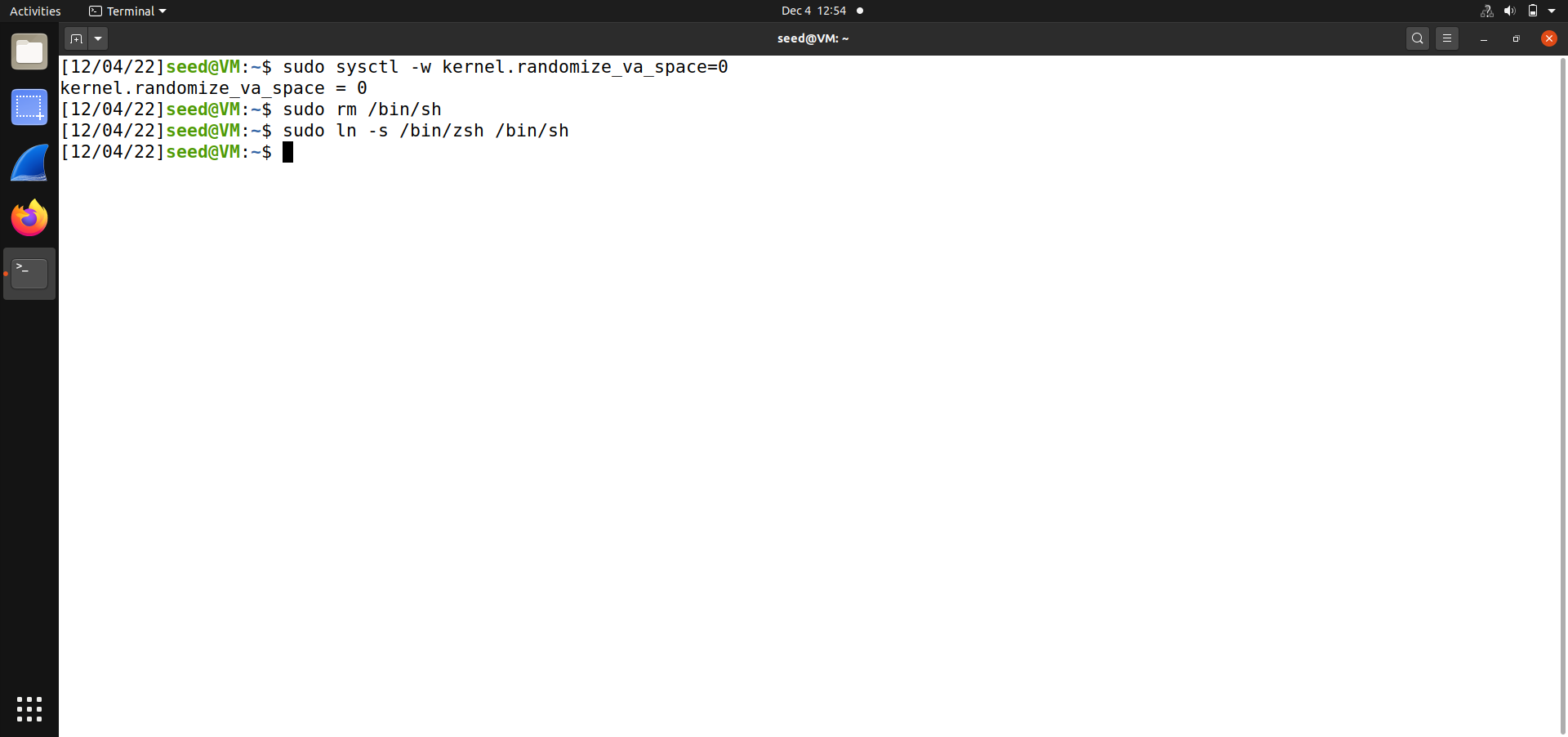
Name:Ramya Ajay

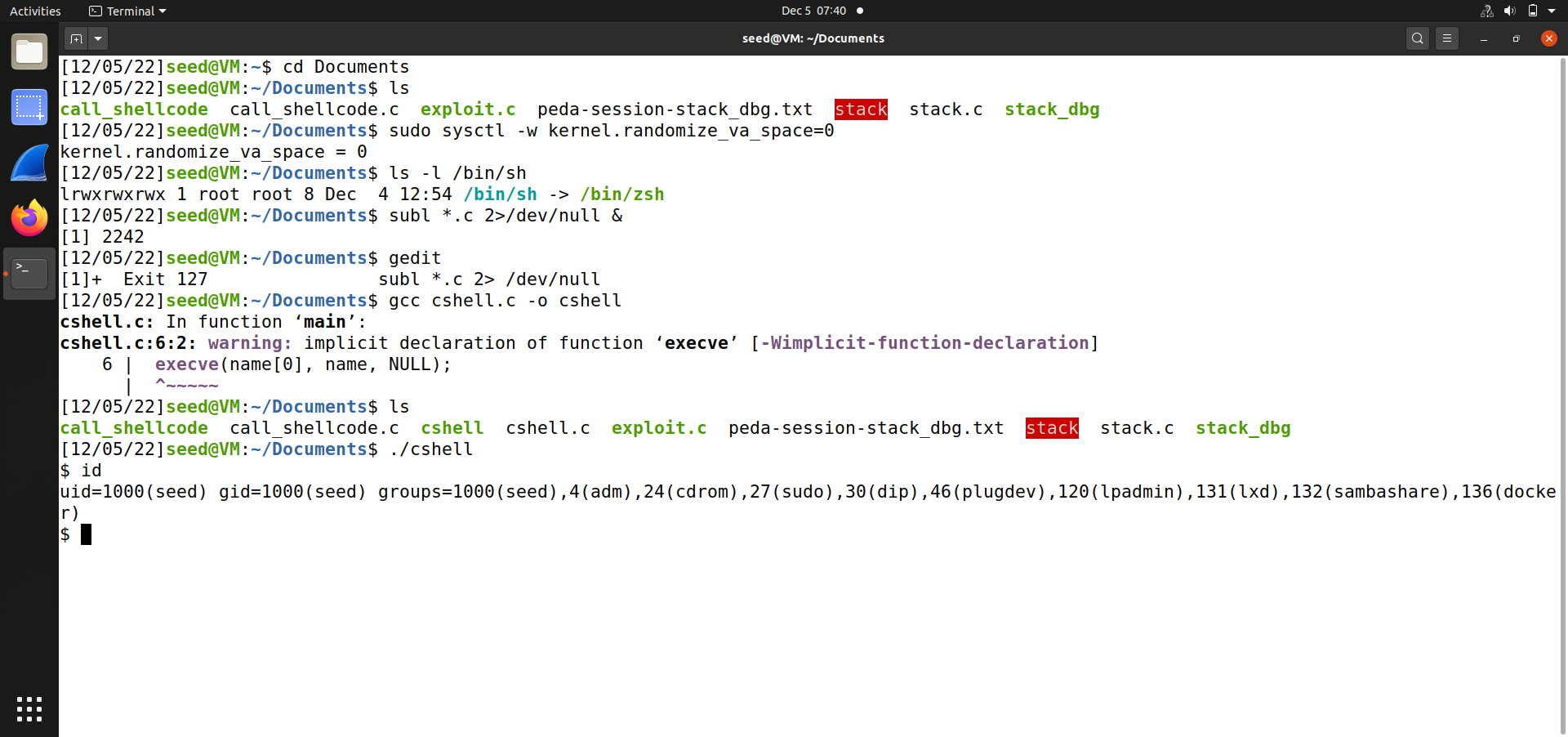
Roll No:CB.EN.P2CYS22004

Date:5/12/2022 Secure Coding Lab 11-13

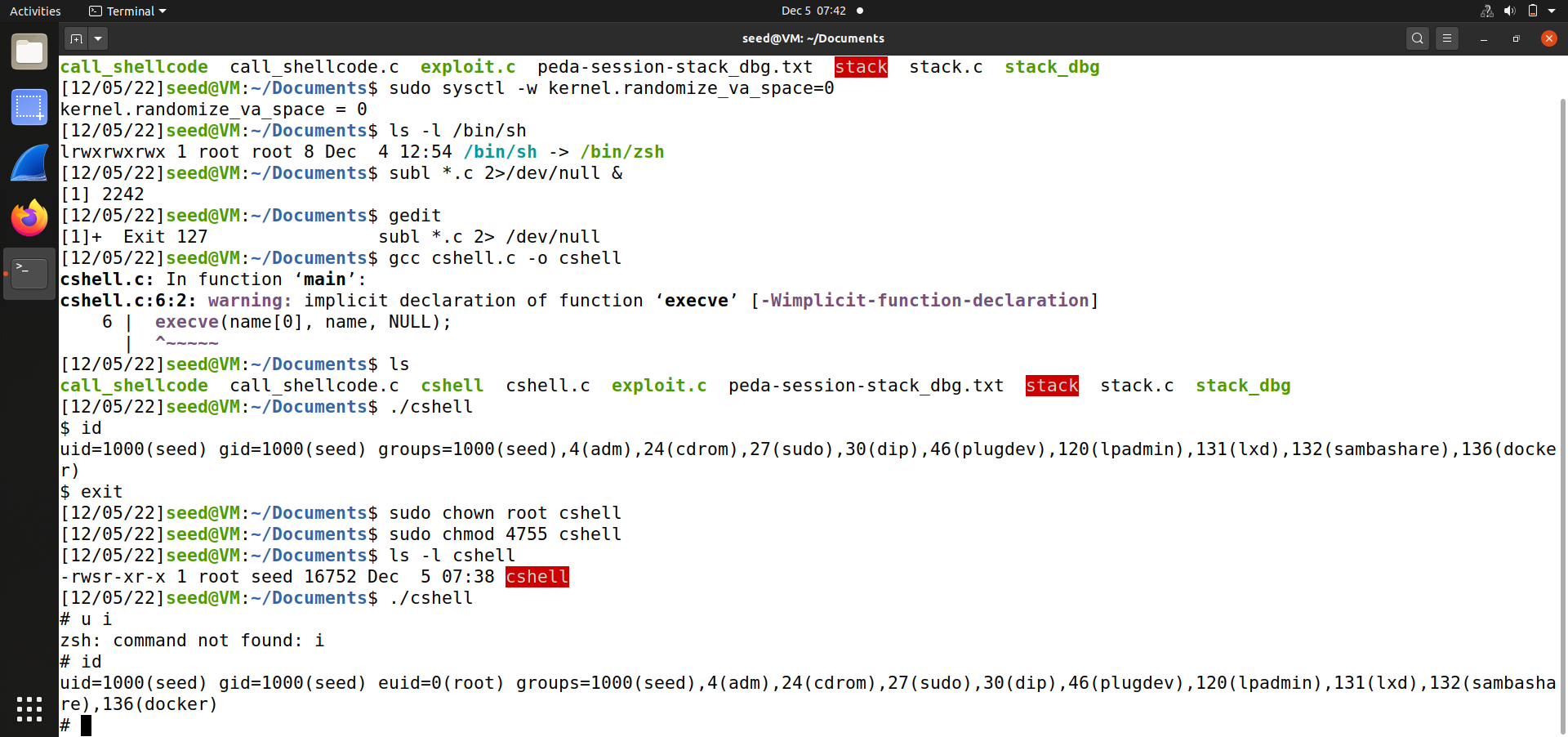
