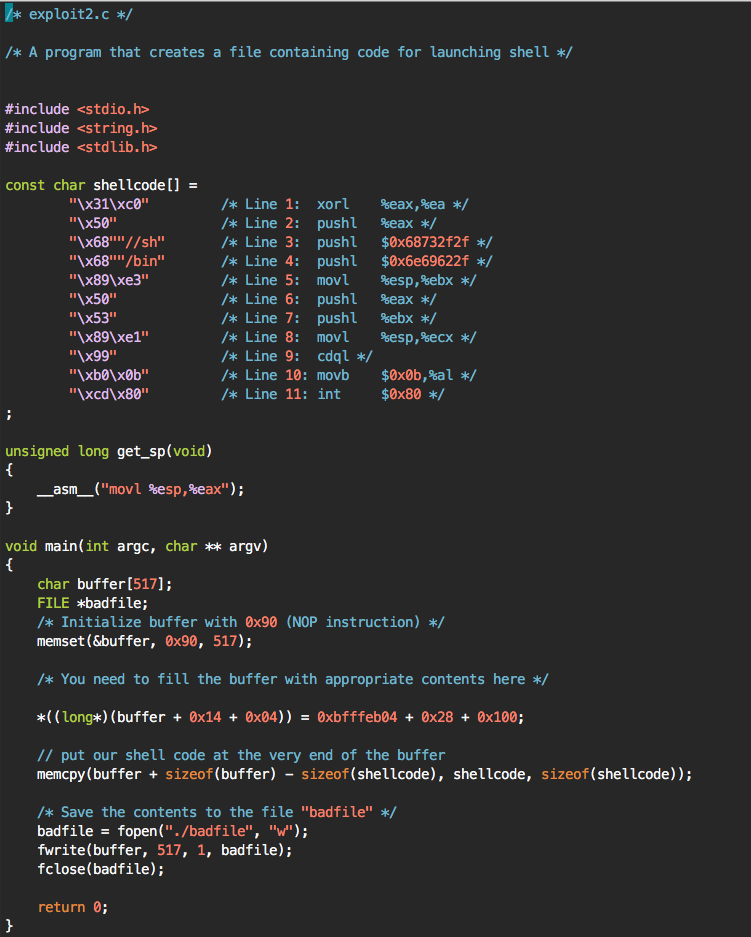
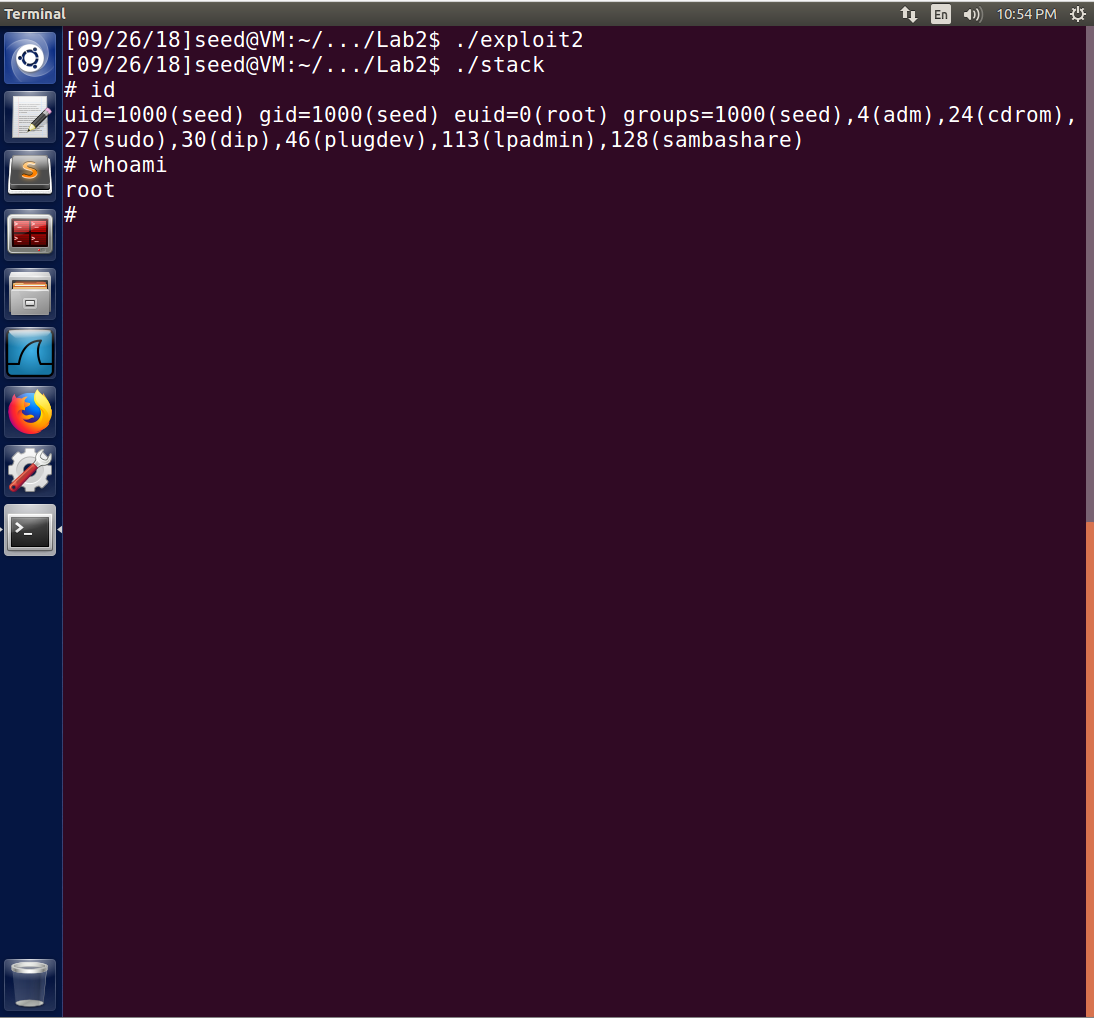
Lab 2 Report

