**Victor Udeh  
CS405 M2-3  
Date:   
  
  
Buffer Overflow Prevention Summary**

In this activity, I was tasked with resolving a buffer overflow vulnerability in a C++ application. The vulnerability allowed a user to input more than 20 characters, which could potentially overwrite the constant account number. The code initially allowed unchecked input, leading to possible unintended behavior.

**Approach Taken**

1. **Input Limitation**: I utilized the std::cin.width() function to limit the input length to the size of the user\_inputbuffer. This ensured that no more than 19 characters could be entered by the user, preserving space for the null terminator.
2. **Error Handling**: After inputting the value, I added a check to determine if the number of characters entered exceeded the buffer limit. If the input exceeded the allowed size, an error message was displayed, and the program exited gracefully.
3. **Code Modifications**: The modified code prevents the buffer overflow by explicitly limiting the number of characters that can be entered into user\_input. I added appropriate comments explaining the changes, such as the use of std::cin.width() and the error-checking condition.

**Issues Encountered and Solutions**

* **Handling Input Size**: To ensure that the input length did not exceed the buffer, I needed to understand how to properly use the width() function with cin and correctly handle the edge cases.
* **Truncation vs. Notification**: During testing, I found that inputs longer than 19 characters were truncated rather than causing an immediate error. I adjusted the error-checking logic to handle these situations appropriately.

**Testing**

* **Successful Test**: Inputs under 20 characters, such as "Hello123" and "VictorUdeh", were accepted and displayed correctly without modifying the constant account\_number.
* **Overflow Prevention Test**: An input of more than 19 characters, such as "ABCDEFGHIJKLMNOPQRSTUVWXYZ", resulted in truncation to "ABCDEFGHIJKLMNOPQRS", demonstrating the prevention of the overflow without affecting the account number.

Below is a screenshot of the console output demonstrating both successful and overflow-prevention scenarios:

**Conclusion**

This solution effectively mitigates the buffer overflow vulnerability by ensuring that user input cannot exceed the buffer's capacity. The use of standard library functions such as width() and proper error handling allows the program to manage user inputs safely while preserving the integrity of other variables in memory.

A screenshot of a computer program

Description automatically generated