

## RVCExIITB CTF Writeup

This is my writeup for the all the challenges i completed in ctf conducted by RVCExIITB



Photo by [Jefferson Santos](#) on [Unsplash](#)

Reverse Engineering

Unscramble

# Unscramble

## 100

Cerggl zhpu abguvat gb fnl, whfg qvir va zna

*Author - Bipin Raj*

 [chall.java](#)

I first started to decode the hint given and I found nothing :(

---

Cerggl zhpu abguvat gb fnl, whfg qvir va zna

---



ROT13 ▾



---

Pretty much nothing to say, just dive in man

---

Then Started to analyse the Java program

This is the python code used to solve this challenge

```
import base64
```

```
def xor_with_key(hex_input, key):
    xored = ""
    for i in range(0, len(hex_input), 2):
```

```

hex_char = int(hex_input[i:i+2], 16)
hex_char ^= key
xored += chr(hex_char)
return xored

def hex_to_ascii(hex_input):
    ascii_str = ""
    for i in range(0, len(hex_input), 2):
        ascii_str += chr(int(hex_input[i:i+2], 16))
    return ascii_str

def caesar_shift(input_str, amount):
    shifted = ""
    for c in input_str:
        if c.isalpha():
            base = 'A' if c.isupper() else 'a'
            shifted += chr((ord(c) - ord(base) - amount) % 26 + ord(base))
        else:
            shifted += c
    return shifted

def main():
    encrypted_flag =
"465a38585060405f685f4465734d6a636d4f45705f4e67384565403d5d5c6e506d5d
3c4d513a513c5862663c58636a736a5c404d504e6639453866676d4f3873"

```

```

# Step 1: XOR with key 9
step1 = xor_with_key(encrypted_flag, 9)
print(f"After XOR: {step1}")

# Step 2: Convert from hexadecimal to ASCII
step2 = hex_to_ascii(step1.encode('utf-8').hex())
print(f"After hex to ASCII: {step2}")

# Step 3: Reverse the Caesar shift by 25
step3 = caesar_shift(step2, 25)
print(f"After Caesar shift: {step3}")

# Step 4: Base64 decode
step4 = base64.b64decode(step3).decode()

```

```

print(f"After Base64 decode 1: {step4}")

# Step 5: Reverse the string
step5 = step4[::-1]
print(f"After reversing: {step5}")

# Step 6: Base64 decode
flag = base64.b64decode(step5).decode()
print("The flag is:", flag)

```

```

if name == "main":
    main()

```

```

After XOR: 0S1QYiIVaVM1zDcjdFLyVGn1L1I4TUgYdT5DX3X5Qko5QjczcUIDYGo0L1ondF1z
After hex to ASCII: 0S1QYiIVaVM1zDcjdFLyVGn1L1I4TUgYdT5DX3X5Qko5QjczcUIDYGo0L1ondF1z
After Caesar shift: PT1RZjJWbWNmaEdkeGMzWHo1MmJ4UVhZeU5EY3Y5R1p5RkdadVJEZHpoM1poeG1a
After Base64 decode 1: ==Qf2VmcfhGdxc3Xz52bxQXYyNDcv9FZyFGZuRDdzt3ZhxmZ
After reversing: ZmxhZ3tzdDRuZGFyZF9vcDNyYXQxb25zX3cxGhfcmlV2fQ==
The flag is: flag{st4ndard_op3rat1ons_w1th_rev}

```

### 1.Reverse XOR Operation:

XOR each pair of hexadecimal digits in the encrypted flag with the key 9.

### 2.Convert Hexadecimal to ASCII:

Convert each pair of hexadecimal digits from the result of the XOR operation to their corresponding ASCII characters.

### 3.Reverse Caesar Shift:

Reverse the Caesar shift by shifting each letter in the result back by 25 positions in the alphabet.

### 4.Base64 Decode:

Base64 decode the string obtained after reversing the Caesar shift.

### 5.Reverse the String:

Reverse the string obtained from the Base64 decode.

### 6.Base64 Decode Again:

Base64 decode the final reversed string to obtain the original flag.

