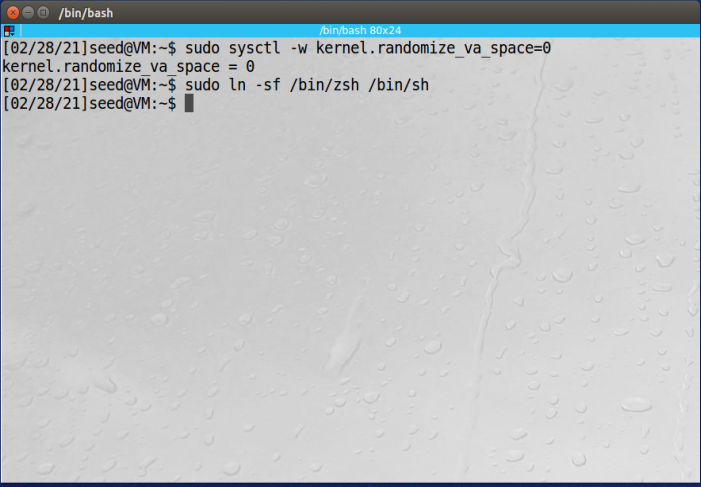
Daniel Quiroga

Network Security

Homework 2

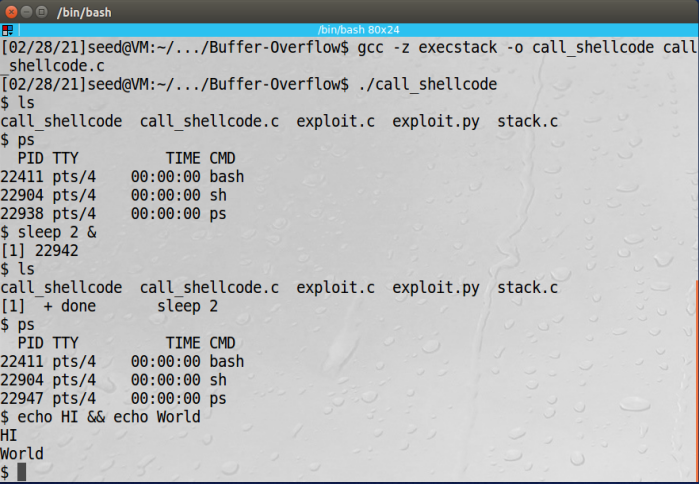
**2.1 Turning Off Countermeasures**

Running through the commands in the section of the description:



**2.2 Task 1: Running Shellcode**

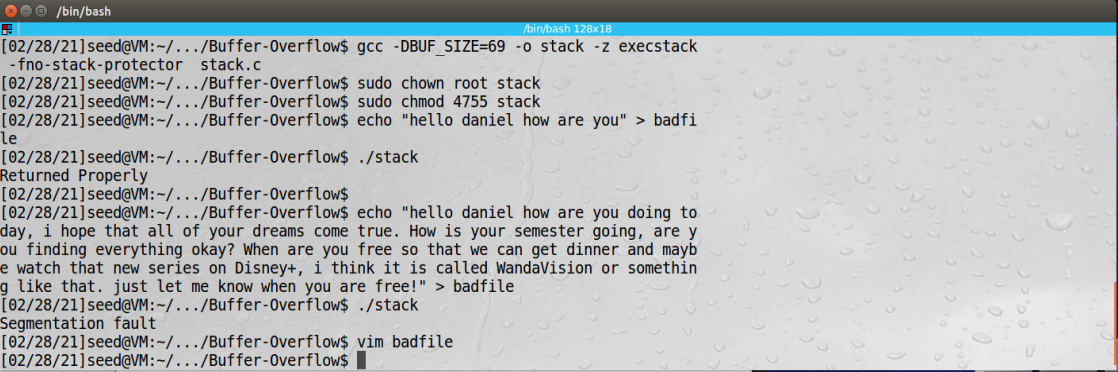
I compiled and ran the executable and tested our a few shell commands to test to see if the shell does what is needed:



It appears all the simple commands and serial commands worked as expected just like a typical shell would work as well. To make sure that the file compiled with no warning simple include of an .h file was needed.

**2.3 The Vulnerable Program**

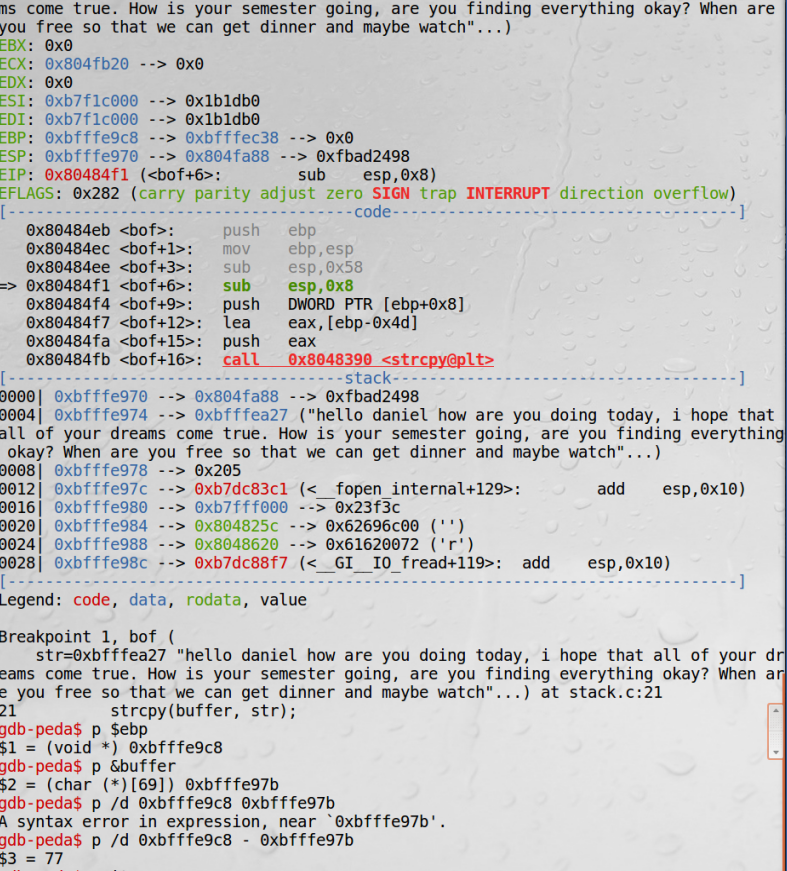
Here I run through the commands and then create a badfile where I see if the program runs as expected with my buffer and if there is a buffer flow attack and so forth. I perform the commands to change the permissions of the file as well:



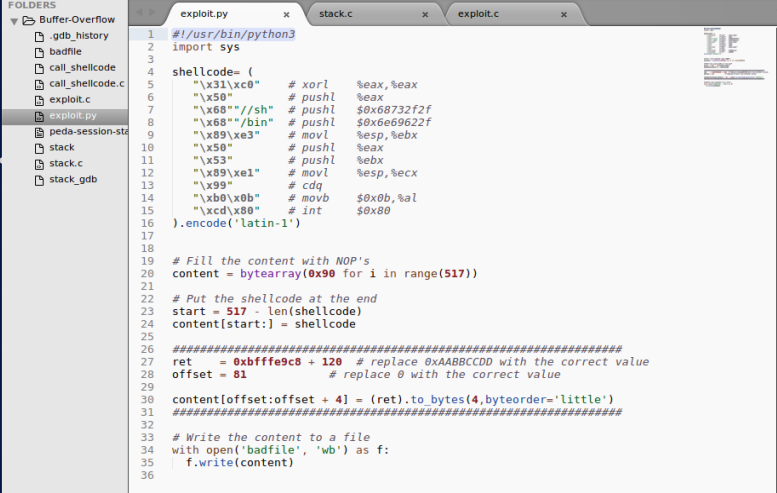
Above we see that when the file contained a small amount of characters that the stack returned properly. But when I wrote the longer piece of characters along with the given 69 buffer size I would have expected a sort of error or something and we got a segmentation fault meaning that we were able to confirm the functionality of the program and executable.

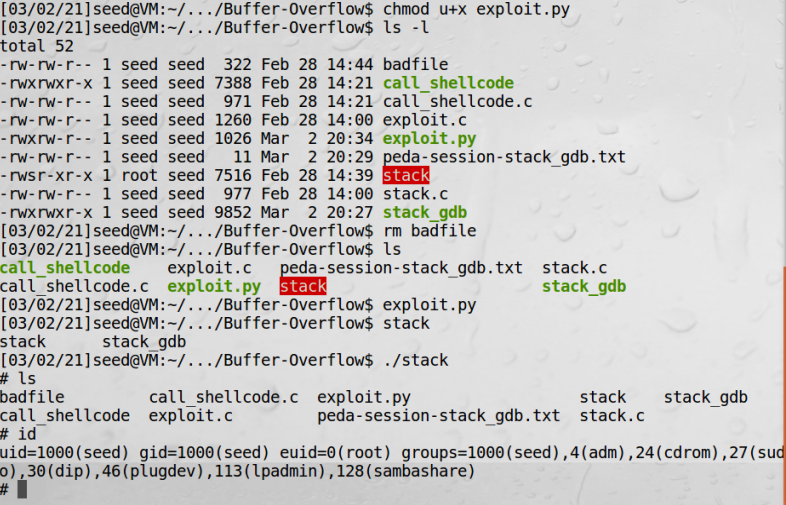
**2.4 Task 2: Exploiting the Vulnerability**

I am getting all the relevant information for my buffer attack in the gdb below:

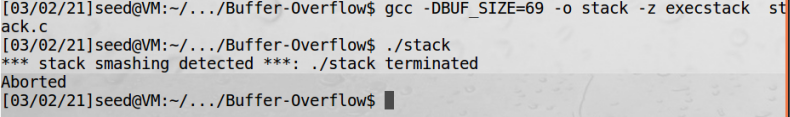


I updated the python file with all relevant data and then made the python file an executable and was able to get a shell or essentially preform the buffer overflow attack:





**2.7 Task 5: Turn on the StackGuard Protection**



As seen, we get a stack smashing error essentially preventing us from executing the executable. This was not seen when we did it in the previous task and hence shows how the buffer overflow attack was performed.

**2.8 Task 6: Turn on the Non-executable Stack Protection**

I was not able to get a shell when compiling this way. The problem is that the stack guard was enabled and hence did not allow us to even run the program. This makes the attack difficult since it will prevent the very methodology that was used to perform this attack.

