696386409498528902613341041001370105599926554336449201639254053730686185468
54682806457454599793641604049393565690125718170974354170907
c: 849400943451868140971077164145789766006790044519056660284139788505127148182796
76719653310085131656902255355736035479477211264471281929345056458175271701322926
29869065767408690933338515031092201300156582261678705431466638561311054299305788
62039428462988841916999110966752852120418314443654577847194212981065790051458601
99643053090630725979940080153749717577340458855104914903065712838947093226010022
07304421926088033578315621070144548559443786751641075443667054917600090888876679
21817785252503806301202554686991115432692420596006479492088528469050134857693450
510469386318763777557436708900297175077459690397155760691
e: 65537

```

n:178808683674731212345503393374370414126806502691588012847430048030885333694319
 12492331334265639012958821412749786557157342550849324006021843531788783331482343
 06755658642505766810999468255437507328103549630028844984787742271808495819368625
 18961281213270400788877269062820776988588429707363467965954351412835588695585146
 75415075754345490292327177001579881275004609169245576653776817570752059355238200
 54585232837290829040881631156704229186755511765240566437905617966027756782026543
 88617527865718074803492960866173935973017822175129545067112292644690275681173464
 74979016155257331809081708000761716701909996463121201618523
 c:106268274386120889373580453307475028823290098539713855308784352030383912506524
 26520293742128762218947360644192134544296762539985782027554330463795836769665810
 96163290513631293897998395199458175238700703661349968148283125536764412617155069
 90606964839347633083446887059371324838219034152317749262380121498634106197920645
 57379445886897751946278954362963003742850761749514008496231044204421562632049264
 6769289975694277677465962807451063295905887267988139744538811910388765565552553
 36603693905253703107577748451524978422573888074360860581077852781783621960422375
 18332699654890164322760050949745321411993390276040114565038
 e:65537

n:257996953616501988078225199527173992386861114350951056723920594315303297784786
 35497957240790054201022277703682287407607481031333199984897692372472432698370710
 10854478686132652920075565595708517732771114451758861088446018508300873385079037
 22020762872007489162315582923828013188329874517059992848664900044525366359985617
 11076136327113362831867480426314256472610908311182683921657174278744681868980046
 98473130873413649406596463889472076237840847331118183603150466674972946074235984
 28981860049745552912110269767308778173357641952068465529348523913427069733122496
 95922854535562552657833796980352020608738659472586587365817
 c:238217650424493458675382241700971543025385941720857816281343140690670158685258
 26500507313280321452213852819867942197038668546609118806532038815662358769402727
 72736236279589628966997638917472919134553678421640812330347699892110146715059503
 80921742620716803816559844042539603305745390646740367386285187767548668881628961
 31948661665928036982357656937911602083295106827967054420659924486407797366485173
 84742290200971919503325485763365858884696720499627466416096337999707288291449500
 07130145449602940049859971638862279639671491494561825801978970341095659776179486
 10494614392264946583963198021752446404919871541871914731827
 e:65537

n:205309817716158372748102197531666795732664726927750170987806106581577166517176
 39489927078647888507579146066131295127215552749070066281687950244394787700898318
 94939041352697348508880803254717417991333846587937126748262095517316914657574784
 08816801351995096103835942116883468480924646490917128441692677898972790437329828
 21975816676254573463990453001548668653796255925852899023975885795914028774311805
 06842468184210395463206603279107764313823135799560658892368641916432278828253293
 73945274992167837532784960595004549337242976604611964741159601329364824098851240
 02594240887960369864461226794302340667133735814171347665447
 c:518865159291465226525778710386386494268696443263157861425240137503306786343443
 99803307222444485469944095410719701120096678929713052914876554741796488992939734
 54267603322109915450201222431101789322138738727729872971590920014581134393802839
 43404302535790498366078455913440797533144398370090761248732906886337917213231286
 06773049026121162750179503503802625169258091523401333493641377438131212375453458
 82895706581529461663874441123989042554609111586680210814240806135864439482408097

```

41033188191207344612332961247947111209452110528697705784702321083812493094737023
2578969215145423499581653567104815921889097206281265383742
e:65537

n:955805696610149009777322833076477402037155289112201117576420375549861036072358
78754471547419410169681267460481344301273591387003967664056197513071733185757503
43140245767181422642276133835172215094972213556106879574657964525847051388253717
80630863471340890289176710702832139932658769982899365933283166177840196747518767
83335646432432221398473062617518278224565027543827712302206183497993004263759305
19118026159738045657125476302251275568188607694659952576865383999409881617842355
43702808326759081535657973280087049886793073698072199863437026342178156503366521
7878135278846122154318996021540829617121927424990081151771
c:660012443700489014316513881029380698155854373744035495637249707469865730920019
51204797691558056652514209118345561407707434429631429323035900146300855515622086
18343502641784093917216954672690269586771067987918555354031782846815769749151439
45396586874362630437356539899167006856021808524315092091325519880695114986911171
47480338393672372698716467169372797894548100695279085338243967601134363808615256
32663809430669520067261079689791632231290571177197049650342583937703058151506814
83591281791261419337232780010420754910282528823397651063165106555151291720503147
525205316698901559432365880588477333554574282127584635156
e:65537

```

Kita diberikan sebuah file flag.enc berisi kumpulan n, c, dan e sebanyak ratusan value tanpa diberikan script encryptnya. Bisa dilihat bahwa n dan c yang ditampilkan tiap grupnya memiliki value yang berbeda-beda, namun eksponen yang digunakan selalu sama. Maka dari itu, kami menyimpulkan bahwa plaintext flag diencrypt dengan modulus yang berbeda-beda dan hasilnya ditampilkan di sini.

Solution

Awalnya kami berpikir bahwa ini merupakan challenge bertemakan hastad broadcast & chinese remainder theorem karena melibatkan multiple modulus namun eksponen sama. Namun eksponen yang bernilai 65537 terlalu tinggi sehingga nampak tidak memungkinkan untuk kami menggunakan approach tersebut.

Dari membaca deskripsi, kami mendapat informasi bahwa kapasitas prima yang digunakan adalah terbatas. Artinya, dari modulus-modulus beraneka ragam tersebut kita akan mengecek apakah ada 2 modulus yang memiliki faktor prima yang sama. Kita dapat melakukannya dengan mengecek value GCD kedua modulus tersebut dan jika nilainya bukan 0,

maka GCD itulah faktor primanya. Apabila faktor prima ditemukan, tinggal menghitung faktor prima lainnya dengan $n//p$ dan mengconstruct private key untuk mendecrypt.

Dengan memakai permutasi untuk memfilter 2 modulus yang berbeda, beginilah solvernya

```
from Crypto.Util.number import *
import math
import itertools

file = open('flag.enc', 'r').read().splitlines()
Ns = []
Cs = []
Es = []
for i in range(0, len(file), 4):
    print(i)
    n = int(file[i][2:])
    c = int(file[i+1][2:])
    Ns.append(n)
    Cs.append(c)

idx = [i for i in range(len(Ns))]
for i in itertools.permutations(idx, 2):
    a, b = i
    z = math.gcd(Ns[a], Ns[b])
    if z != 1:
        print('a', a, 'b', b, z)
        break

n = Ns[a]
c = Cs[a]
p = z
q = n//p
phi = (p-1)*(q-1)
e = 65537
d = inverse(e, phi)
print(long_to_bytes(pow(c, d, n)))
```

Output =

