**Buffer Overflow Coding**

Justin Starr

Department of STEM

CS 405 – Secure Coding

Professor Mike Alesso

July 14, 2024

**Buffer Overflow Coding**

**Screenshots**

Successful execution on first attempt:

**A screenshot of a computer

Description automatically generated**

One failed attempt with second attempt successful:

**A screenshot of a computer

Description automatically generated**

Program terminates after three failed attempts:

A screenshot of a computer

Description automatically generated

Program running script with exploit from command:

A screenshot of a computer

Description automatically generated

Program running script without exploit from command:

A screenshot of a computer program

Description automatically generated

**Process Summary**

The way I decided to approach this problem was to first allow the user to enter a value and place that value into a string called test\_input. I then used a while loop to compare the length of the test\_input variable to the size of the user\_input variable to ensure that the user does not input a value that would cause a buffer overflow. A buffer overflow would occur if the length of test\_input is greater than or equal to the size of the user\_input variable, therefore inside the while loop, a message is displayed to the user that a buffer overflow is detected. The while loop affords the user the opportunity to make an attempt at entering a valid value three times. If within these three attempts the user does not enter a valid value, meaning they enter a value that will cause a buffer overflow, the program will terminate execution, preventing the attempted buffer overflow.

When the user inputs an acceptable value, the while loop is exited and the test\_input string is then copied to the user\_input variable. Lastly, the user’s input is displayed back to the user as well as the unmodified account\_number variable. The while loop is essentially what detects when an overflow is about to happen and prevents it from happening. The while loop is also how the program will react when the user attempts a buffer overflow, and it is what protects the code from a buffer overflow.

The main issue I encountered while writing my code was deciding how I wanted the program to react when the user attempted a buffer overflow. I wasn’t sure if the program should simply terminate on a single attempt or if the program should have a loop to allow the user multiple attempts for inputting a value. At first, I implemented a continuous loop, however, I decided while running the PythonWithExploit.py file as the input for the program from a shell, this would cause the program to run in a continuous loop that had to be aborted manually in order to terminate the program’s execution. I realized this was itself a vulnerability because it consumes system resources and could be exploited to cause a DoS (denial of service). Therefore, I decided to modify my program to only allow a user three attempts to enter valid input. This ensures that when the program is run in conjunction with a script from a shell, the program will always terminate, even if there is an attempted buffer overflow from the script.