1) In order to perform the Buffer Overflow attack, first we disable the countermeasure in the form of Address Space Layout Randomization. If it is enabled then it would be hard to predict the position of stack in the memory. So, for simplicity, we disable this countermeasure by setting it to 0 (false) in the sysctl file, as follows:



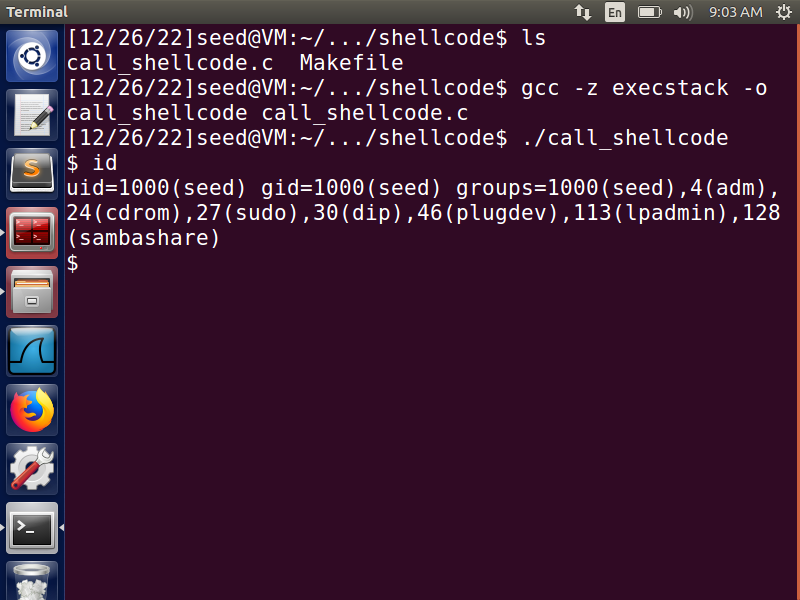
2) Running Shellcode :



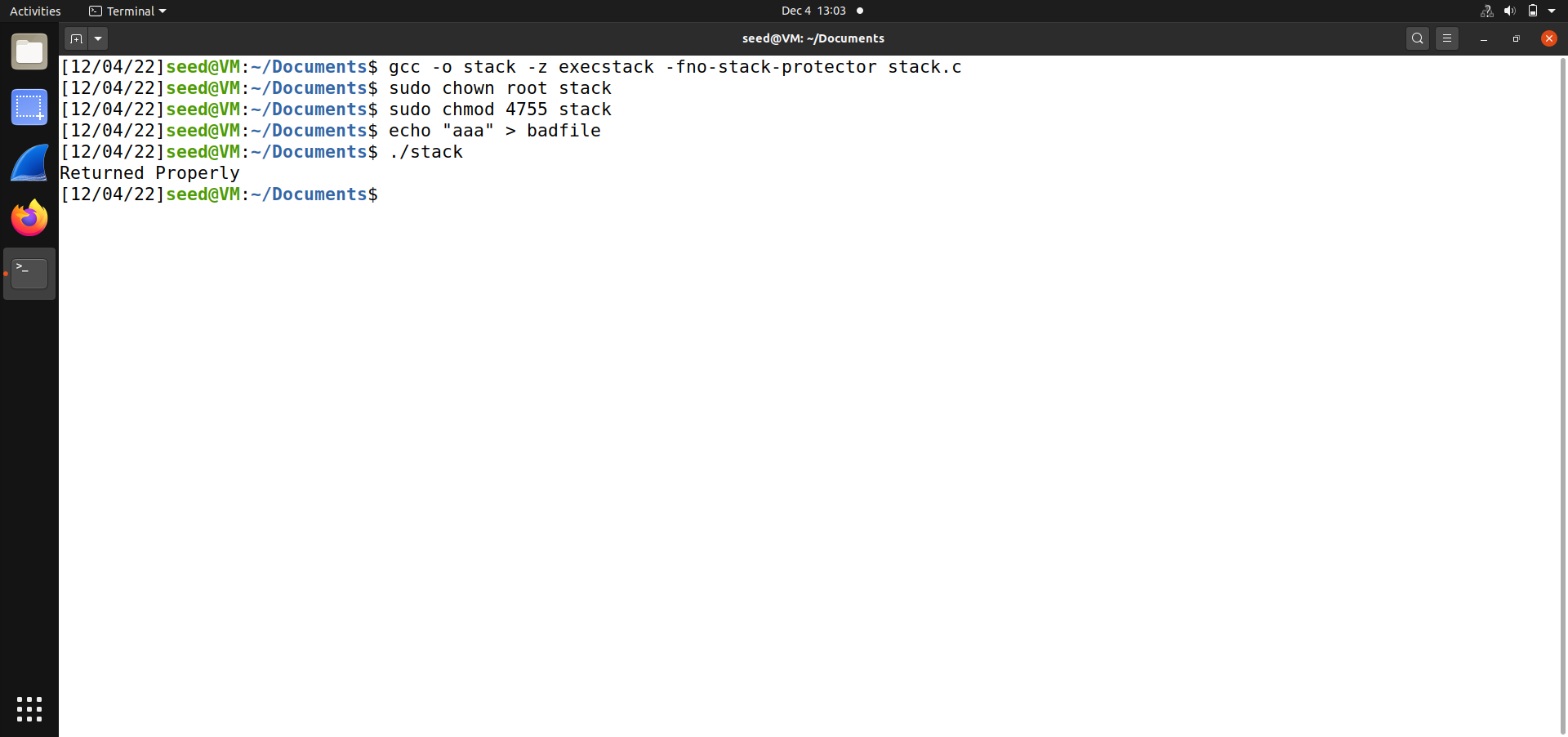
Running shellcode as setuid program:



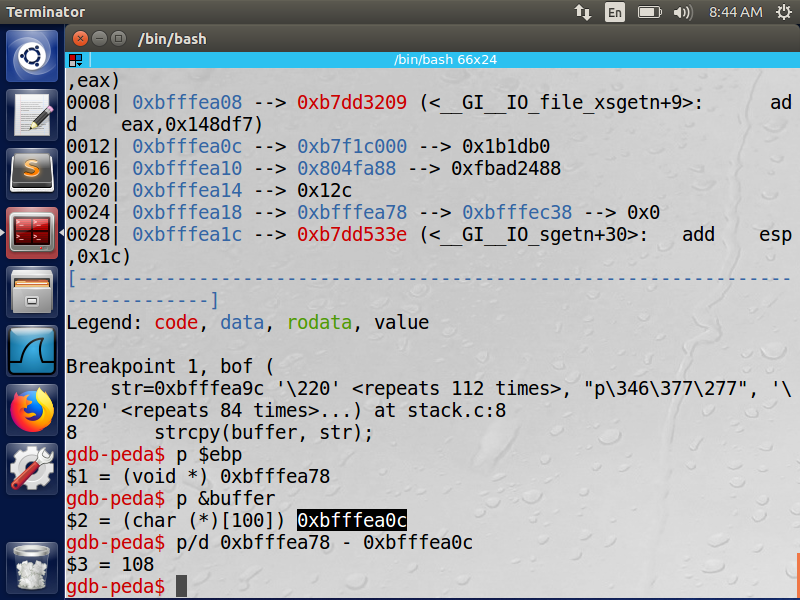
Assembly version of the above shellcode:

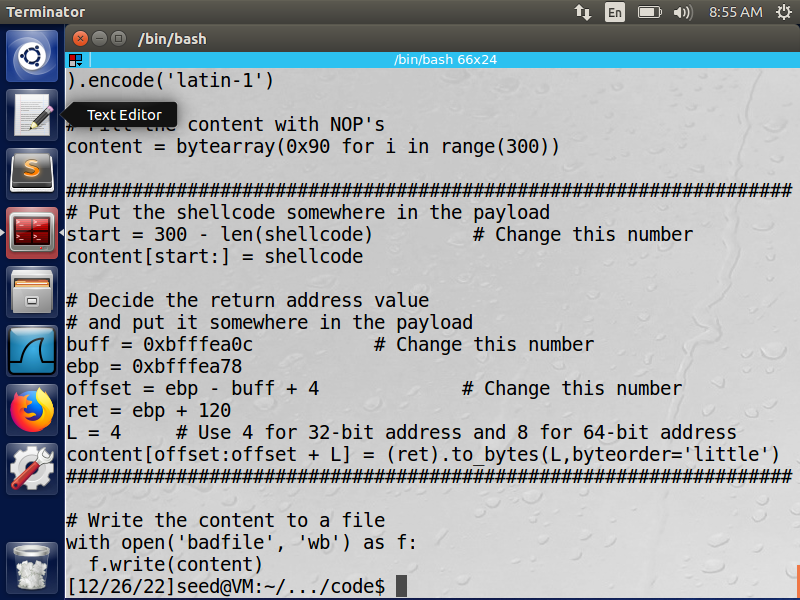


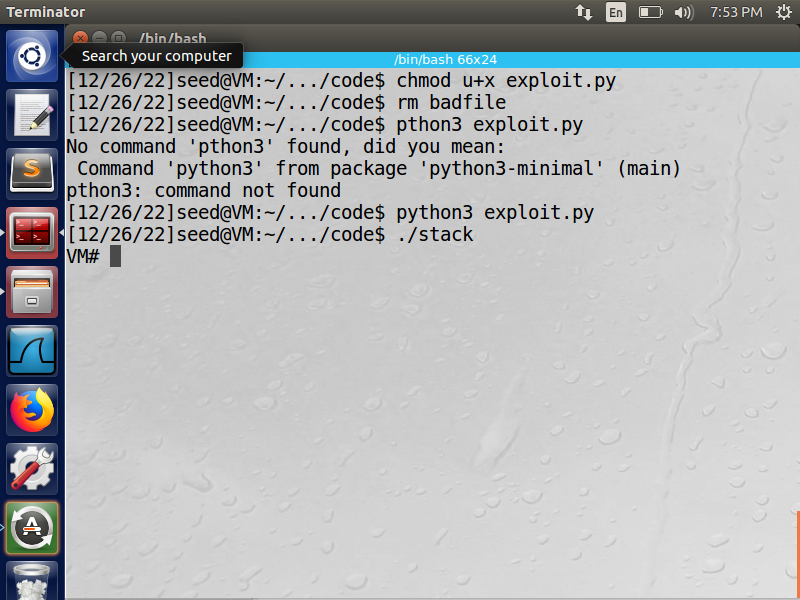
3) Next, we compile the given vulnerable program stack.c and while compiling, we disable the StackGuard Protection mechanism and make the stack executable by passing the respective parameters to the command. Also, the compiled program, stored in ‘stack’, is then made a SETUID root program.