```
1900
1912
1916
1920
1924
1928
1932
1936
1940
1944
1948
1952
1956
1960
1964
1968
1972
1976
1980
1984
1988
1992
1996
a 450 b 499 175529706714319325596645062750171641971655662684519431846937552223674406742958821032317047697110170332913179386558300117209404069878374719
362813949395601028349000879775399354014812184080581901596248432314704124752618618921317241567998664346926098095517225908947470737641
b'JCTF2023{d0nt_r3us3_y0ur_pr1m3s_4g41n_4nd_4g41n}'
```

Flag

=

JCTF2023{d0nt_r3us3_y0ur_pr1m3s_4g41n_4nd_4g41n}

FORENSIC

Dinosaur

Challenge 32 Solves

Dinosaur

100

The stegosaurus is one of the few creatures that likes to eat blowfish. The key of its favorable taste to a blowfish is dinosaur. It initializes his day by using blowfish. Although it wasn't the best food of the prehistoric era, the stegosaurus always leaves a FeedBack which until now, is still a Cipher for historians to crack. No phrases were used by historians to describe the extinct dinosaur.

By the way, stegosaurus likes to hide.
Stegosaurus... hide?

Author: Giga - Infinicus#6867

https://drive.google.com/file/d/1ymEPI2oZOLubN3VD8SKusp=share_link

Soal diatas memberikan sebuah foto berikut

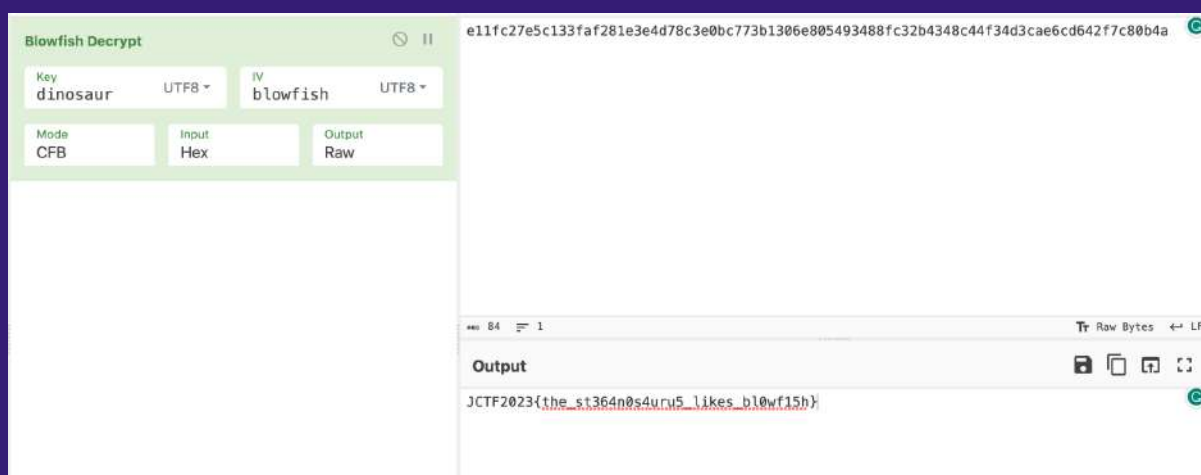


Berdasarkan deskripsi yang diberikan, terdapat clue bahwa challenge forensic kali ini merupakan challenge steganography. Dari clue tersebut, kami menggunakan tools **stegseek** untuk brute-force gambar tersebut. Sama juga seperti di deskripsi, tidak diperlukan phrase apapun untuk mengeluarkan steganography dari gambar ini.

```
clints@Clintswoods-MBP Joints % steghide extract -sf stegosaurus.jpg
Enter passphrase:
wrote extracted data to "insides_of_stegosaurus.txt".
clints@Clintswoods-MBP Joints % cat insides_of_stegosaurus.txt
e11fc27e5c133faf281e3e4d78c3e0bc773b1306e805493488fc32b4348c44f34d3cae6cd642f7c80b4a%
```

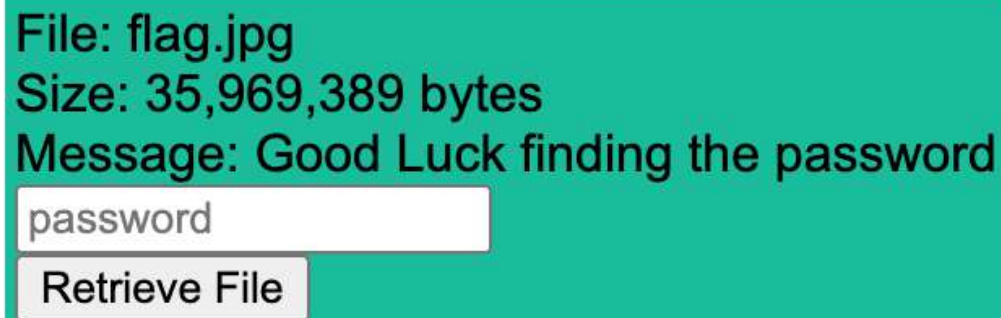
Setelah itu, kami mendapatkan sebuah file bernama “insides_of_stegosaurus.txt” yang isinya merupakan sebuah encrypted strings. Berdasarkan deskripsi soal, terdapat beberapa hal yang di-highlight seperti FeedBack dan Cipher yang dimana merujuk kepada cfb (cipher feedback) dan juga blowfish yang menandakan bahwa metode yang digunakan adalah blowfish (bf-cfb).

Dengan hint “*The key of its favorable taste to a blowfish is dinosaur. It initializes his day by using blowfish.*” Kami mendapatkan bahwa kunci yang digunakan adalah “dinosaur” dan IV (Initialized Vector) adalah “blowfish”. Dengan menggunakan cyberchef, kami berhasil mendapatkan flag dari informasi yang telah dikumpulkan.



Flag = JCTF2023{the_st364n0s4uru5_likes_bl0wf15h}

Terdapat sebuah strings panjang pada function retrieve. Apabila kami buka file tersebut pada sebuah browser akan memunculkan interface seperti berikut.



File: flag.jpg
 Size: 35,969,389 bytes
 Message: Good Luck finding the password

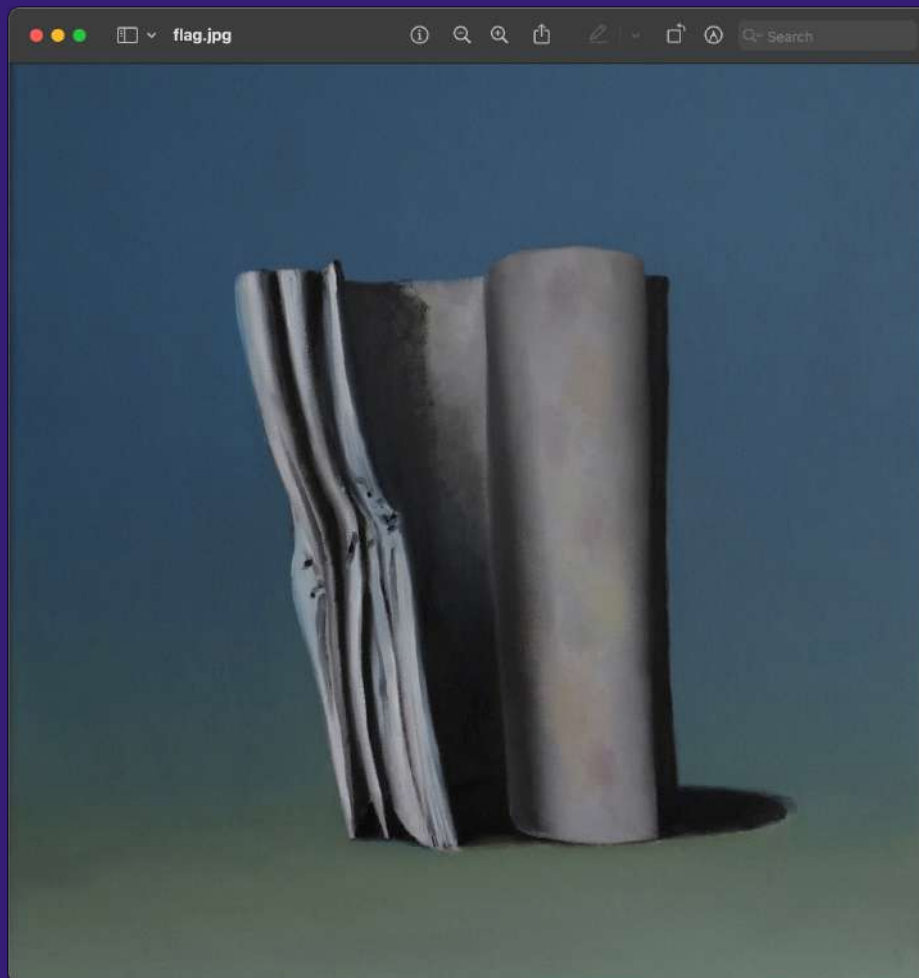
Generated by dundorma

Diminta sebuah password untuk mendownload flag.jpg tersebut. Ketika file html tersebut kami gunakan terhadap command **strings**, kami mendapatkan secret dalam bentuk base64 pada tag small.

```
<table border=0 style='background: #1abc9c'>
<tr>
  <td>
    File: flag.jpg
    <br>
    Size: 35,969,389 bytes
    <br>
    Message: Good Luck finding the password
    <br>
    <input type=password id=passwordid placeholder=password>
    <br>
    <button onclick=retrieve()>Retrieve File</button>
  </td>
</tr>
</table>
<br>
<br>
<br>
<small text="c3VwZXJzZWNyZXRwYXNzd29yZA==">Generated by dundorma</small>
</body>
</html>
```

```
clints@Clintswoods-MBP Joints % echo "c3VwZXJzZWNyZXRwYXNzd29yZA==" | base64 -d
supersecretpassword%
```

Kami mendapatkan password “supersecretpassword” ketika di-decode dengan base64. Oleh karena itu, kami dapat mendownload flag.jpg tersebut dan apabila dibuka, fotonya seperti berikut.



Setelah menganalisis gambar tersebut lebih lanjut, kami menyadari bahwa terdapat file wav dan txt ketika kami menggunakan tools **binwalk**.

