

# Lab2 report

11912039 郑鑫颖

## a. What happens when you compile without "-z execstack"?

Since "-z execstack" turns off the NX protection to make the stack executable. So when I turn on it, it all reports a segement fault as follows.

```
Reading symbols from BOF...done.
gdb-peda$ run
Starting program: /root/Desktop/Lab2-BufferOverflows/BOF
Buffer overflow vulnerability starting up...

Program received signal SIGSEGV, Segmentation fault.

[-----registers-----]
EAX: 0x1
EBX: 0xb7fb6000 --> 0x1a5da8
ECX: 0xbffff220 --> 0x80cd0bb0
EDX: 0xbffff1e8 --> 0x80cd0bb0
ESI: 0x0
EDI: 0x0
EBP: 0x80cd0bb0
ESP: 0xbffff1f0 --> 0xbffff200 --> 0x8
EIP: 0xbffff20c --> 0x6850c031
EFLAGS: 0x10282 (carry parity adjust zero SIGN trap INTERRUPT direction overflow)

[-----code-----]
=> 0xbffff20c: xor    eax,eax
0xbffff20e: push  eax
0xbffff20f: push  0x68732f2f
0xbffff214: push  0x6e69622f

[-----stack-----]
0000| 0xbffff1f0 --> 0xbffff200 --> 0x8
0004| 0xbffff1f4 --> 0x1
0008| 0xbffff1f8 --> 0x200
0012| 0xbffff1fc --> 0x804a008 --> 0xfbad2488
0016| 0xbffff200 --> 0x8
0020| 0xbffff204 --> 0x0
0024| 0xbffff208 --> 0xbffff2f4 --> 0x0
0028| 0xbffff20c --> 0x6850c031

[-----]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0xbffff20c in ?? ()
```

## b. What happens if you enable ASLR? Does the return address change?

Definitely yes. When I enable the ASLR, the return address changes each time. So if I don't change my code, I would get a segement fault.

```
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# echo 2 > /proc/sys/kernel/randomize_va_space
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# ./BOF
Buffer overflow vulnerability starting up...
address: bff8e29c.
Segmentation fault
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# ./BOF
Buffer overflow vulnerability starting up...
address: bfc8a7ec.
Segmentation fault
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# ./BOF
Buffer overflow vulnerability starting up...
address: bfd00d0c.
Segmentation fault
```

c. Does the address of the buffer[] in memory change when you run BOF using GDB, /home/root/Desktop/Lab2-BufferOverflows/BOF, and ./BOF?

They are all different.

GDB:

```
gdb-peda$ run
Starting program: /root/Desktop/Lab2-BufferOverflows/BOF
Buffer overflow vulnerability starting up...
str(shell code) address: bffff20c.
buffer address: bffff1d4.
```

/home/root/Desktop/Lab2-BufferOverflows/BOF and ./BOF

```
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# ~/Desktop/Lab2-BufferOverflows/BOF
Buffer overflow vulnerability starting up...
str(shell code) address: bffff1ec.
buffer address: bffff1b4.
Segmentation fault
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# ./BOF
Buffer overflow vulnerability starting up...
str(shell code) address: bffff24c.
buffer address: bffff214.
```

The reasons are as follows:

The contents of the environment variable are changed. When the program is run directly, the variable holds the execution path of the program, but when the program is run through GDB, the variable holds the execution path of GDB. So the address of the variable will change due to different environment.