# Lab2: SEED Labs – Buffer Overflow Attack Lab

Mayur Suresh

CSC 302-01: Computer Security

Date: February 21st, 2023

## Intro:

In this lab we will observe the Buffer overflow attack. Where when a program has reached its capacity to maintain data in the assigned stack memory address and creates a ‘overflow’ we attempt to use the vulnerability caused when the program overflows into the next memory address. We can use this vulnerability to control the flow of the program by altering its path to a malicious code upon execution.

In order to start this buffer overflow attack, we need to strip several security implications in place. The security mechanism that needs be striped off is the ‘Address Space Randomization’ mechanism which Linux operated systems use to randomize the starting addressees of the heap and stack of a program. This makes it difficult to obtain the exact addressed which is one of the most important steps of the buffer overflow attack. We use the command below. We also use another command to link the `/bin/sh` to the z shell or the `zsh` in the Set-UID program.

Text, timeline

Description automatically generated

## Task 1: Getting Familiar with Shellcode

In this task we will look at the Shellcode which will launch the shell and we need to get used to this environment as it will be used to launch the buffer overflow attack. But since we cannot just compile and run the C code and use the binary file to run the shell we will need to use assembly code to launch the shell.

The shell code written in assembly first invokes the execve() function when the register esp is moved to ebx which executes the `/bin/sh` command. We now invoke the shellcode. We have a c program called `call\_shellcode.c` provided which will test the shellcode as shown below.

Text

Description automatically generated

We now run the build file `Makefile` which builds and compile the programs with the necessary security tags striped. By running this file we see 2 binary files creates, a 32 bit and a 64 bit files.



When we run both binary files and get into the shell we see that both the 32 bit and 64 bit versions behave the same. Both where able to access the directories and the addresses.

Text

Description automatically generated

## Task 2: Understanding the Vulnerable Program