4) Since, we have disabled Address Space Layout Randomization, we know that our process will be stored in around the same memory always in the stack. So, in order to find the address of the running program in the memory, we compile the program in debug mode. Debugging will help us to find the ebp and the offset, so that we can construct the right buffer payload that will help us to run our desired program.

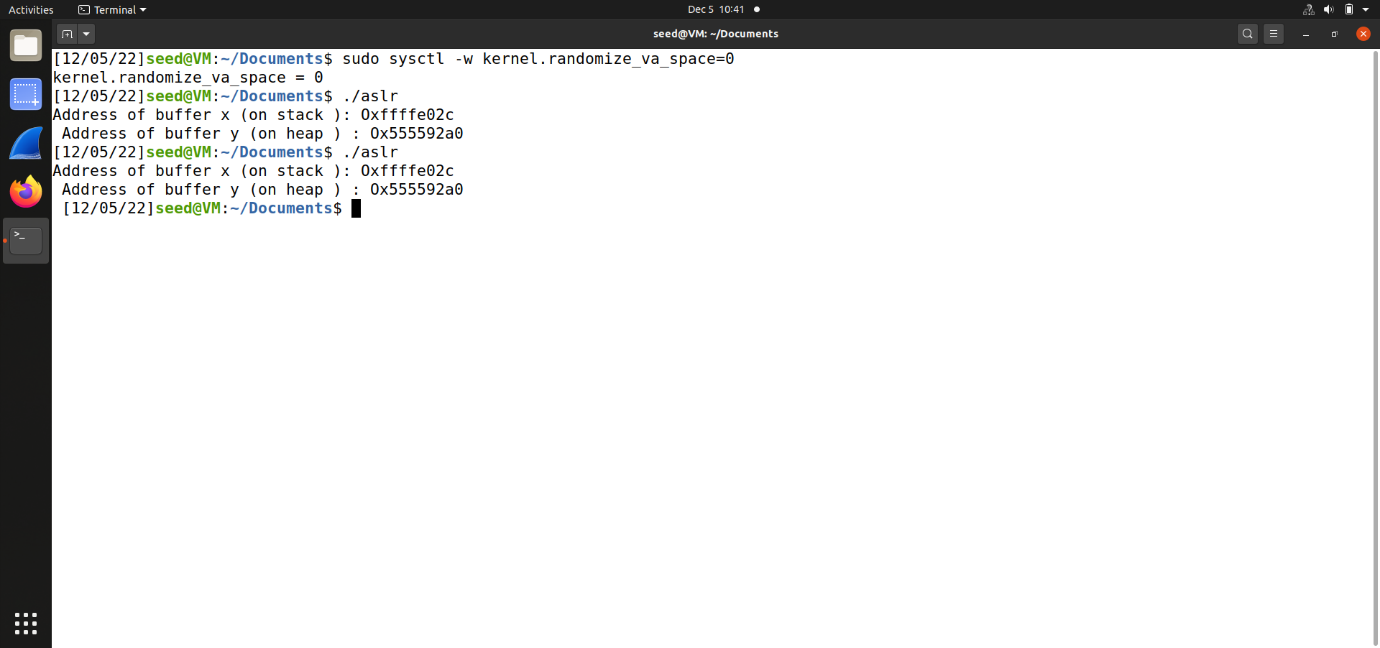


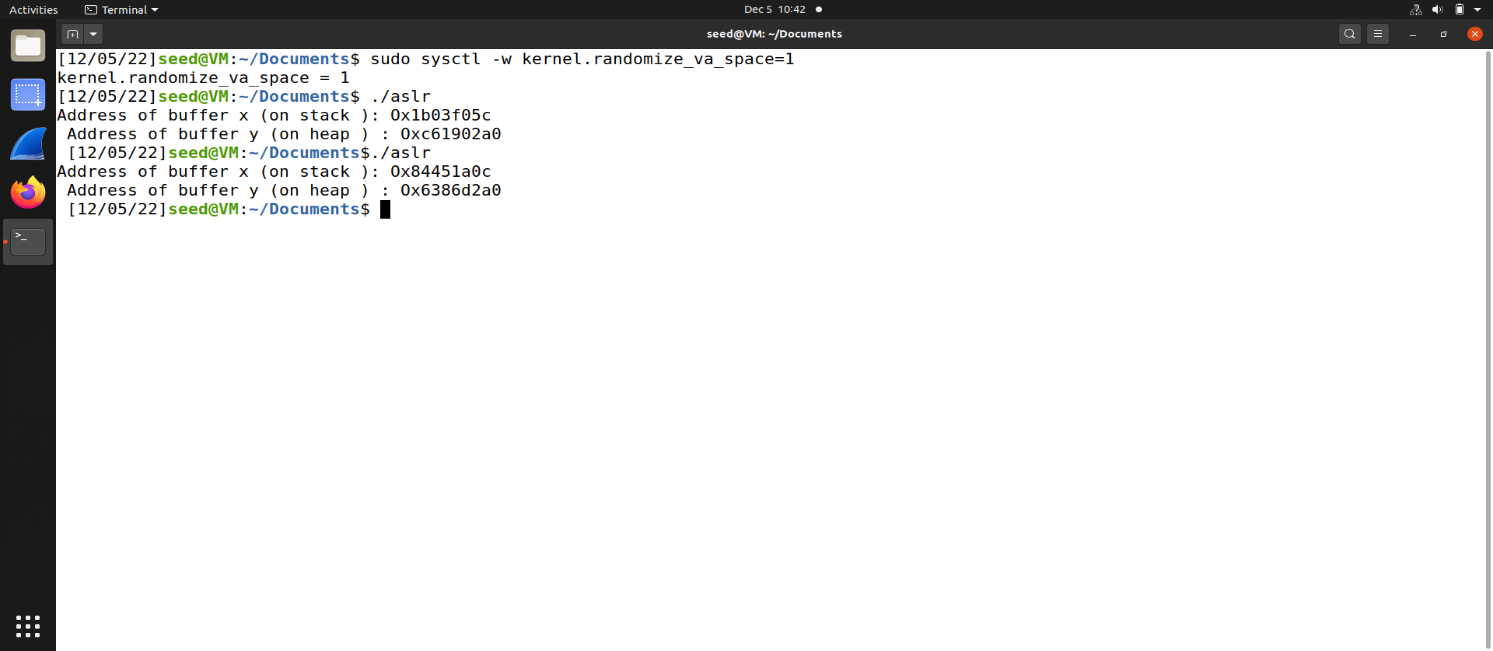


