Laynie Tierney

CS-405-R4888 Secure Coding

20 March 2024

2-2 Activity: SQL Injection Coding

* In this code, I have an example of an in-memory database system with functions to initialize the database, run queries, and prevent SQL injection attacks. The main focus is on the **run\_query()** function, which executes SQL queries, and the **run\_query\_injection()** function, which attempts to inject SQL code to simulate an SQL injection attack.
* To prevent SQL injection attacks, I need to detect and sanitize user input to ensure that it does not contain any malicious SQL code. Specifically, I need to prevent queries like "OR value=value;", which is a common SQL injection attack pattern.
* To achieve this, I added the **run\_query\_injection()** function, which generates a potentially injected SQL query by appending "or 1=1;", "or 2=2;", "or 'hi'='hi';", or "or 'hack'='hack';" to the original query. This function then calls the **run\_query()** function with the potentially injected SQL query.
* In the **run\_query()** function, I clear any prior results and execute the SQL query. If the query execution fails, I display an error message indicating that the data failed to be queried from the USERS table.
* During testing, I run the original query and the injected query multiple times to ensure that the prevention mechanism is effective.
* Overall, by implementing the **run\_query\_injection()** function and ensuring that user input is properly sanitized, I can prevent SQL injection attacks and protect the integrity of the database system.

A screenshot of a computer program

Description automatically generated