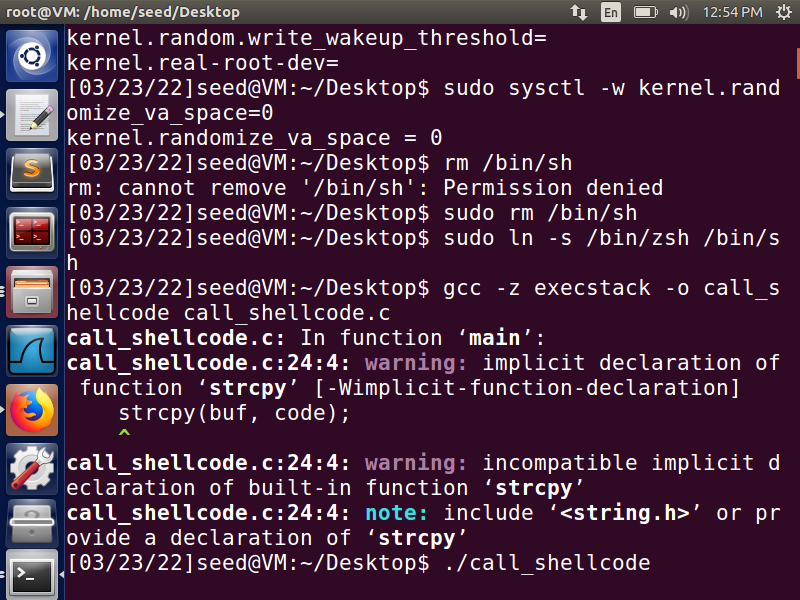
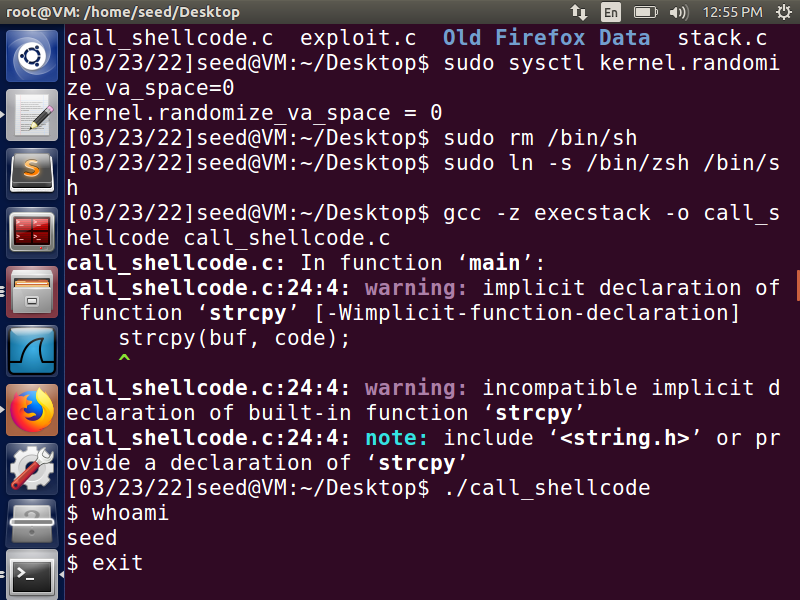
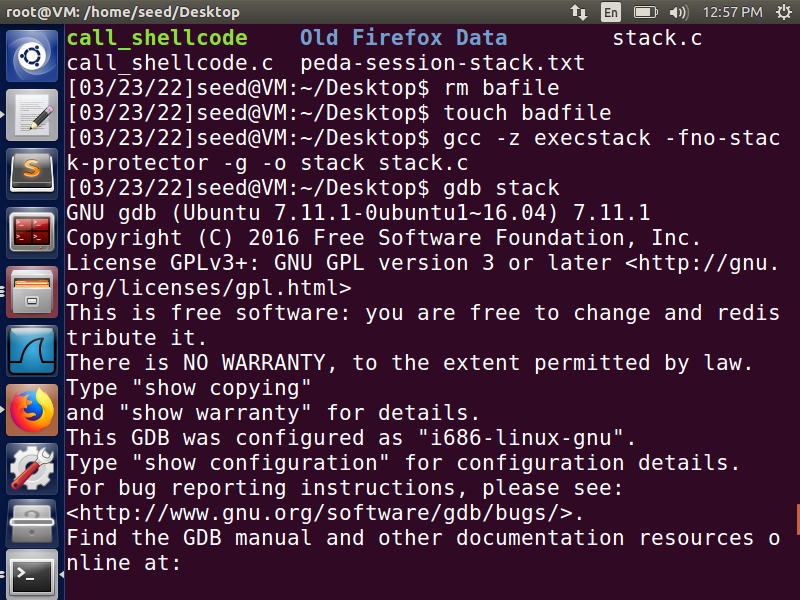
Assignment 3

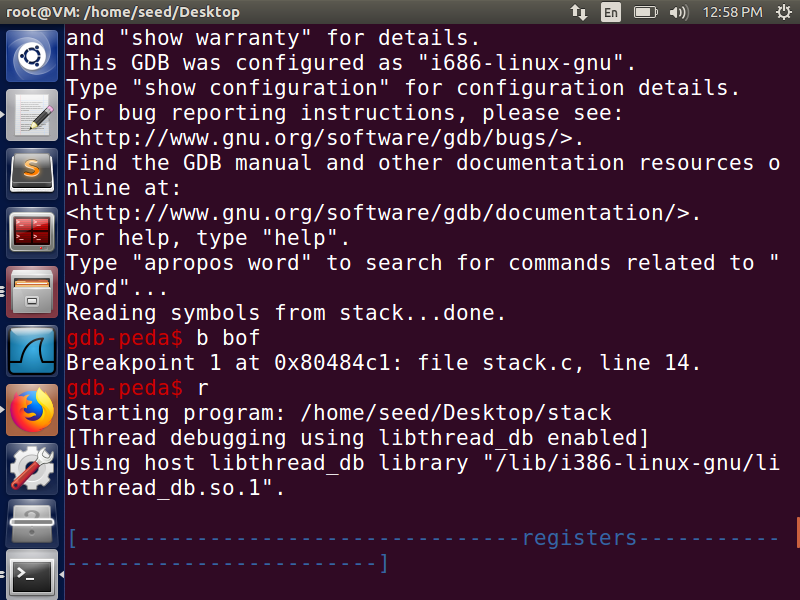
Task 1: Exploiting the vulneraibility.

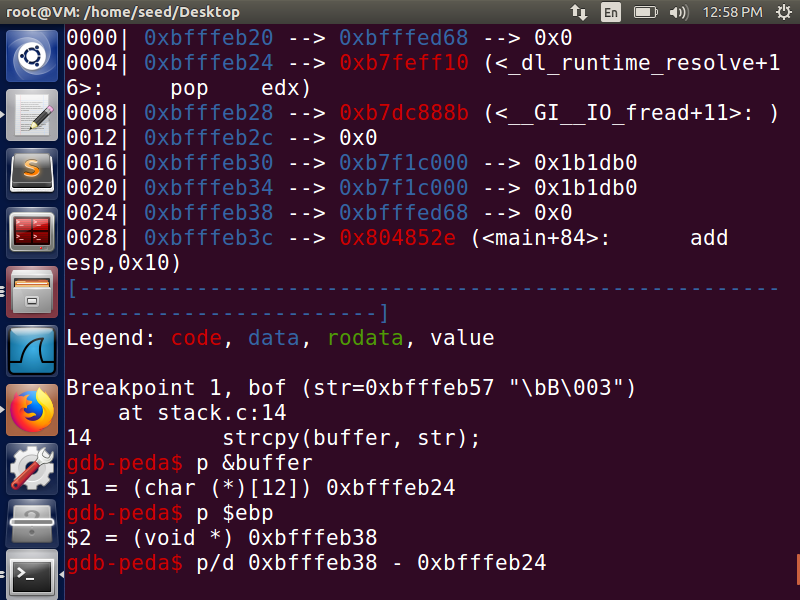
* First I ran the “sudo sysctl -w kernel.randomize\_va\_space=0” to disable the default address randomization which is provided by Ubuntu and everal other Linuz based systems.
* Then I ran the “sudo rm /bin/sh” and “sudo ln -s /bin/zsh /bin/sh” to change my bin/sh to bin/zsh.
* Then I complied the the call\_shellcode.c using the command “gcc -z execstack -o call\_shellcode call\_shellcode.c”
* I then exucted the call\_shellcode using ./call\_shellcode.
* I have already turned off the address randomization, then made the stack executable and turned off the stack guard protection.
* Compile the exploit program and create the badfile.
* After making changes to the exploit.c, I compile it using “gcc -o exploit exploit.c” and ran “./exploit” which creates the badfile and then ran “./stack”.
* After executing the stack program, the output is shell prompted indicating that we have exploited the buffer overflow mechanism and /bin/sh shell code has been executed.
* Following are the screenshots.

