CS 405 Module Two Buffer Overflow Activity Guidelines and Rubric

Overview

You are a senior software developer on a team of software developers who are responsible for a large banking web application. Your manager has learned that there is some code that seems to randomly change customer account numbers. The other developers spent a lot of time tracing down the issue, and they narrowed it down to when a user inputs a number. Under some circumstances, this resulted in changing the account number. The team has tried a few things, but they are stumped.

Following some guidance from an older testing guide, the developers isolated enough to replicate the problem in a standalone test case. Doing this allowed them to reliably show how to trigger the buffer overflow. However, they don't know what to do to prevent it in the future. They came to you for help.

You will learn to do the following:

- Detect when a buffer overflow is about to happen
- Write code to prevent a buffer overflow
- Decide how to react when the user attempts a buffer overflow
- Protect your code from a buffer overflow

Prompt

This assignment presents code that collects a number from the user and then displays the number and a secret account number. Entering too many characters triggers the overwrite of the account number. Using the existing source code as a starting point, you need to prevent the user input from impacting the account number.

- The source code has been commented using TODO to explain the detailed rules you must follow.
- There may be more than one way to solve this problem. It is key to demonstrate that your implementation prevents the users from modifying the account number through entering too much data.
- Do not forget that you can leverage capabilities provided by the standard C++ library to help you achieve success.

Please comment on any changes you make in the code to explain the logic, formulas, or data types you are adding. You will also create a brief written summary of the approach taken, why it will stop the overflow, any issues you encountered, and how you resolved those issues.

Specifically, you will prepare the following:

- Buffer overflow defensive coding solutions
- C/C++ program functionality and best practices
- A summary of your process in a Word document that contains a screenshot of the application console output

To complete this assignment, download the Buffer Overflow source code files ZIP to use as guidance as you move through the activity. You will use your development environment to complete this activity.

What to Submit

Submit the completed source code as a ZIP file. Also include a Word document that contains a complete screenshot of the application console output and a brief summary of your process.

Module Two Buffer Overflow Activity Rubric

Criteria	Proficient (100%)	Needs Improvement (75%)	Not Evident (0%)	Value
Buffer Overflow Coding	Completes the coding activity by successfully preventing buffer overflow for all data types; notifies the calling function; displays the buffer overflow condition to the user	Completes most of the coding activity, but needs to address one or more of the following: does not prevent both buffer overflow; does not work for all data types; prevents the overflow, but does not notify the calling function; does not display the result of the overflow to the console	Does not attempt criterion	70
C/C++ Program Functionality and Best Practices	Demonstrates industry standard best practices, including in-line comments and appropriate naming conventions to	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include implementation	Does not attempt criterion	10

	enhance readability of code; develops functional C/C++ code that illustrates software design pattern approach	of software design patterns in code		
Process Summary	Provides a summary of the debugging that is thorough and systematic, including specific types of bugs; accurately describes the corrections and provides screenshots of the application console output, from the compiler, and of the error list	Provides a summary of the debugging, but the summary lacks detail, the approach is unsystematic, specific types of bugs are not differentiated, the corrections are not fully described, or a screenshot is missing	Does not attempt criterion	20
			Total:	100%