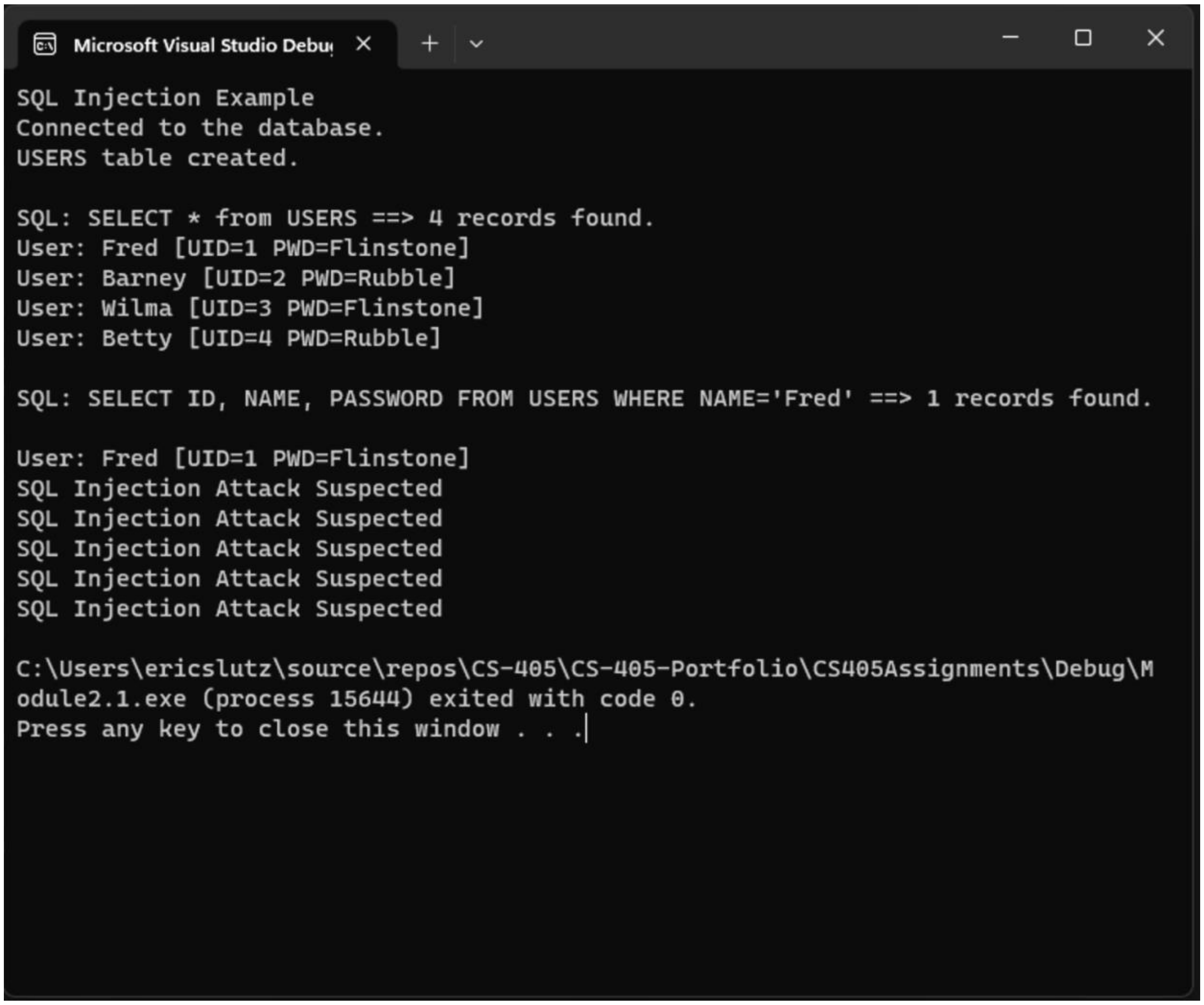
CS-405 Secure Coding – Module 2 Assignment 1

Eric Slutz

Southern New Hampshire University

Module Two Assignment One

SQL injection tests screenshot



```
SQL Injection Example
Connected to the database.
USERS table created.

SQL: SELECT * from USERS ==> 4 records found.
User: Fred [UID=1 PWD=Flinstone]
User: Barney [UID=2 PWD=Rubble]
User: Wilma [UID=3 PWD=Flinstone]
User: Betty [UID=4 PWD=Rubble]

SQL: SELECT ID, NAME, PASSWORD FROM USERS WHERE NAME='Fred' ==> 1 records found.

User: Fred [UID=1 PWD=Flinstone]
SQL Injection Attack Suspected
SQL Injection Attack Suspected
SQL Injection Attack Suspected
SQL Injection Attack Suspected
SQL Injection Attack Suspected

C:\Users\ericslutz\source\repos\CS-405\CS-405-Portfolio\CS405Assignments\Debug\M
odule2.1.exe (process 15644) exited with code 0.
Press any key to close this window . . .
```

Summary of SQL injection detection process

For this assignment, there was only one type of SQL injection that needed to be protected against, an “OR value=value;” attack. Due to the method parameter only taking in the entire SQL query string and not being able to modify any of the calling code, I could not update the method to use parameterized input. Instead, I had to find a way to take the full SQL query and check for this specific type of attack without hardcoding any values and without preventing different types of queries from running. Since this attack has a specific pattern, I was able to create a regular expression that looked for an "or" statement followed by either two matching numeric or character/string values. One edge case I tested was two strings where the casing of the letters didn't match. The regex did not recognize these values as matching, but as a SQL query it was valid, causing the SQL injection to succeed. After spending some time refining the regex pattern, I was able to get it to detect all the SQL injection attempts during testing.