**Nebula**

# Nebula

300

Agents, we've intercepted a binary file from our undercover operative. Dive in, analyze it, and extract the hidden intel. The clock is ticking!

```
7 int64_t var_38;
8 __builtin_strncpy(&var_38, "_UX^BK\nOfA\tKf\\Wz",
9 int64_t var_28 = 0x447756080e69404b;
0 int32_t var_10 = 0x18;
1 char var_11 = 0;
```

← → ⌂ gchq.github.io/CyberChef/#recipe=From\_Hex('Auto')Reverse('Character')&input=MHg0NDc3NTYwODBF

Download CyberChef [Download](#)

Last build: 23 days ago - Version 10 is here! Read about the new feature

Operations	Recipe	Input
rever	From Hex	0x447756080E69404BLU
Reverse	Delimiter Auto	
Remove Diacritics	Reverse	
Remove line numbers	By Character	
From Case Insensitive Regex		
ECDSA Signature Conversion		
Parse IPv4 header		
Disassemble x86		

abc 20 ≡ 1 ↗ 20

Output

```
K@i so b5 VwD
```

main.py

Operations	Save	Run	Output
1 encrypted_string = "_UX^BK\nOfA\tKf\\Wz@i...VwD"			Key: 57, Decrypted Message: flag{r3v_xOr_enCryP71oN}
2 decrypted_messages = []			==== Code Execution Successful ===
3			
4 # Brute-force XOR key			
5 for key in range(256):			
6     decrypted_message = "".join([chr(ord(char) ^ key) for char in encrypted_string])			
7     if decrypted_message.startswith("flag{"):			
8         print(f"Key: {key}, Decrypted Message: {decrypted_message}")			
9         break			
10			

Python code

```
encrypted_string = "_UX^BK\nOfA\\tKf\\WzK@i□□VwD"  
decrypted_messages = []
```

### Brute-force XOR key

```
for key in range(256):  
    decrypted_message = "".join([chr(ord(char) ^ key) for char in encrypted_string])  
    if decrypted_message.startswith("flag{"):  
        print(f"Key: {key}, Decrypted Message: {decrypted_message}")  
        break
```

### Cryptography

Secrets From The Past

# Secrets From The Past 100

Dive into history with a dusty old journal found in an attic, packed with secret messages from a World War II spy. Your task: decode the messages using mechanical means. These cryptic notes could hold long-lost stories that might rewrite history. Can you unravel the wartime secrets concealed within the coded language? Every cracked cipher is a step closer to revealing the untold tales hidden for decades. Get ready to decode and uncover the mysteries of the past!

*Author - Bipin Raj*



configuratio...

configuration.txt

File Edit View

|Enigma M3

UKW B

VI	1A	2B
I	3C	4D
III	5E	7G

bq cr di ej kw mt os px uz gh

cipher - mcwjaqo{s3zc4l\_j3nkp0\_symz34aoyx3?}

cryptii

Students and Teachers, save up 60% on Adobe Creative Cloud.

VIEW Plaintext ▾

rvcectf(g3rm4n\_t3chn0\_unbr34kabl3?)

ENCODE DECODE Enigma machine ▾

MODEL Enigma M3

REFLECTOR UKW B

ROTOR 1 POSITION RING VI - 1 A + - 2 B +

ROTOR 2 POSITION RING I - 3 C + - 4 D +

ROTOR 3 POSITION RING III - 5 E + - 7 G +

PLUGBOARD bq cr di ej kw mt os px uz gh

FOREIGN CHARS |  
Include Ignore

← Decoded 36 chars

VIEW Ciphertext ▾

mcwjaqo{s3zc4l\_j3nkp0\_symz34aoyx3?}

XOR Enigma

# XOR Enigma

150

Your task is to retrieve the flag.

Author - Ananya Bhat

Challenge.txt

```
X1: b3c8d73e3a9b23df7cc1253277a4878ef65bcfe9735f29d84424
X2^X1: fb3514ac2e94885e9d5ec915821650572d5e0b842e9630f32b1b
X2^X3: d2656867798e8584ec34ab2d4562b1a9c82b8fcf1feeeddf70e2
FLAG^X1^X3^X2: 07c1de3e3867c32fe29cbd6957a2695f0e021f4b58c2b03446bb
```

To solve this XOR Enigma challenge, we need to use the given XORED values to retrieve the flag.  $\text{FLAG} = (\text{FLAG} \oplus \text{X1} \oplus \text{X3} \oplus \text{X2}) \oplus \text{X1} \oplus \text{X3} \oplus \text{X2}$

Python code to solve the challenge

```
from binascii import unhexlify
```

**Given hexadecimal strings**

```
X1_hex = "b3c8d73e3a9b23df7cc1253277a4878ef65bcfe9735f29d84424"
X2_X1_hex = "fb3514ac2e94885e9d5ec915821650572d5e0b842e9630f32b1b"
X2_X3_hex = "d2656867798e8584ec34ab2d4562b1a9c82b8fcf1feeeddf70e2"
FLAG_X1_X3_X2_hex =
"07c1de3e3867c32fe29cbd6957a2695f0e021f4b58c2b03446bb"
```

**Convert hex to bytes**

```
X1 = unhexlify(X1_hex)
X2_X1 = unhexlify(X2_X1_hex)
X2_X3 = unhexlify(X2_X3_hex)
FLAG_X1_X3_X2 = unhexlify(FLAG_X1_X3_X2_hex)
```

**XOR function**

```
def xor_bytes(a, b):
    return bytes(x ^ y for x, y in zip(a, b))
```

## Find X2

```
X2 = xor_bytes(X2_X1, X1)
```

## Find X3

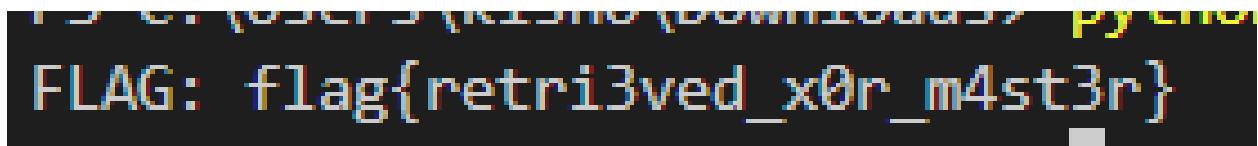
```
X3 = xor_bytes(X2_X3, X2)
```

## Find FLAG

```
FLAG = xor_bytes(FLAG_X1_X3_X2, xor_bytes(X1, xor_bytes(X3, X2)))
```

## Print the FLAG as text

```
print("FLAG:", FLAG.decode())
```



```
FLAG: flag{retri3ved_x0r_m4st3r}
```

Builder Bob and Alice

# Builder Bob and Alice

## 200

Bob sends a message to Alice and Charlotte. Help Alice and Charlotte decrypt the message.

*Author - Ananya Bhat*



message.txt

Modulus N1:

42912177063137856790134396660159463800520001541008404987700507470624

21449988359200686359240923271551547777242609206985640742460474280587  
02591438336354875385912113367812170140583119952718402254809563407665  
54675704097608902403126500806982757366189523318775082296632391374524  
3562262084682435720233192587715830559

Public exponent 1 :

38603263397610649045276278024810304676508067100298889205533064151956  
42358529227628226424022795789188386362467526529100940481427228918962  
47145254930177193887644017516737007121616205743252470102524562046452  
02284428559250285013655711099889127934696611167432770514911603448732  
7385428052869529026582167188809884767

Ciphertext1:

87502995845613296640748517793461033238581559539831264070261405010457  
50904507397467842160348342428403095393600366516338788248447796879276  
10789885956103024554033886772949584330693028965337829141771560607189  
58383452389999702278314396059598387560745347823772156262268912853464  
875822933280715397288456397500467082

Modulus N2:

42912177063137856790134396660159463800520001541008404987700507470624  
21449988359200686359240923271551547777242609206985640742460474280587  
02591438336354875385912113367812170140583119952718402254809563407665  
54675704097608902403126500806982757366189523318775082296632391374524  
3562262084682435720233192587715830559

Ciphertext2:

13606671489326854202680451938949469697733836249223291102563052866524  
93670784934496330657647036916984283419345845956998197935883129461033  
71760765032815931350400960037961574476205220355485393336049341594068  
05022210982063656428403476388602689487414083214496871695111186845383  
618146998255478322713071884032213425

Public exponent 2 :

16375039649593585292390496620481532437752973603469434507564693050788  
75716076113620696837184604694440539081435510231185806051766143264771  
89584938691146244409712347416452049032774643069943984015216081063830  
86133632457067299706359236870718333482835235923710147875841875049939  
4554746092209859925767816955766723283

To decrypt the message sent to Alice and Charlotte, we need to use the Chinese Remainder Theorem (CRT) attack. This attack is applicable because the same modulus N is used with different exponents e1 and e2.

This is my Python code to solve.

```
from sympy import mod_inverse
```

### Given values

N =  
42912177063137856790134396660159463800520001541008404987700507470624  
21449988359200686359240923271551547777242609206985640742460474280587  
02591438336354875385912113367812170140583119952718402254809563407665  
54675704097608902403126500806982757366189523318775082296632391374524  
3562262084682435720233192587715830559

e1 =  
38603263397610649045276278024810304676508067100298889205533064151956  
42358529227628226424022795789188386362467526529100940481427228918962  
47145254930177193887644017516737007121616205743252470102524562046452  
02284428559250285013655711099889127934696611167432770514911603448732  
7385428052869529026582167188809884767

e2 =  
16375039649593585292390496620481532437752973603469434507564693050788  
75716076113620696837184604694440539081435510231185806051766143264771  
89584938691146244409712347416452049032774643069943984015216081063830  
86133632457067299706359236870718333482835235923710147875841875049939  
4554746092209859925767816955766723283

C1 =  
87502995845613296640748517793461033238581559539831264070261405010457  
50904507397467842160348342428403095393600366516338788248447796879276  
10789885956103024554033886772949584330693028965337829141771560607189  
5838345238999702278314396059598387560745347823772156262268912853464  
875822933280715397288456397500467082

C2 =  
13606671489326854202680451938949469697733836249223291102563052866524  
93670784934496330657647036916984283419345845956998197935883129461033  
7176076503281593135040096003796157447620522035548539336049341594068  
05022210982063656428403476388602689487414083214496871695111186845383  
6181469982554783322713071884032213425

**Step 1: Verify that  $\gcd(e1, e2) = 1$**

```
def extended_gcd(a, b):
if a == 0:
return b, 0, 1
gcd, x1, y1 = extended_gcd(b % a, a)
x = y1 - (b // a) * x1
y = x1
return gcd, x, y
```

```
gcd, a, b = extended_gcd(e1, e2)
assert gcd == 1, "gcd(e1, e2) is not 1"
```

**Step 2: Compute the combined ciphertext C using the formula  $C = (C1^a * C2^b) \% N$**

```
def mod_exp(base, exp, mod):
if exp < 0:
base = mod_inverse(base, mod)
exp = -exp
return pow(base, exp, mod)
```

```
M = (mod_exp(C1, a, N) * mod_exp(C2, b, N)) \% N
```

**Step 3: Print the decrypted message M**

```
print(M)
```

**Convert the decrypted message to a readable string if it's in byte format**

```
print(bytes.fromhex(hex(M)[2:]).decode('utf-8'))
```

```
17333139023728957124685418399751573378854259303384745173582659331953788829329996627518269717062951364050685063660253997347120509
flag{MoDU10_Att4ccKk_c0mpLete_Y0u_m4de_b0B_H4ppy!#%$}
```

**OSINT**

# Vacation

ns

## 150

On June 5th 2024, it was finally time for Ajay's well deserved vacation. He had been working his socks off for the past couple months and the city of Joy had deprived him of any sort of joy. All set, he packed his bags and left for the airport.