```
clints@Clintswoods-MBP Joints % binwalk flag.jpg
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, JFIF standard 1.01
150185	0x24AA9	Zip archive data, at least v2.0 to extract, compressed size: 35818888, uncompressed size: 37462166, name: flag.wav
35969111	0x224D857	Zip archive data, at least v2.0 to extract, compressed size: 110, uncompressed size: 151, name: hint.txt
35969367	0x224D957	End of Zip archive, footer length: 22


```

clints@Clintswoods-MBP Joints % binwalk -e flag.jpg

```

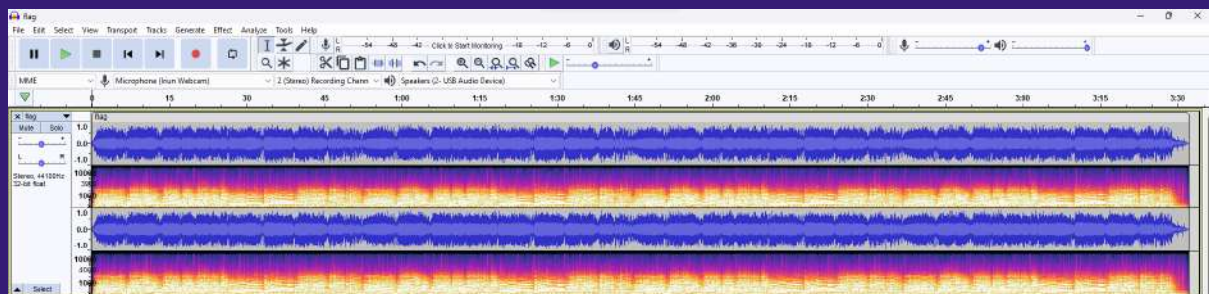
DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, JFIF standard 1.01
150185	0x24AA9	Zip archive data, at least v2.0 to extract, compressed size: 35818888, uncompressed size: 37462166, name: flag.wav
35969111	0x224D857	Zip archive data, at least v2.0 to extract, compressed size: 110, uncompressed size: 151, name: hint.txt
35969367	0x224D957	End of Zip archive, footer length: 22

```

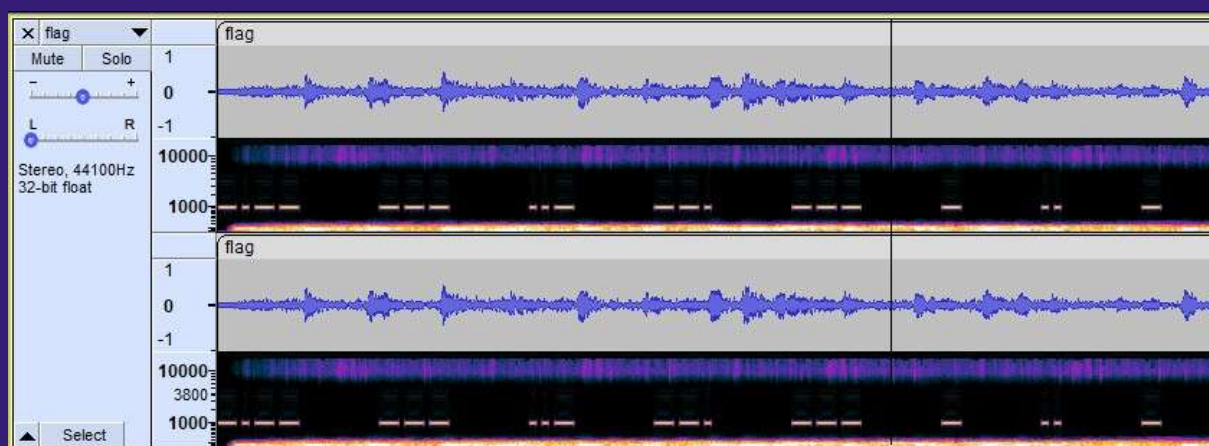
clints@Clintswoods-MBP Joints % cd _flag.jpg.extracted
clints@Clintswoods-MBP _flag.jpg.extracted % ls
24AA9.zip      flag.wav      hint.txt
clints@Clintswoods-MBP _flag.jpg.extracted % cat hint.txt
listen to flag.wav. It's supposed to be mono, but the left and right channels are slightly different. Figure out what's the difference and get the flag%
clints@Clintswoods-MBP _flag.jpg.extracted %

```

Setelah meng-extract menggunakan binwalk dan juga melihat hint, kami mengetahui bahwa flagnya terdapat pada flag.wav. Oleh karena itu, kami menggunakan tools **audacity** untuk analisis lebih lanjut.

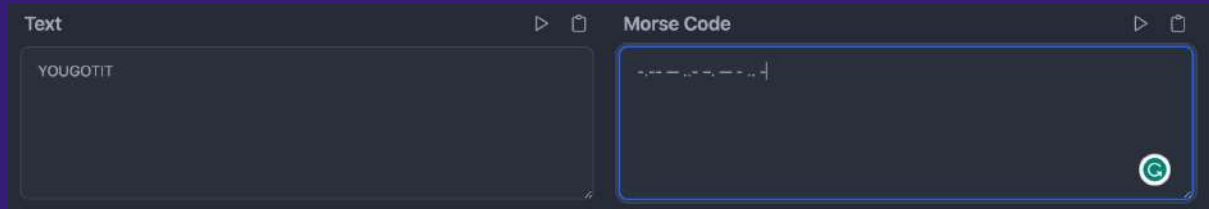


Dikarenakan ramainya suara pada file wav ini, kami menggunakan bantuan effects “*Vocal Reduction and Isolation*” dari audacity untuk menurunkan suara vocal yang terdapat pada wav tersebut sehingga menghasilkan wav seperti dibawah ini.



HAHA HOHO AWIKWOK GG GEMING

Terdapat kode morse pada wave spectrogram, setelah menerjemahkan sandi morse tersebut ke dalam alphabet, kami mendapatkan strings “YOUGOTIT”. Dengan ini, kami menambahkan format flag JCTF2023{.*} dan kami mendapatkan flagnya.



Flag = JCTF2023{YOUGOTIT}

Spartan Ghosts

Challenge 9 Solves X

Spartan Ghosts

588

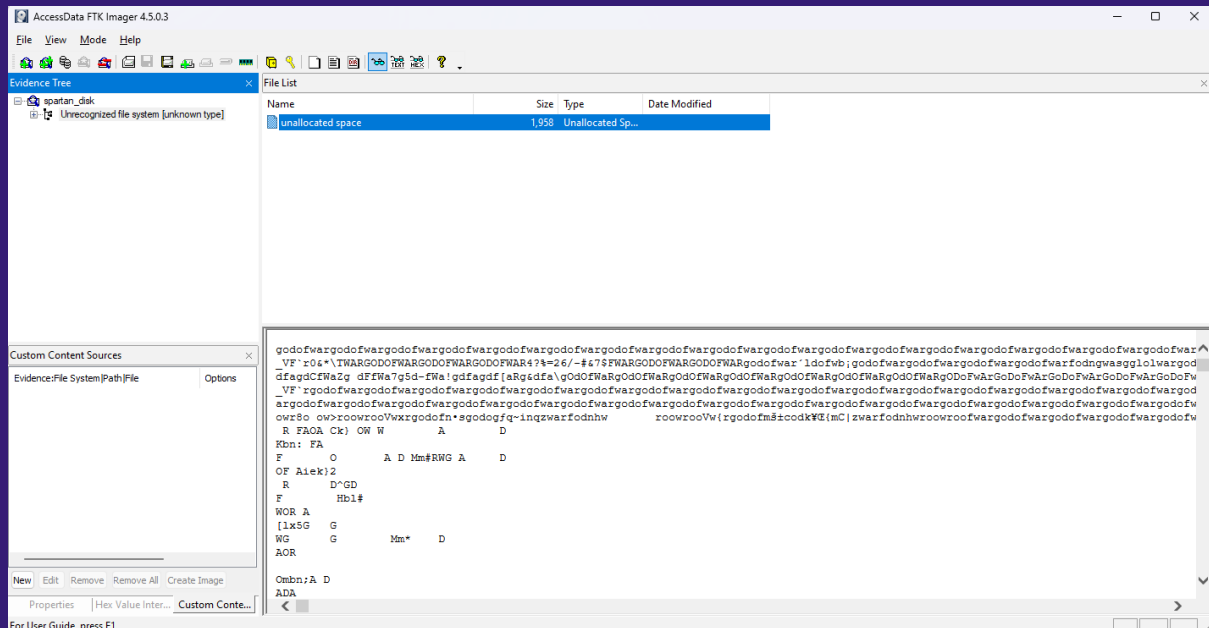
I heard some rumors about a demigod being worshipped by almost all of Greek. I heard he was from Sparta, a land of fierce warriors. Although, something bad happened there. Jota went there to find out more about this demigod, but there was nothing there. Jota only found this disk, which he believes is either encrypted, or corrupted. There was a label which says "EXCLUSIVE FOR [unreadable] ONLY. Contains the screams of the people that once lived here."

