HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÁO CÁO BÀI THỰC HÀNH

Kiểm thử xâm nhập

Lỗ hổng Buffer Overflow

Giảng viên: Đinh Trường Duy

Nhóm lớp: 02

Sinh viên: Hoàng Trung Kiên

Mã sinh viên: B20DCAT098

Hà Nội – 2024

Mục lục

1. **Mục đích.**

Giúp sinh viên hiểu rõ nguyên nhân, cơ chế cũng như cách thức khắc phục, xử lý lỗ hổng tràn bộ đệm.

2. **Yêu cầu đối với sinh viên.**

Có kiến thức về ngôn ngữ lập trình C.

Tìm hiểu về Stack, Function, Buffer Overflow và các cơ chế bảo vệ của hệ thống.

3. **Nội dung thực hành.**

Khởi động lab

Chạy lệnh: *labtainer -r bufoverflow* trong terminal của Labtainer

```
student@ubuntu:~/labtainer/labtainer-student$ labtainer bufoverflow
non-network local connections being added to access control list
Started 1 containers, 1 completed initialization. Done.



Buffer Overflow Lab -- Read this first

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


Press <enter> to start the lab
```

- Tắt các cơ chế bảo vệ có liên quan

sudo sysctl -w kernel.randomize_va_space=0

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

biên dịch call_shellcode.c

gcc -m32 -z execstack -o call_shellcode call_shellcode.c

```
ubuntu@bufoverflow:~$ gcc -m32 -z execstack -o call_shellcode call_shellcode.c
ubuntu@bufoverflow:~$
```

Biên dịch chương trình stack.c và chuyển quyền thành set-root-uid

sudo su gcc -m32 -o stack -z execstack -fno-stack-protector stack.c

chmod

4755

stack

exit

```
ubuntu@bufoverflow:~$ sudo su
root@bufoverflow:/home/ubuntu# gcc -m32 -o stack -z execstack -fno-stack-protector stack.c
root@bufoverflow:/home/ubuntu# chmod 4755 stack
root@bufoverflow:/home/ubuntu# exit
exit
ubuntu@bufoverflow:~$
```

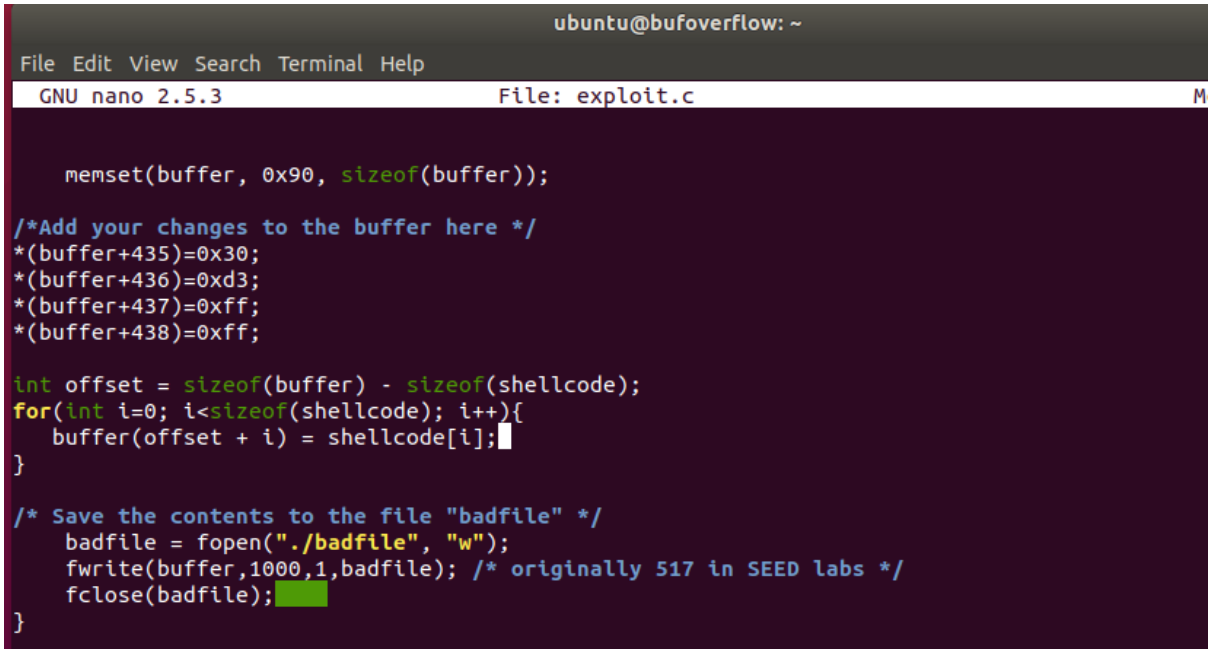
Nhiệm vụ 1: Khai thác lỗ hổng bảo mật.

Tìm offset của buffer

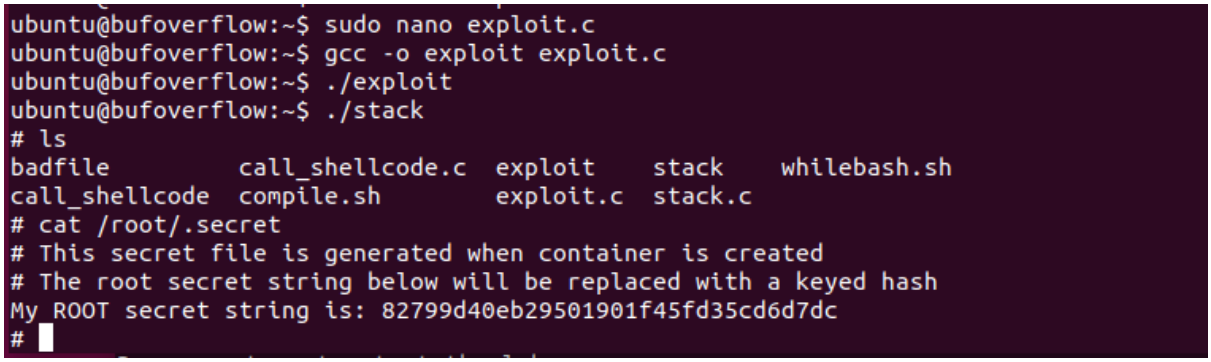
```
(gdb) disassemble bof
Dump of assembler code for function bof:
   0x080484bb <+0>:     push   %ebp
   0x080484bc <+1>:     mov    %esp,%ebp
   0x080484be <+3>:     sub    $0x1b8,%esp
   0x080484c4 <+9>:     sub    $0x8,%esp
   0x080484c7 <+12>:    pushl  0x8(%ebp)
   0x080484ca <+15>:    lea    -0x1af(%ebp),%eax
   0x080484d0 <+21>:    push   %eax
   0x080484d1 <+22>:    call   0x8048370 <strcpy@plt>
   0x080484d6 <+27>:    add    $0x10,%esp
   0x080484d9 <+30>:    mov    $0x1,%eax
   0x080484de <+35>:    leave
   0x080484df <+36>:    ret
End of assembler dump.
```

```
ubuntu@bufoverflow: ~
File  Edit  View  Search  Terminal  Help
Breakpoint 1 at 0x80484ca
(gdb) r
Starting program: /home/ubuntu/stack

Breakpoint 1, 0x080484ca in bof ()
(gdb) i r
eax            0xffffd2f4       -11532
ecx            0x804b0a0        134525088
edx            0x3e8    1000
ebx            0x0      0
esp            0xffffd114       0xffffd114
ebp            0xffffd2d8       0xffffd2d8
esi            0xf7fc8000       -134447104
edi            0xf7fc8000       -134447104
eip            0x80484ca        0x80484ca <bof+15>
eflags         0x296    [ PF AF SF IF ]
cs             0x23     35
ss             0x2b     43
ds             0x2b     43
es             0x2b     43
fs             0x0      0
gs             0x63     99
(gdb) i r $ebp
ebp            0xffffd2d8       0xffffd2d8
(gdb)
```



Biên dịch và chạy



Ta đã có quyền root và đọc được file secret

Nhiệm vụ 2: Ngẫu nhiên hóa địa chỉ.

