**SQL Injection: Persistence Development Mitigation SOP**

Queries to the database from the application can be altered by different parameters. When the query has been changed beyond the scope of what the parameters were to be expected it is considered to be a SQL Injection. SQL Injection gives the ability to the hacker to change the context of the query. Typically, SQL injections occur when parameters are populated from untrusted sources.

**Defense Against SQL Injection: Persistence**

During development, parameterized SQL statement should be used instead of user-supplied values especially if the parameters can come from an untrusted source. This limits what values of the parameters of the query. It's absolutely vital to sanitize user inputs to insure that they do not contain dangerous codes, whether to the SQL server or to HTML itself.

**Example**

String userName = ctx.getAuthenticatedUserName();

String itemName = request.getParameter("itemName");

String query = "SELECT \* FROM items WHERE owner = '"

            + userName + "' AND itemname = '"

            + itemName + "'";

ResultSet rs = stmt.execute(query);

**Explanation**

The value of itemName can be change the context of the query. If the value of itemName is “name' OR 'a'='a”, then the query will return all of the records regardless of the itemName.

**Recommendation**

To prevent an attack from having the ability of changing the context of the SQL query, the variable itemName is injected into the query programmatically. Using the setString method to enforce strong type checking, since it will properly escape the input by automatic entrapment within double quotes.

String userName = ctx.getAuthenticatedUserName();

String itemName = request.getParameter("itemName");

String query =

         "SELECT \* FROM items WHERE itemname=? AND owner=?";

PreparedStatement stmt = conn.prepareStatement(query);

stmt.setString(1, itemName);

stmt.setString(2, userName);

ResultSet results = stmt.execute();

**References**

1. [OWASP – Information Leakage](https://www.owasp.org/index.php/Top_10_2007-Information_Leakage_and_Improper_Error_Handling)