Extract more information about this mysterious thing. Note: flag dalam lowercase.

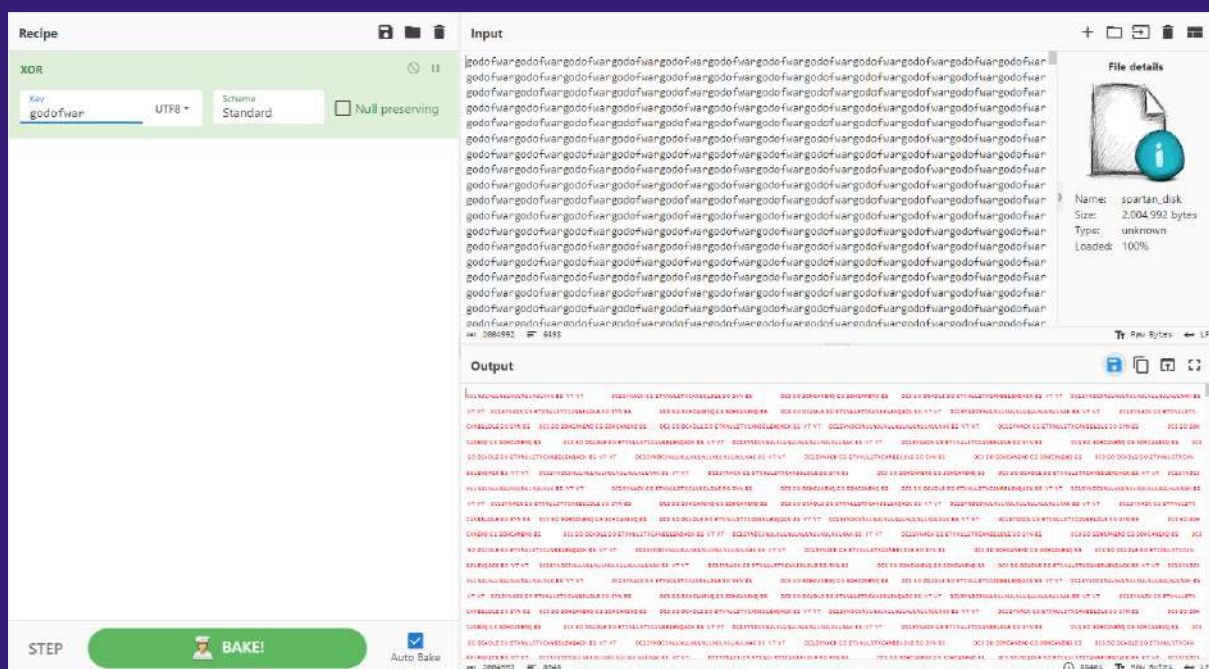
Author: Giga - Infinicus#6867

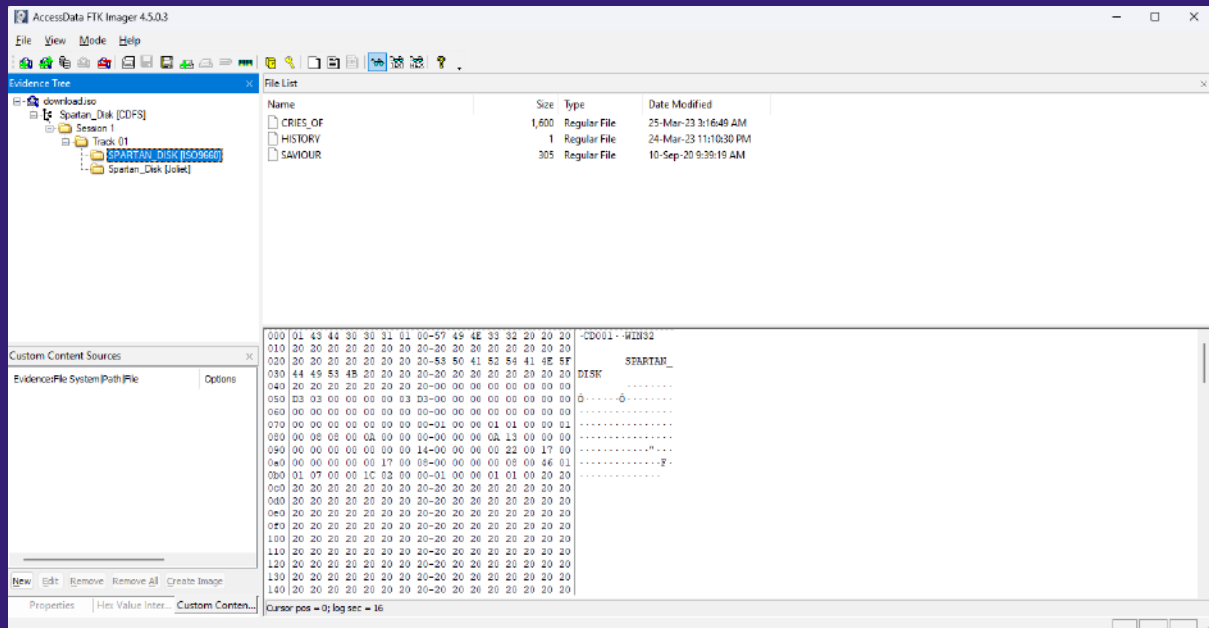
https://drive.google.com/file/d/1s9QctiunWeiNgiD3Dw8eusp/share_link

Pada challenge ini, kami diberikan sebuah disk image file bernama spartan_disk. Untuk menganalisa disk image seperti ini, kami menggunakan **FTK Imager** untuk melihat konten yang terdapat pada disk image tersebut.



Namun ketika kami buka menggunakan tools FTK Imager, kami menemukan bahwa isi yang ada di dalam image tersebut encrypted sama seperti keterangan yang diberikan pada soal. Berdasarkan deskripsi soal, terdapat hint EXCLUSIVE FOR yang dimana kami berasumsi kami dapat men-decrypt disk image ini dengan menggunakan XOR. Pada isi encrypted disk yang kami analisa, terdapat repeated strings yaitu “godofwar”. Kami kemudian mencoba untuk XOR file image disk ini dengan string “godofwar”.





Dengan meng-XOR file image_disk dengan strings “godofwar”, kami berhasil mendekripsi file tersebut dan mendapatkan 3 file di dalamnya.

```

===== cries_of_sparta
ExifTool Version Number      : 12.57
File Name                    : cries_of_sparta
Directory                   : .
File Size                    : 1638 kB
File Modification Date/Time  : 2023:03:25 10:16:49+07:00
File Access Date/Time       : 2023:04:16 12:15:09+07:00
File Inode Change Date/Time  : 2023:04:16 12:15:09+07:00
File Permissions             : -rwxrwx-rw-
File Type                    : MP3
File Type Extension          : mp3
MIME Type                    : audio/mpeg
MPEG Audio Version           : 1
Audio Layer                  : 3
Sample Rate                  : 44100
Channel Mode                  : Single Channel
MS Stereo                    : Off
Intensity Stereo             : Off
Copyright Flag               : False
Original Media               : True
Emphasis                     : None
VBR Frames                   : 8236
VBR Bytes                   : 1638080
VBR Scale                    : 80
Encoder                      : LAME3.100.0.
ID3 Size                     : 205
Artist                       : ?????
Comment                     : ?????
Comment (xxx)               : ?????
Title                       : ?????
Year                        : ?????
Warning                      : [minor] Frame 'TDRC' is not valid for this ID3 version
Recording Time               : ?????
Album                       : ?????
User Defined Text            : (TXXX) ?????
Track                       : ?????
Genre                       : ?????
Date/Time Original          : ?????
Audio Bitrate                : 60.9 kbps
Duration                    : 0:03:35 (approx)

```

```
===== saviour
ExifTool Version Number      : 12.57
File Name                    : saviour
Directory                    : .
File Size                    : 312 kB
File Modification Date/Time  : 2020:09:10 16:39:19+07:00
File Access Date/Time       : 2023:04:16 12:15:50+07:00
File Inode Change Date/Time  : 2023:04:16 12:15:09+07:00
File Permissions             : -rwxrw-rw-
File Type                    : JPEG
File Type Extension         : jpg
MIME Type                    : image/jpeg
JFIF Version                 : 1.01
Resolution Unit              : None
X Resolution                  : 1
Y Resolution                  : 1
Exif Byte Order              : Big-endian (Motorola, MM)
Orientation                  : Horizontal (normal)
Color Space                   : sRGB
Exif Image Width             : 1003
Exif Image Height            : 1280
Image Width                  : 627
Image Height                 : 800
Encoding Process              : Baseline DCT, Huffman coding
Bits Per Sample              : 8
Color Components              : 3
Y Cb Cr Sub Sampling         : YCbCr4:2:0 (2 2)
Image Size                   : 627x800
Megapixels                   : 0.502
```

Ketika kami mencoba untuk menganalisa ketiga filenya, 1 file merupakan txt, 1 file mp3, dan 1 file jpg. Setelah menganalisis file jpgnya, kami tidak mendapatkan lead apa-apa dan juga txt yang tidak mempunyai secret apapun.

Ketika menganalisa file mp3nya, kami menyadari bahwa pada mp3 file `cries_of_sparta` merupakan slow-motion mp3 yang mengeja masing-masing alfabet dari flag dimulai dengan JCTF2023. Dengan menggunakan tools fast-forward yang disediakan online seperti <https://mp3cut.net/id/change-speed>, kami mendapatkan flag challenge ini dari wav tersebut.

Flag = JCTF2023{dream_on_kratos}

MISCEL

Mega Sus

Challenge 43 Solves X

Mega SUS

100

My friend who really like a rythm game send this file to me, claiming he got it from a rhytm game called Project Sekai. It's really *sus*.

Author: Arif ('saj#6550)

https://drive.google.com/file/d/1zowdSmoGXeZns2J0l8lktRe/view?usp=share_link

Di soal ini kami diberikan sebuah file bernama flag.sus. Kami baru pertama kali mendapatkan file seperti ini dan memutuskan untuk melakukan riset terlebih dahulu terhadap file tersebut. Kemudian kami mendapatkan hint dari link <https://squ1rrel.dev/sekai-sus> yang memberi tahu bahwa file tersebut merupakan file yang digunakan untuk membentuk sebuah rhythm game sesuai dengan deskripsi yang diberikan pada soal.

Dengan menggunakan bantuan level editor dari <https://paletteworks-editor.vercel.app/>, kami berhasil mendapatkan flag dari challenge ini.



Flag = JCTF2023{rEALLysusXDD}

Strange Message

Strange Message

700

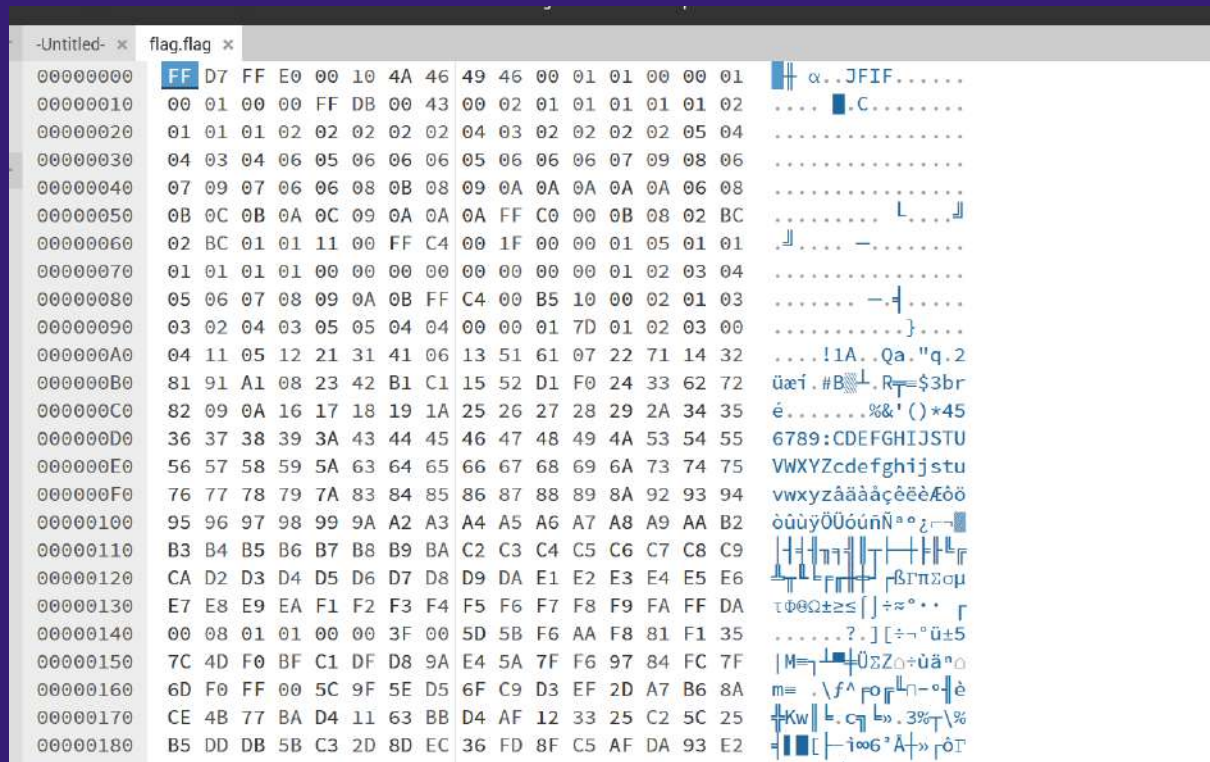
Jota wants to send a message to Krint but in an unusual way. Jota sent her a corrupted file and gave him a hint. Jota says he just sent some pictures that were encrypted with the same key. Can you help Krint decrypt it?

Author: BROP #9678

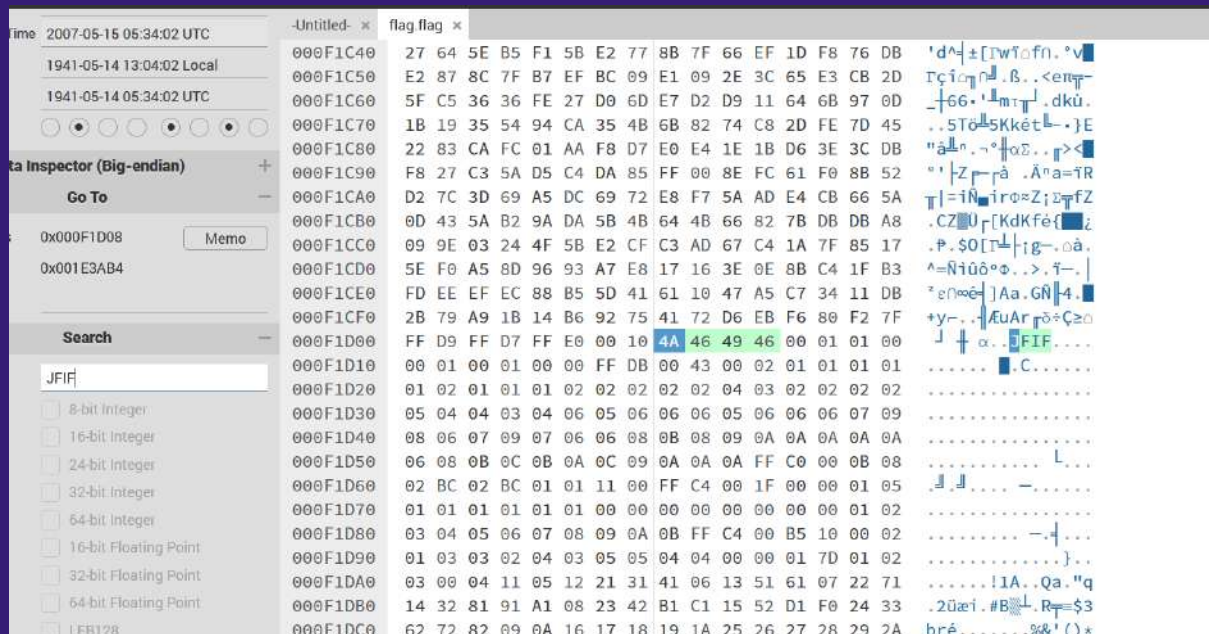
https://drive.google.com/file/d/1nrMPCXXHjhSu81ZWjWyhgotUAR1JH9Hm/view?usp=share_link

Summary

Pada soal ini kita diberikan sebuah file **flag.flag** yang diencrypt oleh pembuat soal. Informasi yang kita dapat dari deskripsi adalah bahwa file tersebut merupakan beberapa gambar yang diencrypt menggunakan key yang sama.



Namun kita hanya diberikan 1 buah file. Kemudian setelah meneliti lebih lanjut, pada pertengahan file, kita menemukan header jpg lain yang ditandai dengan mark **JFIF**.



Maka dari itu, kita memisahkan menjadi kedua file yang berbeda dan bisa dilakukan dengan script berikut:

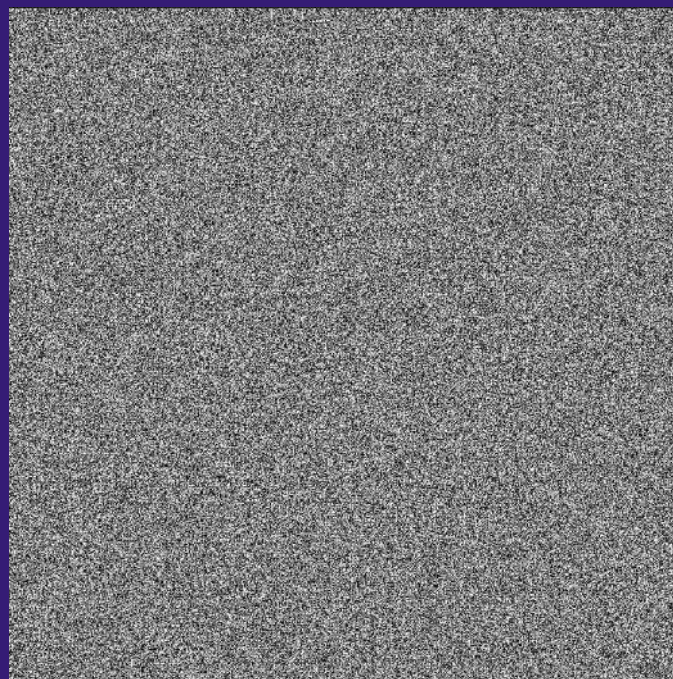
```
file = open('flag.flag', 'rb').read()

x = file.split(b'\xff\xd7\xff\xe0')

file1 = open('file1.jpg', 'wb')
file1.write(b'\xff\xd7\xff\xe0' + x[1])

file2 = open('file2.jpg', 'wb')
file2.write(b'\xff\xd7\xff\xe0' + x[2])
```

Namun kami juga menyadari bahwa kedua buah file memiliki file header yang salah (ff d7 ff e0), padahal file header jpg yang benar adalah ff d8 ff e0. Maka dari itu, kami menggantinya dengan hexedit dan diperoleh 2 file berikut.



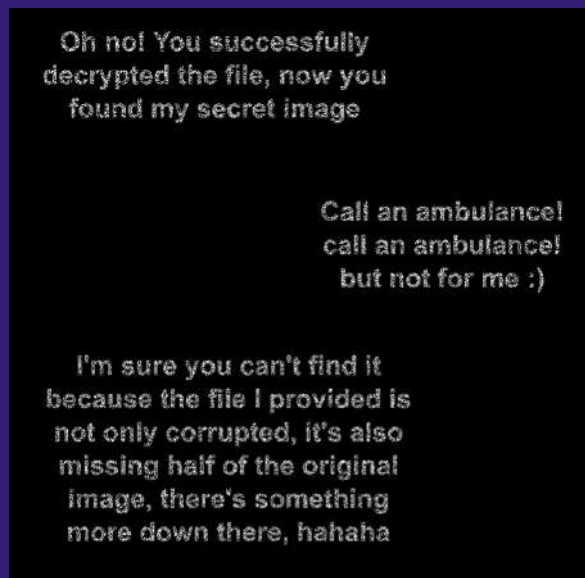
Melanjutkan objektif, kami mencari informasi mengenai enkripsi 2 buah file image menggunakan key yang sama. Pencarian membawa kami kepada [referensi https://crypto.stackexchange.com/questions/88430/how-to-decrypt-two-images-encrypted-using-xor-with-the-same-key](https://crypto.stackexchange.com/questions/88430/how-to-decrypt-two-images-encrypted-using-xor-with-the-same-key)

Kami mengadopsi code dari sumber tersebut dan menggunakannya pada 2 file yang kita dapat.

```
from PIL import Image, ImageChops
im1 = Image.open('file1suer.jpg')
im2 = Image.open('file2suer.jpg')

im3 = ImageChops.add(ImageChops.subtract(im2, im1), ImageChops.subtract(im1,
im2))
im3.show()
im3.save("./im3.png")
```

Dan hasilnya adalah



Lho, ternyata belum nampak flagnya. Pesan yang kami dapat pada gambar hasilnya memberi kita informasi bahwa flag nya masih ada pada file sebelumnya dan tidak terlihat karena setengah bagiannya menghilang. Maka dari itu, kami memutuskan untuk memanipulasi height dengan memainkan hexadecimal dari kedua gambar, kemudian mendecryptnya lagi.

Mark hexadecimal untuk jpg adalah FF C0 dan original dimensionnya adalah 700*700 (0x02 0xbc * 0x02 0xbc), dan kami menemukannya pada hexedit.

Untitled- x	flag.flag x	file1ext.jpg x	file2ext.jpg x	
00000000	FF	D8 FF E0 00 10 4A 46	49 46 00 01 01 00 00 01	⌕ α..JFIF.....
00000010	00	01 00 00 FF DB 00 43	00 02 01 01 01 01 01 02■.C.....
00000020	01	01 01 02 02 02 02 02	04 03 02 02 02 02 05 04
00000030	04	03 04 06 05 06 06 06	05 06 06 06 07 09 08 06
00000040	07	09 07 06 06 08 0B 08	09 0A 0A 0A 0A 0A 06 08
00000050	0B	0C 0B 0A 0C 09 0A 0A	0A FF C0 00 0B 08 02 BCL.⌕
00000060	02 BC	01 01 11 00 FF C4	00 1F 00 00 01 05 01 01	⌕.....-.....
00000070	01	01 01 00 00 00 00 00	00 00 00 00 01 02 03 04
00000080	05	06 07 08 09 0A 0B FF	C4 00 B5 10 00 02 01 03-⌕

-Untitled- x	flag.flag x	file1ext.jpg x	file2ext.jpg x	
00000000	FF	D8 FF E0 00 10 4A 46	49 46 00 01 01 00 00 01	⌕ α..JFIF.....
00000010	00	01 00 00 FF DB 00 43	00 02 01 01 01 01 01 02■.C.....
00000020	01	01 01 02 02 02 02 02	04 03 02 02 02 02 05 04
00000030	04	03 04 06 05 06 06 06	05 06 06 06 07 09 08 06
00000040	07	09 07 06 06 08 0B 08	09 0A 0A 0A 0A 0A 06 08
00000050	0B	0C 0B 0A 0C 09 0A 0A	0A FF C0 00 0B 08 02 BCL.⌕
00000060	02 BC	01 01 11 00 FF C4	00 1F 00 00 01 05 01 01	⌕.....-.....
00000070	01	01 01 00 00 00 00 00	00 00 00 00 01 02 03 04
00000080	05	06 07 08 09 0A 0B FF	C4 00 B5 10 00 02 01 03-⌕
00000090	03	02 04 03 05 05 04 04	00 00 01 7D 01 02 03 00}
000000A0	04	11 05 12 21 31 41 06	13 51 61 07 22 71 14 32!1A..Qa."q.2
000000B0	81	91 A1 08 23 42 B1 C1	15 52 D1 F0 24 33 62 72	üæí. #B⌕L.R=\$3br
000000C0	02	00 04 16 17 18 19 1A	25 26 27 28 29 2A 2B 2C

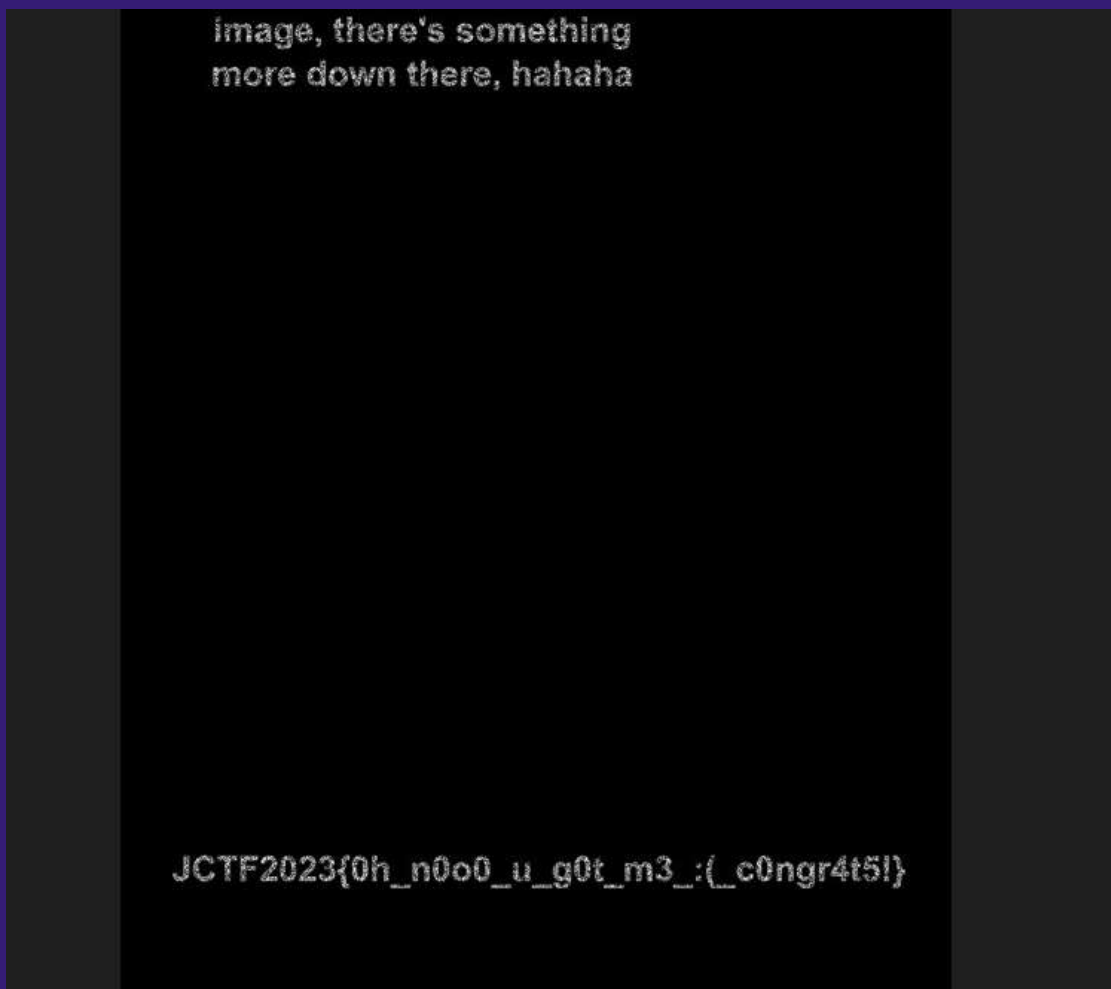
Karena yang mau kita manipulasi adalah heightnya, maka kita akan mengubah value 02 BC yang pertama sebagai pengatur tinggi menjadi 10 FF (4351).

-Untitled- x	flag.flag x	file1ext.jpg x	file2ext.jpg x	
00000000	FF	D8 FF E0 00 10 4A 46	49 46 00 01 01 00 00 01	⌕ α..JFIF.....
00000010	00	01 00 00 FF DB 00 43	00 02 01 01 01 01 01 02■.C.....
00000020	01	01 01 02 02 02 02 02	04 03 02 02 02 02 05 04
00000030	04	03 04 06 05 06 06 06	05 06 06 06 07 09 08 06
00000040	07	09 07 06 06 08 0B 08	09 0A 0A 0A 0A 0A 06 08
00000050	0B	0C 0B 0A 0C 09 0A 0A	0A FF C0 00 0B 08 10 FFL.⌕
00000060	02 BC	01 01 11 00 FF C4	00 1F 00 00 01 05 01 01	⌕.....-.....

Lalu keduanya diexport dan didapatkan 2 file dengan height yang baru.



Lalu kita jalankan lagi script decrypt yang tadi dan dapatkan hasilnya



Flag = JCTF2023{0h_n0o0_u_g0t_m3_:(_c0ngr4t5!}

Feedback

FeedBack

100

Terimakasih telah mengikuti Penyisihan JCTF 2023. Silahkan mengisi feedback berikut ini:

<https://forms.gle/sgd345gnNDVZ1mZi6>

Submit

Summary

Tinggal mengerjakan feedback form dan yes

Flag = JCTF{thanks_for_filling_this_feedback}

OSINT

WhereIsThis

Challenge 68 Solves X

whereIsThis

100

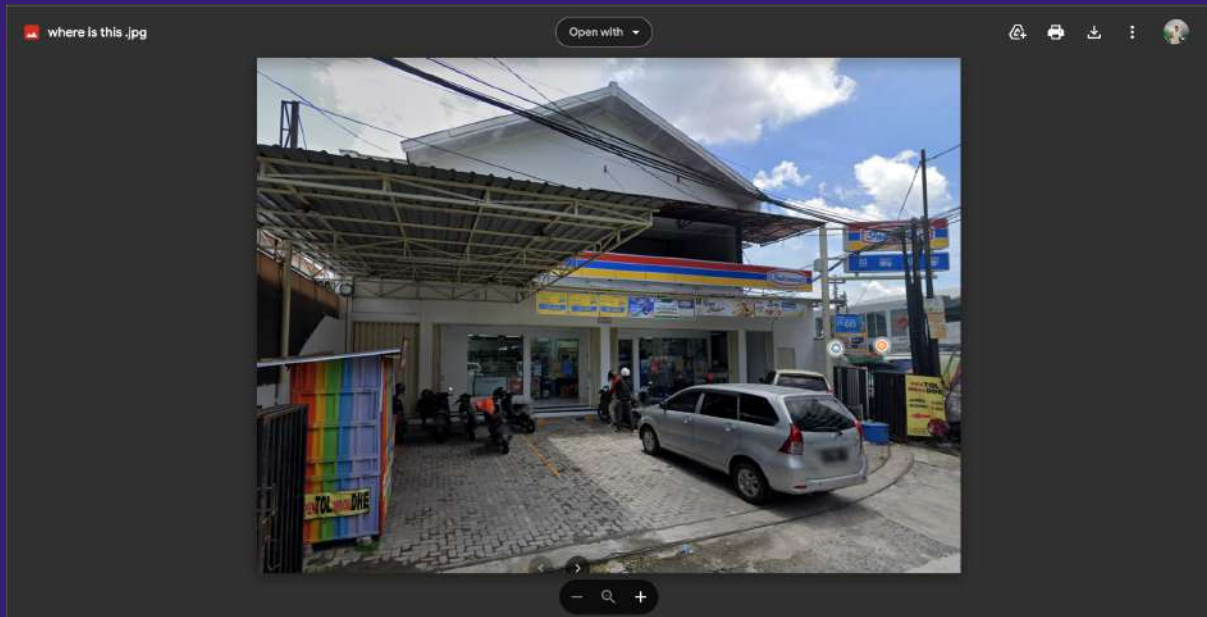
Jota and Krint headed from Tugu Jogja to the north, for some reason Jota and Krint separated, Krint's cellphone ran out of battery and the last photo she sent was a photo of Indomaret version dated January 2022, please help Jota find Indomaret's address to meet Krint. Enter your answer in capital letters using the format JCTF2023{PLUSCODE_KELURAHAN}.

Author: Jears #8964

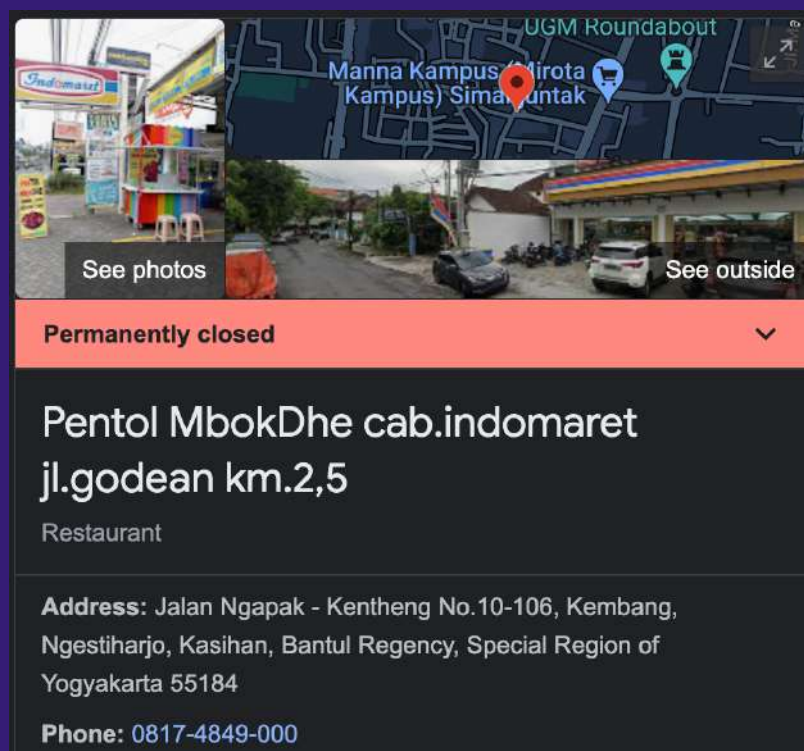
https://drive.google.com/file/d/1labiA8bnjYmeM6HHYkJ9__629EQX7Hs/view?usp=share_link

Pada soal ini, kita diminta untuk mencari lokasi dari gambar yang diberikan dalam bentuk jpg.

HAHA HOHO AWIKWOK GG GEMING

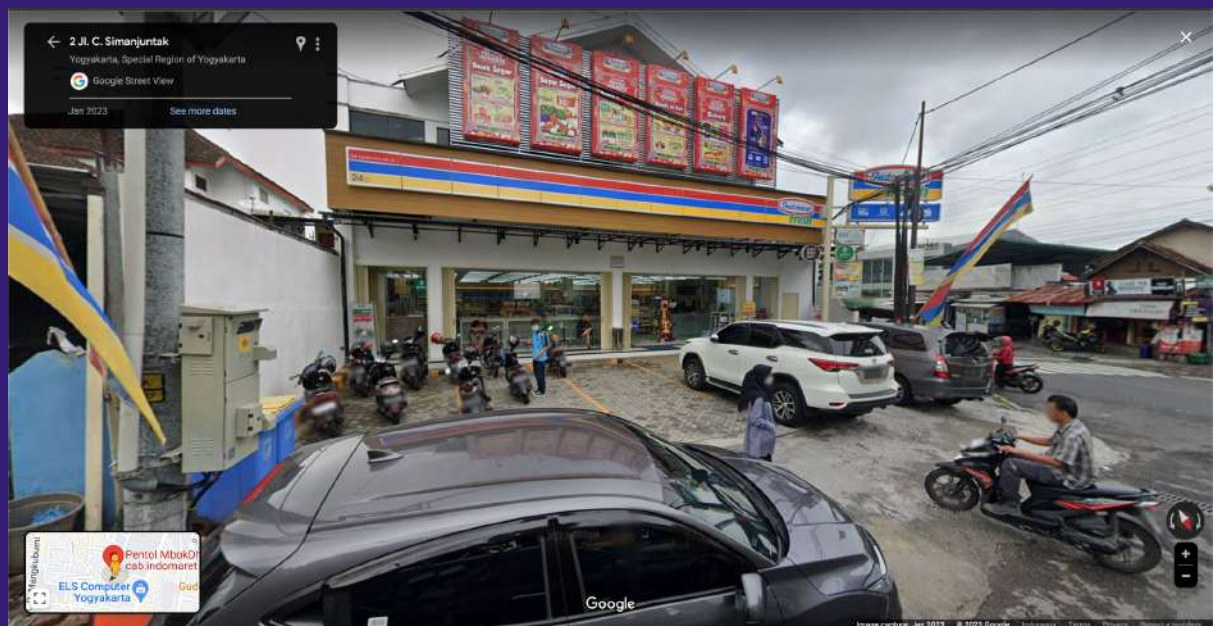
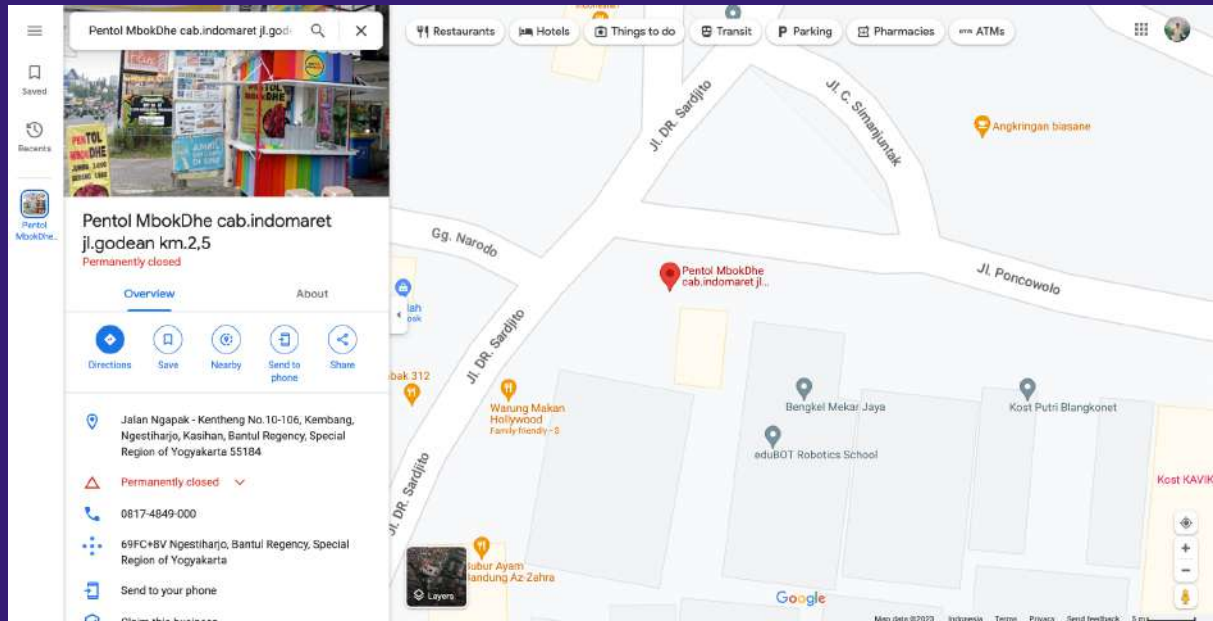


Terlihat bahwa terdapat Indomart dan juga PENTOL MBOK DHE pada foto tersebut. Dikarenakan Joints CTF berasal dari UGM yang berada pada Jogjakarta, kami berinisiatif untuk mencari indomart pentol mbok dhe pada daerah Jogjakarta di google maps dan kami mendapatkan hasil pencarian seperti dibawah ini.



HAHA HOHO AWIKWOK GG GEMING

Dalam analisis lebih lanjut, kami mencoba menelusuri secara real life dengan menggunakan fitur “*Street View*” dari google maps.



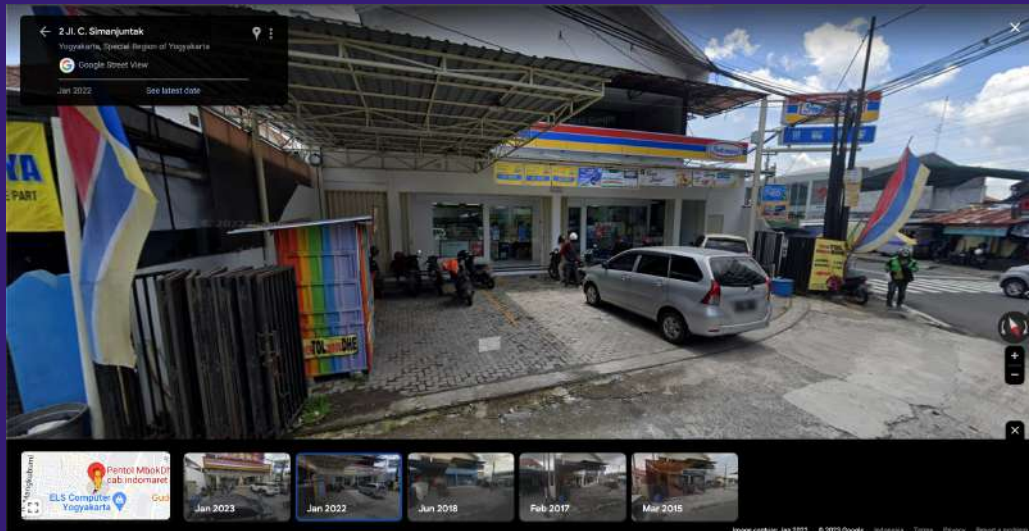
HAHA HOHO AWIKWOK GG GEMING

Kami mendapati bahwa lokasi yang kami dapatkan merupakan lokasi yang tepat dikarenakan adanya kemiripan antara bagian dagangan “hollywood” pada bagian kanan Indomart.

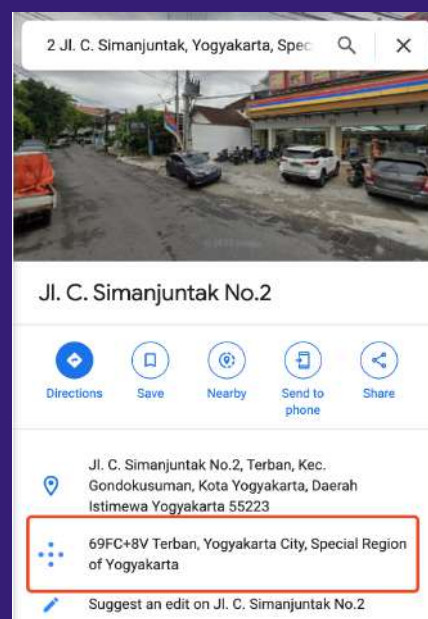


HAHA HOHO AWIKWOK GG GEMING

Bedasarkan hint dari soal, terdapat bulan dan tahun tempat itu di foto. Oleh karena itu, kami mencari tampilan pada waktu yang sesuai yaitu Januari 2022, dan mendapatkan foto yang persis seperti diberikan pada soal.



Dengan memasukkan pluscode dari google dan juga kecamatan yang merupakan Terban dalam bentuk uppercase, kami mendapatkan flag untuk challenge ini.



Flag = JCTF2023{69FC+8V_TERBAN}