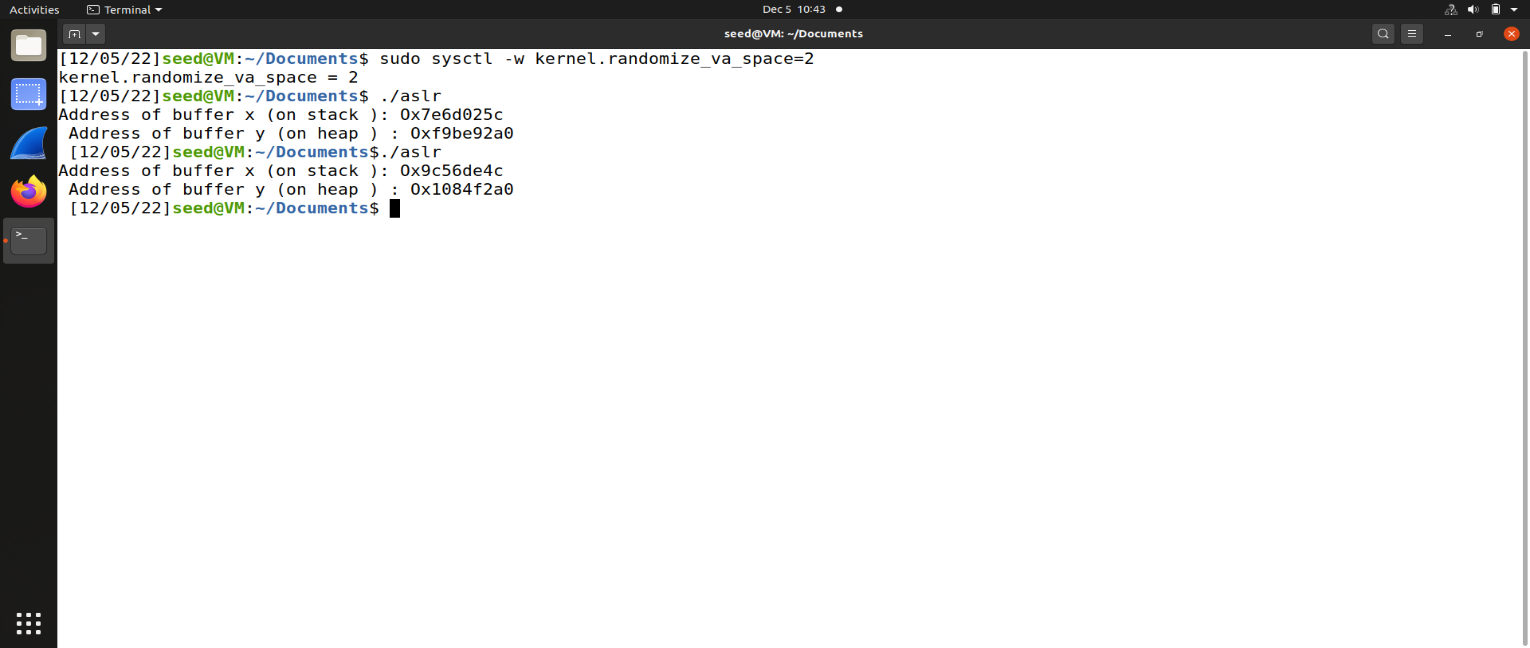


5)Address Randomization

After compiling the code, we run it under different randomization settings. Users (privileged users) can tell the loader what type of address randomization they want by setting a kernel variable called k ernel . randomiza\_v a\_space . As we can see that when the value 0 is set to this kernel variable, the randomization is turned off, and we always get the same address for buffers x and y every time we run the code. When we change the value to 1, the buffer on the stack now have a different location, but the buffer on the heap still gets the same address. This is because value 1 does not randomize the heap memory. When we change the value to 2, both stack and heap are now randomized.

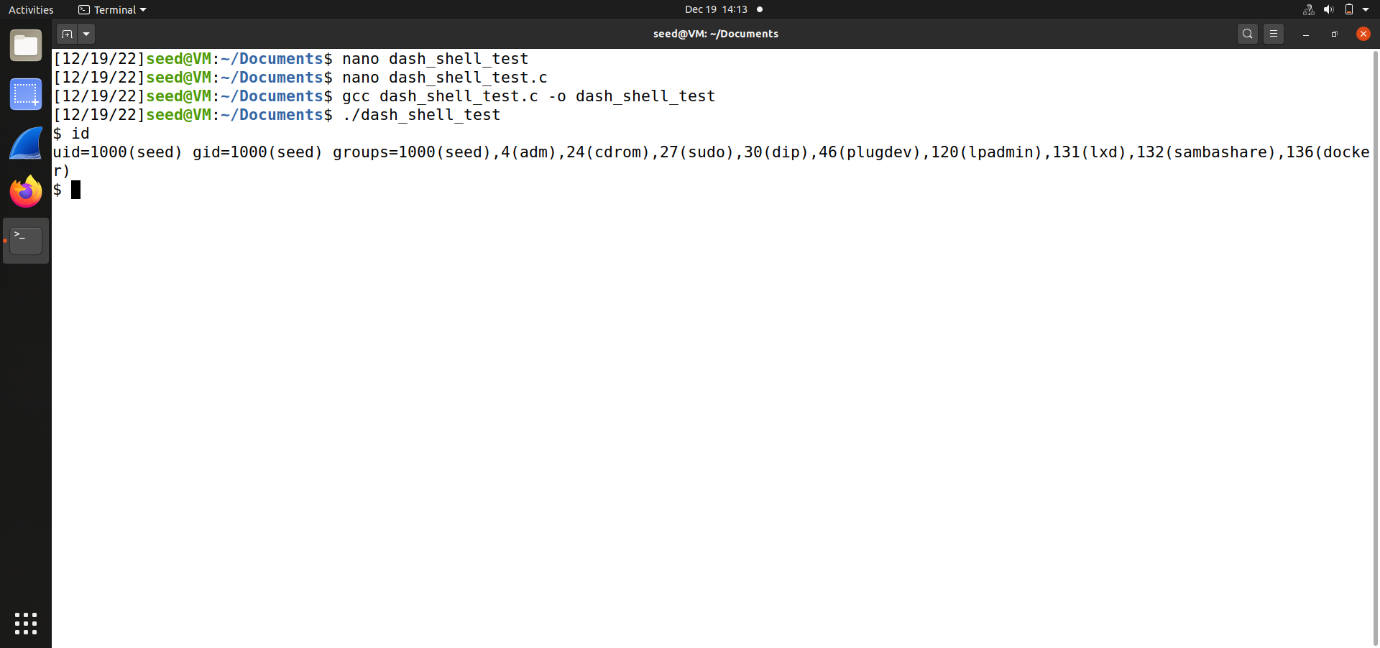






6) Defeating dash countermeasure





After removing the comment of setting the user id to 0, and running the program, we get:

