**405 Module Two SQL Injection Activity Summary**

This project is about protecting a C++ application from SQL injection attacks. SQL injection happens when someone tries to sneak malicious code into your SQL queries, usually to steal data or mess with your system. The project simulated a basic SQL injection attack and prevented it using some defensive coding techniques. The focus is on stopping the "OR value=value" attack, where attackers try to manipulate the query by adding something like "OR 1=1" that always returns true. By preventing these types of attacks, making sure unauthorized users can’t access sensitive data is the goal.

**Code Changes**

The main modification in the code was in the run\_query() function, where I added a regular expression (regex) to detect SQL injection attempts. A regex allows us to identify patterns in the SQL query that look suspicious. If the pattern matches, the query is blocked from running, and the program prints a message saying the query was blocked.

Here’s the important part of the code that was added:

// SQL Injection Prevention

// Using regex to detect patterns like "OR 1=1", "OR 'a'='a'", etc.

// Regular expression for basic SQL injection detection: "OR something=something"

std::regex injection\_regex(R"(or\s+\S+\s\*=\s\*\S+)", std::regex\_constants::icase);

// Check if the SQL query contains an injection pattern

if (std::regex\_search(sql, injection\_regex))

{

std::cout << "SQL Injection detected! Query blocked: " << sql << std::endl;

return false; // Prevent running the query

}

**Explanation of Changes:**

* The regular expression (std::regex) searches for SQL injection patterns like "OR 1=1" or "OR 'x'='x'".
* If such a pattern is found, the program stops the query from running and prints a warning in the console.
* This prevents malicious queries from affecting the database and accessing unauthorized data.

**Console Output Screenshot**

**A screenshot of a computer

Description automatically generated**

This screenshot shows the output after running the SQL Injection prevention code.

1. The program connects to the database and successfully creates the **USERS** table.
2. The first query (SELECT \* from USERS) retrieves all user records, showing Fred, Barney, Wilma, and Betty.
3. The second query (SELECT ID, NAME, PASSWORD FROM USERS WHERE NAME='Fred') successfully finds Fred's record.
4. After that, SQL injection attempts are blocked, as seen by the message **"SQL Injection detected! Query blocked"** for queries like SELECT ID, NAME, PASSWORD FROM USERS WHERE NAME='Fred' or 2=2;
5. The program exits with code 0, meaning everything ran smoothly.

**Successful Build Screenshot**

A screenshot of a computer program

Description automatically generated

**References**

Seacord, Robert C. *Secure Coding in C and C++*. Pearson Technology Group, 2013