SQL Injection Attack Written Report

Jeet Kumar

University Of Maryland: Baltimore County

Author Notes:

User input will only work if example code is run first to initialize the database, and then the vulnerable or protected program is run.

“input” is the designated user input for all of the following eleven data points after last\_name since they are, by default, not shown in the shell.

# Vulnerable Program

The first program is designed to be a program that is susceptible to a SQL injection attack. To do that I have to create a user interface which would allow a user to add a record to the UMBC database. In order to get user input, I have implemented source code from an outside source [[1]](#footnote-1) which asks the user for the ID number, first name and last name and autocompletes the record process so that I can save time in between code executions. This program is unprotected because the code is made so that any value can be entered into the database. User input can be run as an SQL statement which can be executed as a command. Another way a hacker can gain access to SQL databases, is by requesting data that has been inserted as a string concatenation. If the SQL statement is written without the correct parameters, the database could be prone to malicious entries. These are both examples of how liabilities can be exposed in the program. A vulnerable program can allow the user can delete records, alter data, and access the entire database if user input is not validated. This makes data retrieval or insertion for external, verified users unmanageable.

# Protected Program

The protected program has been altered to accept user input as a simple database entry and it also disregards any entry that does not qualify. The program works because the parameters are executed as a part the data in the database and not as part of the code that is associated with maintaining the database. [[2]](#footnote-2) Each value is initialized as an unknown value which makes it hard for hackers to retrieve any inserted data. I have also accepted first\_name and last\_name as raw input which allows the program to accept user input without them being able to affect any data in the database and it makes it much more difficult to alter any data. These two precautionary coding techniques are great ways to combat SQL injection attacks.

# Conclusion

In conclusion, if a program is susceptible to a SQL injection attack, then it is most likely very poorly written and managed. Since injection attacks can be used to extract sensitive information like credit card credentials and social security numbers, it is important that programmers understand the ways to combat the dangerous capabilities of hackers in today’s workplace environment.

**References**

1. SheldonSheldon 3, & PearsonArtPhoto. (2012, November 28). Is this Python code vulnerable to SQL injection? (SQLite3). Retrieved from <https://stackoverflow.com/questions/13613037/is-this-python-code-vulnerable-to-sql-injection-sqlite3>
2. W3Schools. (n.d.). SQL Injection. Retrieved from <https://www.w3schools.com/sql/sql_injection.asp>
3. Mike, K., & J.H. (2019, February 27). Sqlite3 python3, user input for database. Retrieved from <https://stackoverflow.com/questions/54897260/sqlite3-python3-user-input-for-database>

1. Mike, K., & J.H. Sqlite3 python3, user input for database [↑](#footnote-ref-1)
2. SheldonSheldon 3, & PearsonArtPhoto. Is this Python code vulnerable to SQL injection? (SQLite3) [↑](#footnote-ref-2)