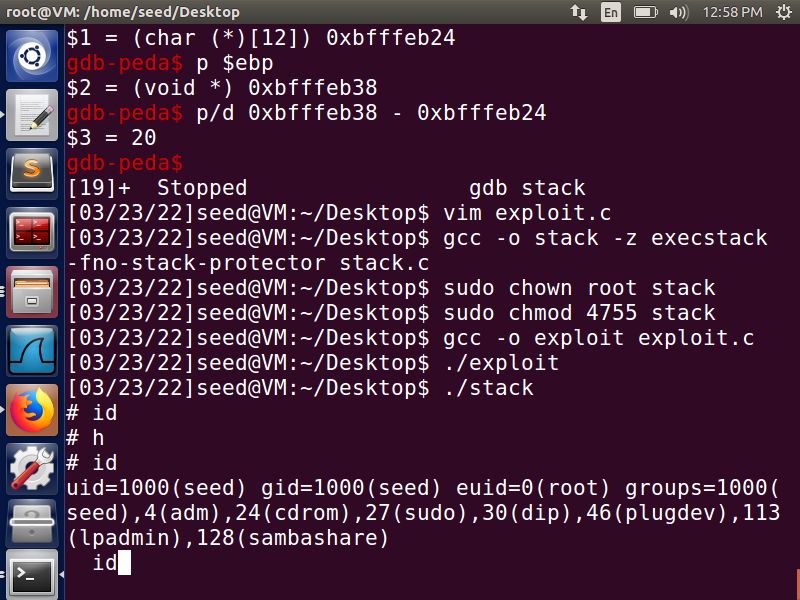


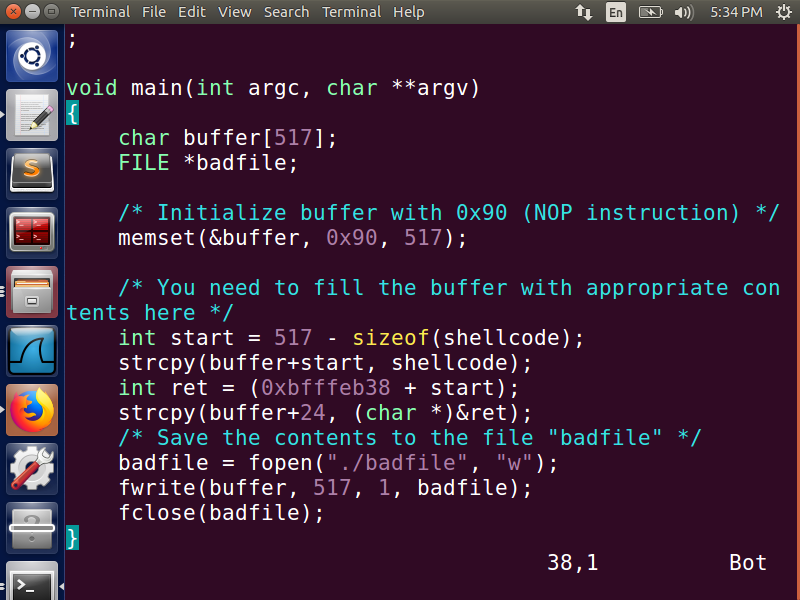








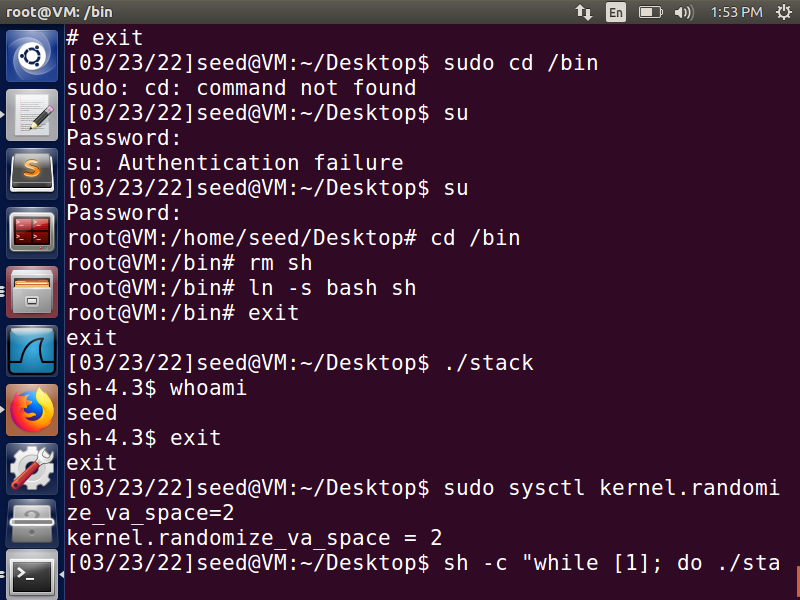




**How I exploited the program.**

* I used gdb debugger to find the return address.
* Inserted a breakpoint at the start of function where buffer overflow attack may occur.
* Printed the address of the start of the buffer.
* Printed the value of ebp register.
* Calculated where the return address is, so I can change the return address and exploit the vulnerability.

**2. Protection in /bin/bash**

* After running the “su” “cd/bin” and linking the bin/sh to the bin/bash when we try to the run the same attack we are getting the normal seed access and not the root access we were getting in the previous step. 

**3. Address Randomization.**

* Earlier in order to perform the buffer overflow attack we had switched off the Linux’s defense mechanism against buffer overflow by turning off the address randomization.
* For this part we activate the address randomization using the command “sudo sysctl -w kernel.randomize\_va\_space=2”.
* I compiled the stack program using stack guard protection and making the executable of the stack.
* When tried to run for the first time using “./stack”. I got segmentation fault.
* As suggested in the assignment. When I try to the run this in an infinite loop , I keep getting segmentation faults. But I think that with patience and letting the program run for a few minutes, I might be able to get the root access.

**4) Stack guard.**

* We now compile the program with the Stack Guard protection.
* We do this using the command “gcc -o stack execstack -z stack.c”
* When we run the excutable ./stack the system recognizes the buffer overflow attack and gives us the smashing detected segmentation fault and aborts the program.

