**2.1 Lab Environment**

For this lab we will disable address space randomization to allow for guessing the exact address easier.

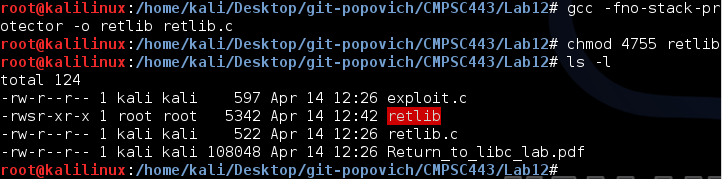
P:\CMPSC443\Git\CMPSC443\Lab11\DerandomizeAddressSpace.PNG

We will leave ExecShield Protection on for this lab, being that we are attempting the “return to lib-c” attack.

We will also have to make sure that we compile our files with the gcc flag “-fno-stack-protector”. This will allow us to have a buffer overflow to overwrite the return address.

**2.2 The Vulnerable Program**

Next, we will compile retlib.c and give users running it root privileges.

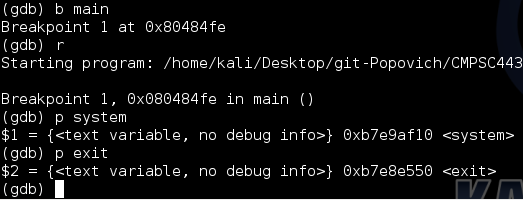


The “retlib” program once again has a buffer overflow vulnerability. It first reads an input of size 40 bytes from “badfile” into a buffer of 12 bytes, causing the overflow.

**2.3 Task 1: Exploiting the Vulnerability**

We will be using “exploit.c” to create “badfile”. The goal is that once badfile is created with malicious payload, upon running “retlib”, we will be granted with a root shell via lib-c.

Before doing this, we must find the location of system, address of “/bin/sh”, and pass the address of “/bin/sh” to the system function. We can find the location of system by using gdb as explained in class.



Here, we also find the address of exit(), which will make quitting the program look less suspicious.

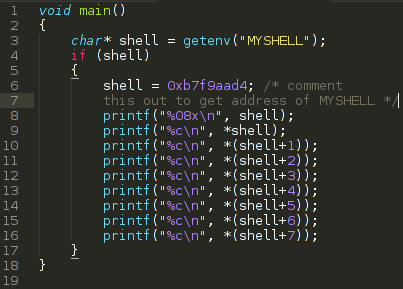
Next, we need to find the address of “/bin/sh”. To do this, our plan of attack will be to create an environment variable with value “/bin/sh”. This is accomplished with the command “export MYSHELL=/bin/sh”. Using the program reviewed during class, we print out the location of “MYSHELL”.

P:\CMPSC443\Git\CMPSC443\Lab12\myshell.PNG

Although this address seems correct, for whatever reason the exploit did not work at this address. The exploit however, does work at the address below found by searching through lib-c for the string “/bin/sh”.

P:\CMPSC443\Git\CMPSC443\Lab12\binshAddress.PNG

The code for my program to see what is really at the addresses:



After running this for both addresses:



Seeing that they have the same string at each address, I’m not sure why it didn’t work for the environment variable. However, the moral of the story is that it did work by searching through lib-c! A screenshot of the shell is below:

