CS 405 Module Two SQL Injection 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 has been a number of unauthorized uses of one of the company's systems. The other developers spent a lot of time tracing the issue, and they narrowed it down to some SQL queries running in the database that were modified from the queries created by the application. It allowed a hacker to get more data back than they were entitled to, including passwords. The other developers have tried a few things, but they don't know how to prevent it.

Following some guidance from an older testing guide, the developers were able to replicate the problem in a standalone test case including a standalone in-memory SQLite database. Doing this allowed them to reliably reproduce the SQL injection attack. However, they don't know what to do next to prevent it from happening in the future. They came to you for help.

You will learn to do the following:

- Anticipate when a possible SQL injection attack is about to occur
- Write code to prevent a suspected SQL injection attack
- Decide how to react when there is an attempted SQL injection attack
- Protect your code from SQL injection attacks

Prompt

This assignment presents code that creates an in-memory database and runs a number of queries with and without SQL injection, always dumping the results to the console. You need to modify the run_query() function to detect a potential SQL injection attack, prevent it, and display that information to the console. The specific type of SQL injection attack is the "OR value=value;" attack, and you must defend against that specific attack with value being any allowed SQL literal (numeric or quoted string).

The following are a few key notes:

- 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 SQL injection attack and displays the information to the console.
- 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 SQL Injection Attack, any issues you encountered, and how you resolved those issues.

Specifically, you will prepare the following:

- SQL injection 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 SQL Injection 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 SQL Injection Activity Rubric

Criteria	Proficient (100%)	Needs Improvement (75%)	Not Evident (0%)	Value
SQL Injection	Completes the coding activity by successfully preventing SQL injection and displaying to the console about the prevention	Completes most of the coding activity, but needs to address one or more of the following: does not prevent SQL injection; does not handle the broader case of `value=value`; does not display the results of the SQL injection mitigation 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 enhance readability of code; develops functional C/C++ code that illustrates software design pattern approach	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include implementation of software design patterns in code	Does not attempt criterion	10
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%