2-3 Activity – Buffer Overflow Coding

Joel Meza

Professor Trevor Hodde

January 14, 2022

CS 405 – Secure Coding

Southern New Hampshire University

**Overview**

As I approached this assignment, I had some different ideas to come up with a solution for a new or current user from the bank to input their account credentials into the program securely without any cause of overflow. Therefore, I started to analysis the code provided already and it seems that the developers were using char user\_input[20] as the prompt data type to convert it into a proper count range of characters from 0 through 19, where it enables the proximity range of the allowed characters from the user’s input. Next, I took that into consideration and made sure what was going on in std::cout << “Enter a value: ” <<endl; would be brought into the std::cin.getline(). By using std::cin.getline() method, this method operates as a transfer functionality which both parameters are placed inside as (user\_input, 20) where it makes the user\_input value to be first converted as previously mentioned. Then it would be brought to the second parameter/argument of “20” to be constrained within the range of 0 through 19 characters, so it may not exploit and be at risk of a buffer overflow. Lastly, I went on with what was provided in the code that the std::cout<< “You Entered: ” << user\_input<<endl; … to generate what was imported first into the program and the std::cout<< “Account Number: ” << account\_number << endl; … access is granted and the corresponding result (account number) is displayed.

Graphical user interface, text

Description automatically generatedText

Description automatically generated