Task 1: Exploiting the Vulnerability

* My Code:

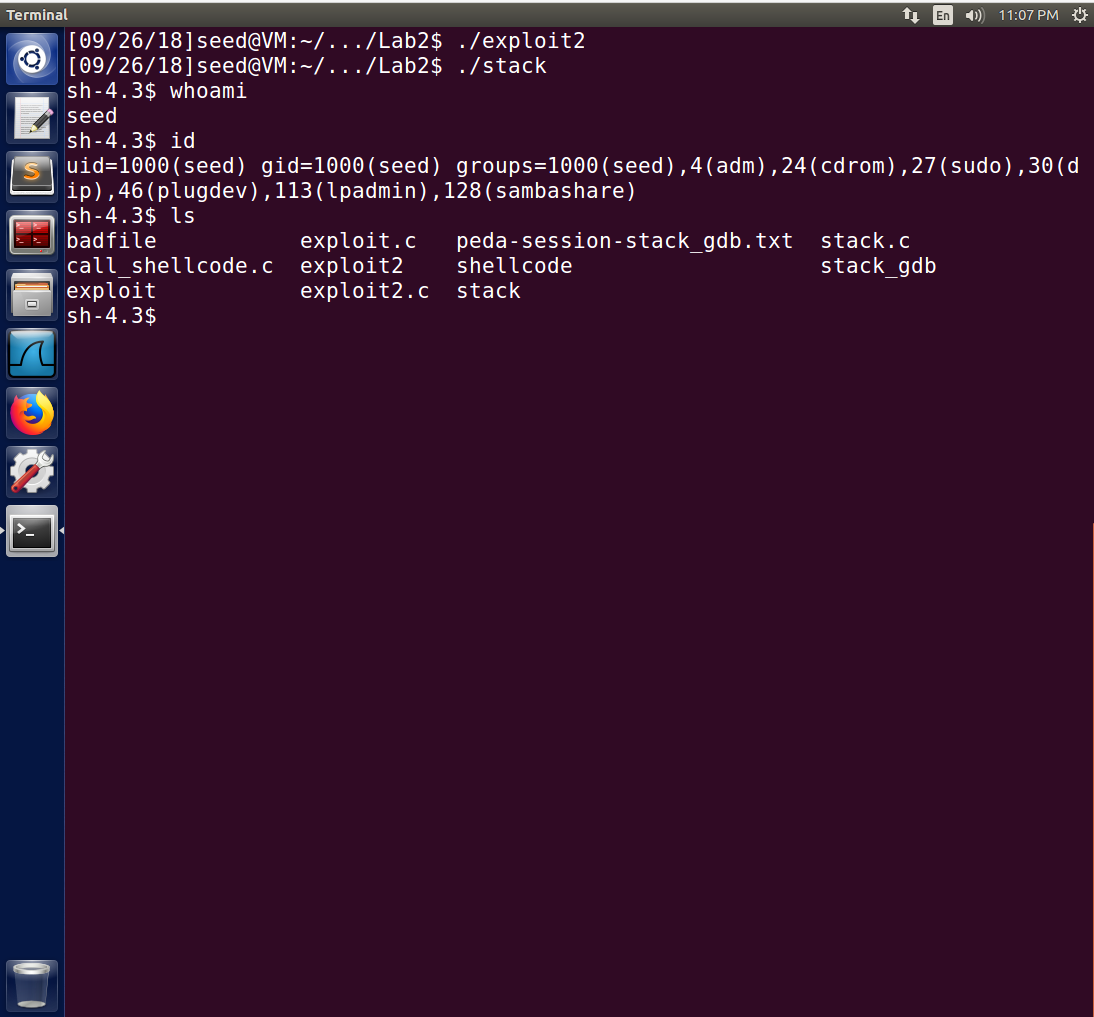


* The important part of the program is the line beneath the fill the buffer comment. The steps to find the hex numbers were demonstrated in class. We fill the buffer with correct number of spaces and the copy the contents into the memory
* The result:



Task 2:

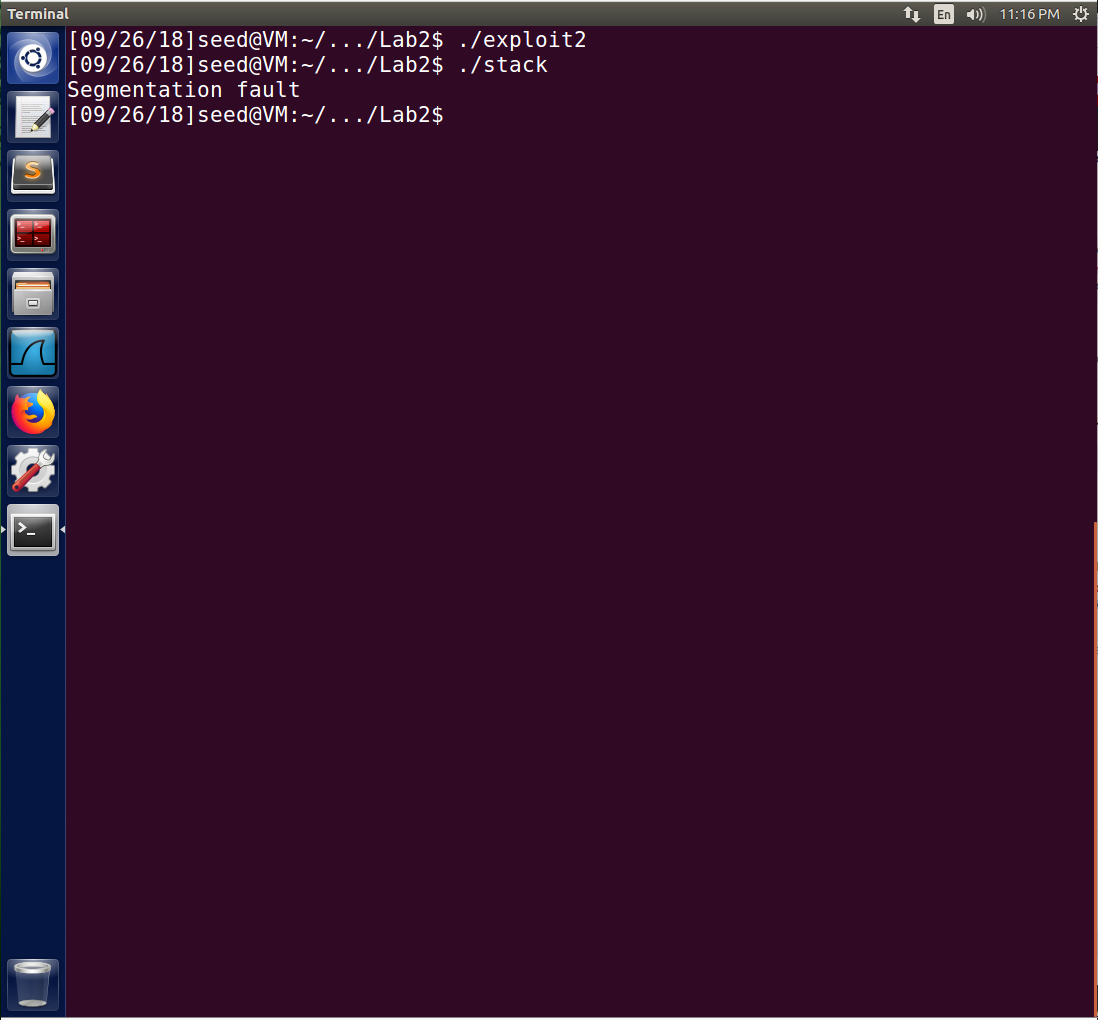
* When running ./exploit2 and then ./stack with /bin/sh pointing to bash, we get the results below:



* What seems to happen is that it pulls up a bash shell, however, it does not have root user privileges. Most likely is that when bash is called, it automatically drops root user privileges in order to protect from having a root user shell invoked for someone who shouldn’t have the access.

