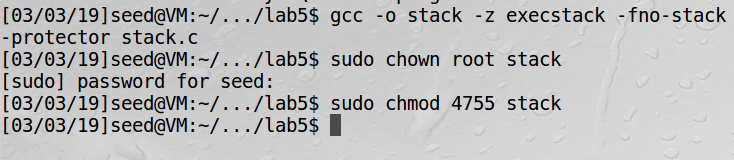
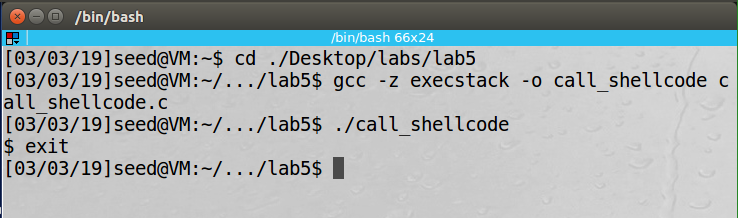
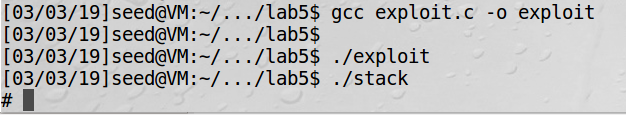
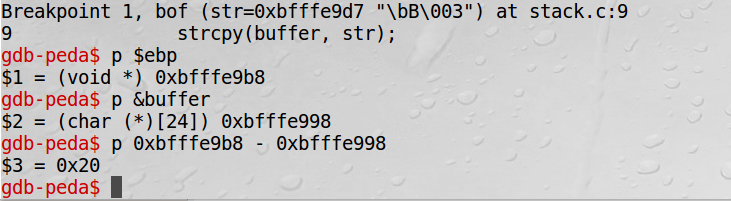
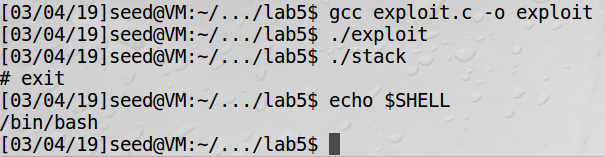
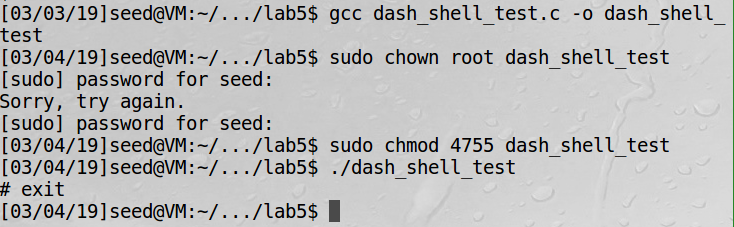
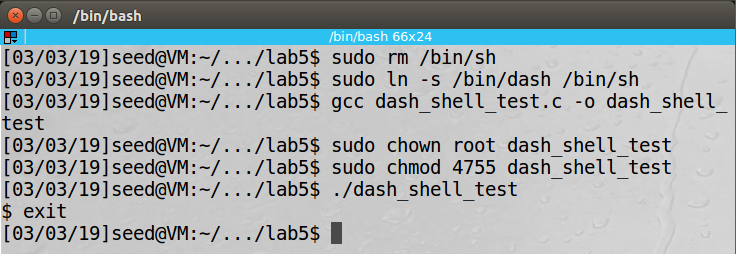
Task 1:



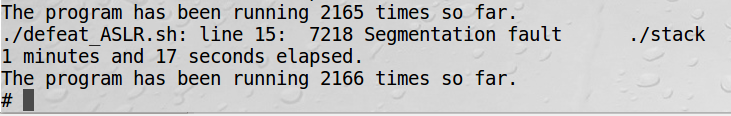
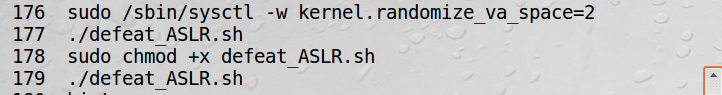
Task 2: The completed code for exploit.c is provided in lab5-source-code. This is the gdb debug to find the length of the buffer so we rewrite the return address.



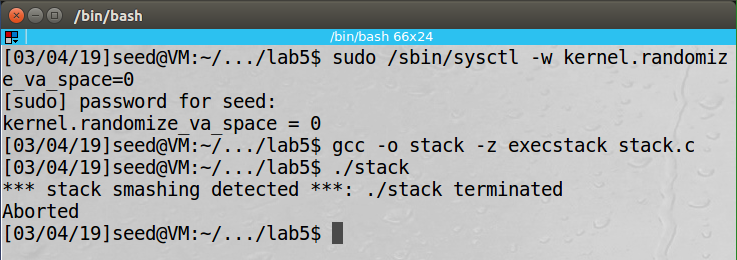
Task 3: The first picture is with the setuid commented out and to no surprise was do not get the root shell. The second picture is when I uncommented the setuid and with that the RUID and EUID match giving us the root shell. Lastly with the additions to the shellcode when we run the stack program we get the root shell using the buffer overflow attack.



Task 4: It was quite surprising to see that running it multiple time seemed to redue the time it took to get the root shell. The second picture was the third time running it and it only took a minute and 17 seconds however the first time only took about 30 minutes.



Task 5: - When I enabled the stack protector it detected the buffer overflow and aborted the execution.



Task 6: - I did end up getting the shell however they reason I didn’t get the root shell was because I running in the bash shell and not the zsh shell when running the set-uid.