Bật ngẫu nhiên hóa địa chỉ cho file

sudo /sbin/sysctl -w kernel.randomize_va_space=2

gcc -m32 -o stack -z execstack stack.c

```
ubuntu@bufoverflow:~$ sudo /sbin/sysctl -w kernel.randomize_va_space=2
kernel.randomize_va_space = 2
ubuntu@bufoverflow:~$ ls
badfile         call_shellcode.c  exploit    stack    whilebash.sh
call_shellcode  compile.sh        exploit.c  stack.c
ubuntu@bufoverflow:~$ ./whilebash.sh
./whilebash.sh: line 24:   444 Segmentation fault      (core dumped) ./stack
./whilebash.sh: line 24:   445 Segmentation fault      (core dumped) ./stack
./whilebash.sh: line 24:   446 Segmentation fault      (core dumped) ./stack
./whilebash.sh: line 24:   447 Segmentation fault      (core dumped) ./stack
./whilebash.sh: line 24:   448 Segmentation fault      (core dumped) ./stack
./whilebash.sh: line 24:   449 Segmentation fault      (core dumped) ./stack
./whilebash.sh: line 24:   450 Segmentation fault      (core dumped) ./stack
./whilebash.sh: line 24:   451 Segmentation fault      (core dumped) ./stack
./whilebash.sh: line 24:   452 Segmentation fault      (core dumped) ./stack
./whilebash.sh: line 24:   453 Segmentation fault      (core dumped) ./stack
./whilebash.sh: line 24:   454 Segmentation fault      (core dumped) ./stack
./whilebash.sh: line 24:   455 Segmentation fault      (core dumped) ./stack
./whilebash.sh: line 24:   456 Segmentation fault      (core dumped) ./stack
./whilebash.sh: line 24:   457 Segmentation fault      (core dumped) ./stack
./whilebash.sh: line 24:   458 Segmentation fault      (core dumped) ./stack
./whilebash.sh: line 24:   459 Segmentation fault      (core dumped) ./stack
./whilebash.sh: line 24:   460 Segmentation fault      (core dumped) ./stack
./whilebash.sh: line 24:   461 Segmentation fault      (core dumped) ./stack
```

Trả lời câu hỏi: Ta có thể nhận được shell không? Nếu không, vấn đề là gì? Làm thế nào để ngẫu nhiên hóa địa chỉ làm cho các cuộc tấn công trở nên khó khăn?
Ta không nhận được shell vì địa chỉ chúng ta set cho return trong hàm bof() của stack mặc định, nên khi nó ngẫu nhiên địa chỉ thì địa chỉ trả về của chúng ta không đúng trong stack nên nó không thành công. Điều đó làm các cuộc tấn công khó khăn thực hiện thành công thì không thể biết được giá trị nào hợp lệ

Nhiệm vụ 3: Stack Guard.

sudo sysctl -w kernel.randomize_va_space=0

gcc -m32 -o stack -z execstack stack.c

```
ubuntu@bufoverflow:~$ sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
ubuntu@bufoverflow:~$ gcc -m32 -o stack -z execstack stack.c
ubuntu@bufoverflow:~$ ./stack
*** stack smashing detected ***: ./stack terminated
/usr/sbin/exec_wrap.sh: line 16:   536 Aborted                 (core dumped) ./stack
ubuntu@bufoverflow:~$
```

Nhiệm vụ 4: Ngăn xếp không thực thi.

gcc -m32 -o stack -fno-stack-protector -z noexecstack stack.c

```
ubuntu@bufoverflow:~$ gcc -m32 -o stack -fno-stack-protector -z noexecstack stack.c
ubuntu@bufoverflow:~$ ./stack
Segmentation fault (core dumped)
ubuntu@bufoverflow:~$
```

Tùy chọn -z noexecstack khi biên dịch chương trình sẽ ngăn chặn khả năng thực thi trên stack. Điều này thường được sử dụng như một biện pháp bảo mật để ngăn chặn việc thực thi mã được đặt vào stack, giảm thiểu khả năng bị tấn công bởi các cuộc tấn

công như tràn bộ đệm. Sau khi biên dịch chương trình với noexecstack, việc tìm và thực thi shellcode từ stack sẽ không còn khả thi.

4. **Checkwork**.

```
student@ubuntu:~/labtainer/labtainer-student$ checkwork
Results stored in directory: /home/student/labtainer_xfer/bufoverflow
Labname bufoverflow

Student              |  gain_root_priv |       while_run |   stack_protect |
==================== | =============== | =============== | =============== |
B20DCAT098           |               Y |               Y |               Y |
What is automatically assessed for this lab:

        gain_root_priv: Did the student get a root shell & display the /root/.secret file?
        while_run: Did the student run the whilebash.sh with aslr on?
        stack_protect: Experimented with enabling stack guard?
```