Task 3: Address Randomization

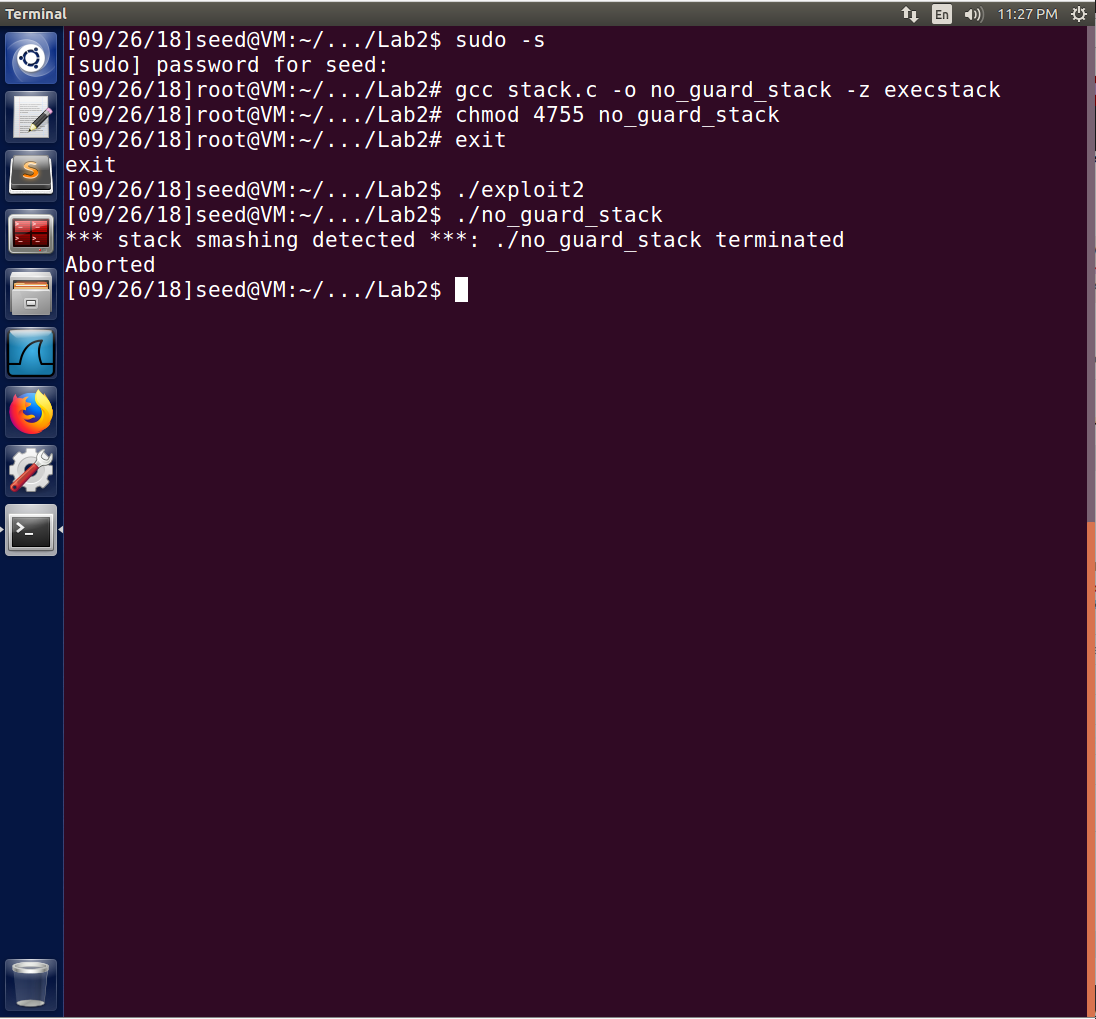
* When running the code with address randomization enabled, we get the following output below:



* When run with address randomization enabled, we get a segfault. We can’t even pull up a shell. This is most likely due to the fact that the stack addresses are no longer in numerical order and have been randomized. That means that our NOP sled is now useless and it will be impossible to accurately determine what address we need to insert our malicious code into. When running the ./stack executable in an infinite loop for about 2 minutes, I was still unable to get the program to work properly without seg-faulting. However, it is theoretically possible for us to brute force it given enough time. If run for a long enough period of time, eventually the correct address in the stack will be guessed and the program will run as it should.

Task 4: Stack Guard

* When compiling the stack program without disabling the stack guard and running it, we get the output below:



* When running with a stack guard, there is information put into the executable that lets the operating system know if the stack is being tampered with. When the operating system sees this, it terminates the program before any malicious code can be run.