Year and Semester	2018 FALL
Course Number	CS-336
Course Title	Intro. to Information Assurance
Work Number	LA-01
Work Name	Buffer Overflows Classic
Work Version	Version 1
Long Date	Wednesday, 24 October 2018
Author(s) Name(s)	Zane Durkin

Abstract

In this article I will be explaining in detail the Tasks I preformed during the SEED security lab.

1 Setting up Lab Environment

In order to preform the classic stack overflow attack, some of the countermeasures will need to be disabled. These countermeasures (listed in sub sections below) are enabled in the Ubuntu operating system, and many others, by default. These security countermeasures are in place to make stack overflow attacks difficult [1].

1.1 Address Space Randomization

Address space Randomization is used to change the starting address of the heap and stack for every run of the program. This security mechanism makes guessing the address of your overflow attack difficult and unpredictable. To make this lab easier, I will disable this using the following command [1].

```
sudo -w kernel.randomize_va_space=0
```

This command will remove the randomization of the starting address for the heap and stack. This makes it much easier to figure out the address I need to use for my exploit, since it will be the same address every time I run my code.

1.2 Non-Executable Stack

By default gcc disables execution of code on the stack. However, since I will be placing my exploit code on the stack, I will need to be able to execute code on the stack for the exploit to work. So to allow for execution of things on the stack I will need to add a flag to gcc when compiling my vulnerable code [1]. To Execute code on the stack, my compile command will look like this:

```
gcc -z execstack -o test test.c
```

And for non-executable stack I can use the following command (although gcc does this by default):

```
# comment
gcc -z nonexecstack -o test test.c
```

References

[1] Wenliang Du, S. U. Buffer overflow vulnerability lab, 2016.