Ajay was going to fly to the airport which had won the Skytrax World Airport Award last year. After all the check-in procedures, he boarded his flight. He trusted the airline highly even though one of the aircraft of the airline had recently dropped 6000 feet! However, that wasn't going to affect his vacation and he was gonna enjoy this one.

Your task is to find out the details of the flight and submit your flag in this format:

```
flag{flightName_aircraftRegistrationNumber_greatCircleDistance(in  
KM)_actualFlightTime(HH:MM)_aircraftTypeAndName}
```

example flag:-

example flag:-`flag{ABC123_EF-IGH_6523_09:30_Airbus E532-694GT}`

### Steganography

Lawliet's Successor Beyond

# Lawliet's Successor

## Beyond

### 100

Beyond was Lawliet's successor and he was told one thing before L's death -

"Remember, in this world, nothing remains concealed forever; as the day dawns, mysteries unravel, and shadows dissipate, leaving secrets exposed to those who dare to seek."

Fun fact - L's unique seated posture optimizes blood flow, directing it solely to the brain, thereby enhancing cognitive faculties and intelligence.

*Author - Bipin Raj*

 [chall.jpg](#)

`jsteg` is a package for hiding data inside jpeg files, a technique known as [steganography](#). This is accomplished by copying each bit of the data into the least-significant bits of the image. The amount of data that can be hidden depends on the filesize of the jpeg; it takes about 10-14 bytes of jpeg to store each byte of the hidden data.

```
└$ ls
chall.jpg  jsteg-linux-amd64  Downloads  LawlietSuccessorBeyond  steg
└─(kali㉿kali)-[~/Downloads/rvctf Writeup/steg/LawlietSuccessorBeyond]
$ ./jsteg-linux-amd64 reveal chall.jpg
zsh: permission denied: ./jsteg-linux-amd64

└─(kali㉿kali)-[~/Downloads/rvctf Writeup/steg/LawlietSuccessorBeyond]
$ chmod +x jsteg-linux-amd64

└─(kali㉿kali)-[~/Downloads/rvctf Writeup/steg/LawlietSuccessorBeyond]
$ ./jsteg-linux-amd64 reveal chall.jpg
flag{th3_be5t_det3ct1ve_0n_pl4neT_e4rTh}
```

## Web

Robot Uprising

# Robot Uprising

## 150

The robot uprising is here. Unravel the secret.

<https://robot-uprising.rvcechalls.xyz/>

robot-uprising.rvcechalls.xyz

Welcome to the Robot World

In a world where robots and humans coexist, there are many secrets hidden in plain sight.

The robots have their own way of communicating, and sometimes, you just need to know where to look and how to ask.

Explore the world of robots and uncover the secrets they hold. Remember, not everything is as it seems.

Pay attention to the details and use your skills to discover hidden information.

To start your adventure, try looking for clues in places where robots might leave messages.

© 2024 Robot World. All rights reserved.

→ C [robot-uprising.rvcechalls.xyz/robots.txt](https://robot-uprising.rvcechalls.xyz/robots.txt)

User-agent: \*

YOU HAVE ENTERED THE ROBOTS BASE!  
BUT NOT THROUGH THE RIGHT DOOR,  
NOT WITH THE RIGHT ID!  
MAYBE WALL-E HAS THE SECRET!

Request			Response		
Pretty	Raw	Hex	Pretty	Raw	Hex
<pre> 1 GET /robots.txt HTTP/1.1 2 Host: robot-uprising.rvcechalls.xyz 3 user-agent: WALL-E 4 Cache-Control: max-age=0 5 Sec-Ch-Ua: "Chromium";v="125", "Not.A/Brand";v="24" 6 Sec-Ch-Ua-Mobile: ? 7 Sec-Ch-Ua-Platform: "Windows" 8 Upgrade-Insecure-Requests: 1 9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.6422.60 Safari/537.36 10 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7 11 Sec-Fetch-Site: none 12 Sec-Fetch-Mode: navigate 13 Sec-Fetch-User: ?1 14 Sec-Fetch-Dest: document 15 Accept-Encoding: gzip, deflate, br 16 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8 17 Priority: u=0, i 18 Connection: keep-alive </pre>			<pre> 1 HTTP/1.1 200 OK 2 Server: nginx/1.24.0 (Ubuntu) 3 Date: Sat, 08 Jun 2024 16:05:26 GMT 4 Content-Type: text/plain; charset=utf-8 5 Content-Length: 134 6 Connection: keep-alive 7 8 9 User-agent: * 10 11 THE ROBOT UPRISE IS ON THE WAY! 12 HERE'S THE SECRET KEY: flag(wallE_might_b3_4n_Ally) 13 </pre>		

## Confidential leak

# Confidential leak

150

In shadows deep, where secrets hide, Credentials leaked, our worlds collide. Paths we trace, in search of light, To find what's lost, and set things right.

<https://confidential-leak.rvcechalls.xyz/>

Author - Bipin Raj

```

/scripts
var express = require('express');
var app = express();
var port = process.env.PORT || 9898;
var crypto = require('crypto');
var bodyParser = require('body-parser')
var salt = 'somestring';
var iteration = /// some number here;
var keylength = // some number here;

```

```

app.post('/login', function (req, res) {
var username = req.body.username;
var password = req.body.password;
if (username !== 'joemama') {

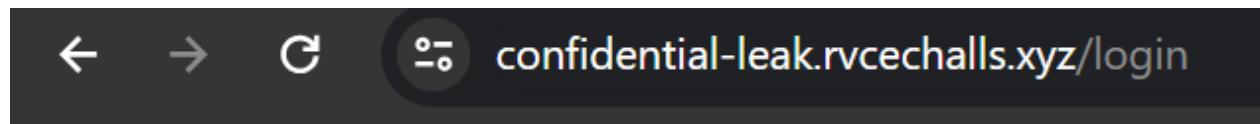
```

```
res.send('Username is wrong');
return;
}
if (crypto.pbkdf2Sync(password, salt, iteration, keylength).toString() =
hashOfPassword) {
if (password =
'plnlrtfpijpuhqylxbgqiiyipieyxvfsavzgbfcusqkozwpngsyejqImjsytrmd') {
// some logic here and return something
} else {
// return flag here
}
} else {
res.send('Password is wrong');
}
});
```

From this got username as joemama

<https://mathiasbynens.be/notes/pbkdf2-hmac>

From this password eBkXQTfuBqp'cTcar&g\*



Flag: flag{d1d\_i\_ju5t\_w1tness\_a\_h4sh\_c0ll1sion??}

Forensics

Labyrinth

# Labyrinth

## 200

Navigate through the labyrinth, and retrieve the flag. Keep count of your steps like 4!5!4! or you may encounter a dead end.

Author - Ananya Bhat

[!\[\]\(3b451835b5cf44dc087a11f8c88642da\_img.jpg\) chall.iso](#)

DVD Drive (I:) CDROM > dir4 > dir5 > dir4 >				
A	E	W	Sort	View
Name	Date modified	Type	Size	
dir1	21-05-2024 09:12 ...	File folder		
dir2	21-05-2024 09:12 ...	File folder		
dir3	21-05-2024 09:13 ...	File folder		
dir4	21-05-2024 09:12 ...	File folder		
dir5	21-05-2024 09:12 ...	File folder		
john	22-05-2024 10:24 ...	File folder		
dirr3	22-05-2024 10:19 ...	File	1 KB	

After Opening dirr3

Ao(mgHXnjO1LkM\EBT+?"5HBPD?kA8-  
'q@oRAEBgm)MF>RcEFDCMAo(mgFDk`3@ps=f<+oiZ@:FM&Bl8\$+I/

After decoding the encrypted text from Base85 we got flag.

The screenshot shows a hex editor interface. In the 'Input' pane, there is a large amount of Base64 encoded data. In the 'Output' pane, the decoded text is displayed: `flag{dir3ct0ries_w1thin_direcToRiEs_t4keth1sflagtoescapeThelabyrinth}`.

**TheChinesePhilosopher**

# TheChinesePhilosopher

## 150

Sun Tzu, a philosopher with an insatiable curiosity about the cosmos, defies convention by becoming an astronaut. His journey into space is fueled by a quest for the universe's ultimate truths. Amidst the silent expanse, he finds awe-inspiring wonders, but ultimately realizes that the true answers lie not in the stars, but within the depths of his own contemplation.

*Author - Bipin Raj*

[space.gif](#)

After extracting all frames in the gif in one of the frames we got some part of the flag :)



It contains flag{m4yb3Th3\_un1v3r53}

We need remaining part of the flag after doing exif in the given gif we got an link to the website where chinese text is present after decoding we got remaining part of the flag

```
[Kati@Kati] ~ ~/Downloads/IVCTF_WiTeUp/forensic/TheChinesePhilosopher]$ exiftool space.gif
ExifTool Version Number : 12.57
File Name : space.gif
Directory : .
File Size : 1204 kB
File Modification Date/Time : 2024:06:08 12:56:53-04:00
File Access Date/Time : 2024:06:08 14:27:52-04:00
File Inode Change Date/Time : 2024:06:08 13:00:04-04:00
File Permissions : -rwxrwx---
File Type : GIF
File Type Extension : gif
MIME Type : image/gif
GIF Version : 89a
Image Width : 320
Image Height : 180
Has Color Map : Yes
Color Resolution Depth : 8
Bits Per Pixel : 8
Background Color : 0
Animation Iterations : Infinite
Comment : GIF edited with https://ezgif.com/add-text
XMP Toolkit : Image::ExifTool 12.47
Creator : socrates
Description : https://katb.in/nenujuhoxay
Title : ThePhilosopher
Frame Count : 23
Duration : 2.30 s
Image Size : 320×180
Megapixels : 0.058
```



katb.in/nenujuhoxay

<Katbin/>

粂簽籼穀米穲簾籽穀簾穀簾穀機穲料簽籽簾管粄粄

The screenshot shows two adjacent browser tabs. The left tab is titled 'dCODE' and contains a search bar with 'e.g. type 'caesar'', a 'BROWSE THE FULL dCODE TOOLS' LIST' button, and a results section showing the ciphertext 'w4s\_ju5t\_a\_5imul4t10n}'. Below the results is a small image of a person's head and a 'SAVE A LIFE INDIA' button. The right tab is titled 'ROT8000 DECODER' and shows the same ciphertext. It includes a 'ROT8000 CIPHERTEXT' input field with a question mark icon, a 'DECRYPT' button, and a note at the bottom: '粂簽籼穀米穲簾籽穀簾穀簾穀機穲料簽籽簾管粄粄'.

After Combining Flag i got flag{m4yb3\_Th3\_un1v3r53\_w4s\_ju5t\_a\_5imul4t10n}

Miscellaneous

Etherreal

150

Deposit, Deploy and Explore

<https://smart-contract-1.rvcechalls.xyz/>

Author - Karan

Locate the flag

Connect Wallet

Wallet Address:

Deploy Contract

smart-contract-1.rvcechalls.xyz/static/js/bundle.js

```
o generate a random string
|rateRandomString(length) {
ctors = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
= "";
= 0; i < length; i++) {
    characters.charAt(Math.floor(Math.random() * characters.length));

lt;

o deploy smart contract
n deployContract() {
window.ethereum !== "undefined") {
vider = new ethers__WEBPACK_IMPORTED_MODULE_4__.Web3Provider(window.ethereum);
vider.send("eth_requestAccounts", []);
ner = provider.getSigner();

te a random flag
g = generateRandomString(10) + "flag{4lway5_v13w_data_1n_all_f0rmats_98098}";
flag = "karan";
```

PWN

The Baker 1

# The Baker 1

100

Anna, the baker, needs help baking a cake. You know what to do. Help her out.

nc rvcechalls.xyz 24695

# The Baker 2

# The Baker 2

150

Anna has implemented a new system to take orders and print it out. Check it out.

nc rvcechalls.xyz 24692

Author - Ananya Bhat



Recipe

Reverse

By Byte

Input

1ls{galf}Ssilb3c

REC 16 = 1

Output

c3blisS}flag{s11

Got flag as flag{s11c3blisS}