SQL Injection Attacks

Murat Kantarcioglu
Overview

- Sql injection attacks are one of the top attacks against web based applications.
- **Example:** License plate recognition system
Example Attack in ASP

```vbs
set ok = execute( "SELECT * FROM Users
    WHERE user=' " & form("user") & 
    " 'AND pwd=' " & form("pwd") & " '"
);

if not ok.EOF
    login success
else fail;

Is this exploitable?
```
Normal Query

Web Browser (Client) → Web Server
Enter Username & Password

Web Server → DB
SELECT * FROM Users WHERE user='me' AND pwd='1234'

Normal Query
Suppose user = "' or 1=1 - -" (URL encoded)
Then scripts does:

```
ok = execute(SELECT ... WHERE user= ' ' or 1=1 - - ... )
```
- The "- -" causes rest of line to be ignored.
- Now ok.EOF is always false and login succeeds.

The bad news: easy login to many sites this way.
Examples in Java:

- String pw = "123456";
  
  // this would come from the user
  String query = "SELECT * from users where name = 'USER' " + "and password = " + pw + "";

  stmt = conn.createStatement();
  rs = stmt.executeQuery(query);
Solution: my favorite one 😊

- Never create query by combining strings coming from the user.
- Instead use Prepared statements.
- Other options such as sanitization could be considered if prepared statements do not work.
Java Prepared Statement Example

```java
public void updateCoffeeSales(HashMap<String, Integer> salesForWeek) throws SQLException {
    String updateString =
        "update COFFEEES set SALES = ? where COF_NAME = ?";
    String updateStatement =
        "update COFFEEES set TOTAL = TOTAL + ? where COF_NAME = ?";

    try (PreparedStatement updateSales = con.prepareStatement(updateString);
         PreparedStatement updateTotal = con.prepareStatement(updateStatement))
    {
        con.setAutoCommit(false);
        for (Map.Entry<String, Integer> e : salesForWeek.entrySet()) {
            updateSales.setInt(1, e.getValue().intValue());
            updateSales.setString(2, e.getKey());
            updateSales.executeUpdate();

            updateTotal.setInt(1, e.getValue().intValue());
            updateTotal.setString(2, e.getKey());
            updateTotal.executeUpdate();
            con.commit();
        }
    } catch (SQLException e) {
        JDBCUtilities.printSQLException(e);
        if (con != null) {
            try {
                System.err.print("Transaction is being rolled back");
                con.rollback();
            } catch (SQLException excep) {
                JDBCUtilities.printSQLException(excep);
            }
        }
    }
}
```