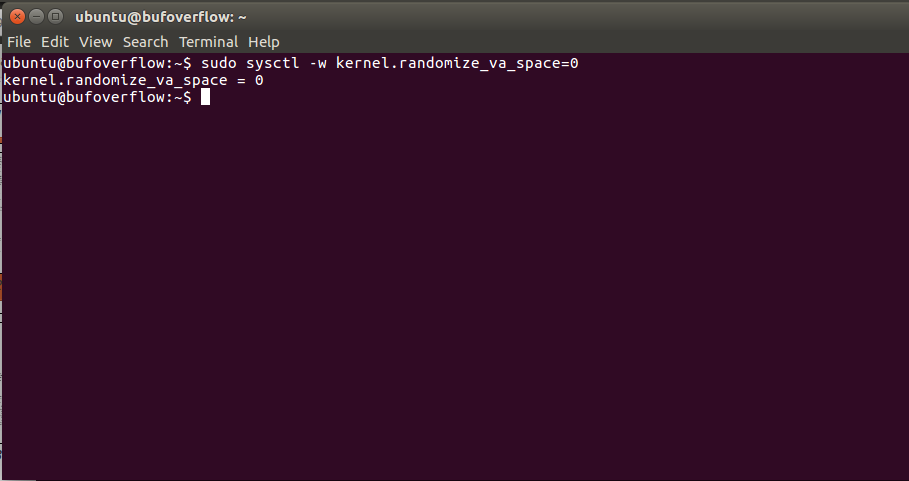
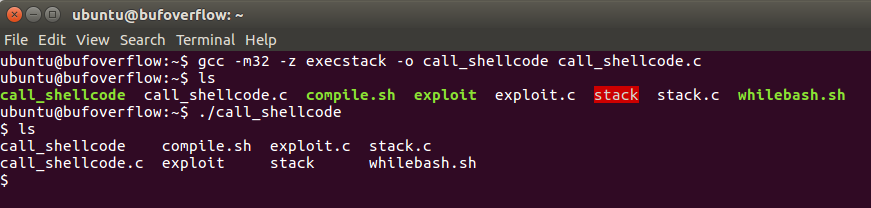
**Labtainer Practical 3: bufoverflow**

**Name: Malshani Prabodha RANCHA GODAGE  
Date: 01/10/2019**

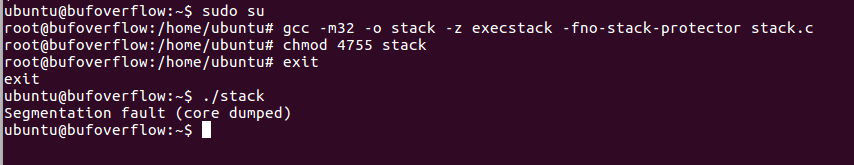
At the beginning I followed initial setup exercises. I have disabled stack randomization to make it easy to guessing exact address for the attack.



Then I compiled the call\_shellCode.c program with executable stack option. Command is gcc -m32 -z execstack -o call\_shellcode call\_shellcode.c and I ran the object file. Then the shell program is running.

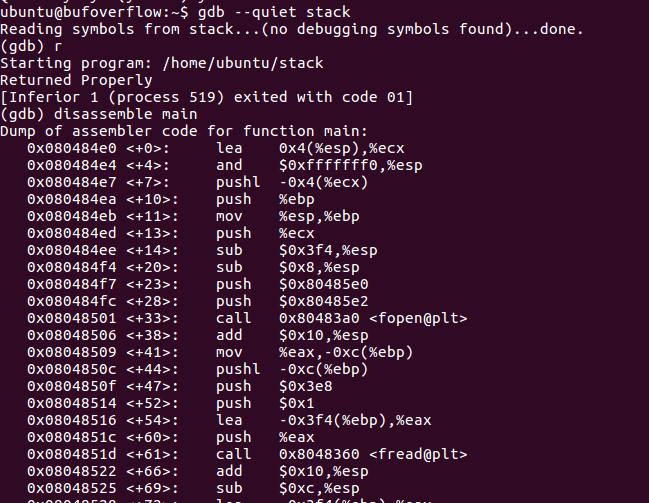


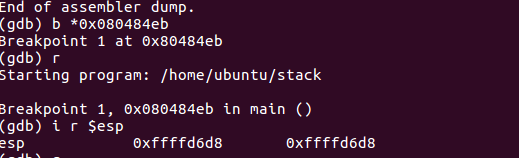
Then I compiled the stack.c file in root access with options to turn off the non-executable stack and StackGuard protections. And also make it setuid permission. Buffer overflow is occurring here, because the data is larger than the memory it supposed to copy.

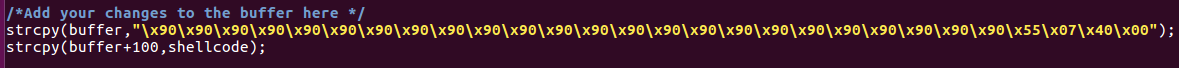


Exercise 1: Exploit the vulnerability

I run the command ‘gdb – quiet stack’ command to enter debug mode in stack program. It showed the assembly output of the main function of stack program. I enter a break pint and run the program and found out value of esp. That means I found out the memory address of buffer. I know the return address is in 60 bytes away from buffer. So I have added 60 (in hex) to buffer address to take the location of the memory that located the address of return address.



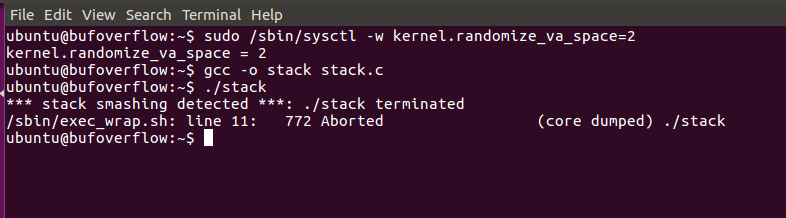




Then I added this address in the end of buffer as big endian style.

Now the memory address of return address changed to another value. Then I compiled and run the exploit.c. It made the bad file which I can attack the stack program. So I compile and run attack stack program.

Exercise 2



After I run the stack with enabling stack randomization I received this message. Whilebash.sh helps to run the program for several times.