SQL Injection

SQL injection is a technique where malicious users can inject SQL commands into an SQL statement, via web page input.

Injected SQL commands can alter SQL statement and compromise the security of a web application.

SQL Injection Based on 1=1 is Always True

Look at the example above, one more time.

Let's say that the original purpose of the code was to create an SQL statement to select a user with a given user id.

If there is nothing to prevent a user from entering "wrong" input, the user can enter some "smart" input like this:

UserId:   


Server Result

SELECT \* FROM Users WHERE UserId = 105 or 1=1

The SQL above is valid. It will return all rows from the table Users, since **WHERE 1=1** is always true.

Does the example above seem dangerous? What if the Users table contains names and passwords?

The SQL statement above is much the same as this:

SELECT UserId, Name, Password FROM Users WHERE UserId = 105 or 1=1

A smart hacker might get access to all the user names and passwords in a database by simply inserting 105 or 1=1 into the input box.

SQL Injection Based on ""="" is Always True

Here is a common construction, used to verify user login to a web site:

User Name:  


Password:  


Server Code

uName = getRequestString("UserName");  
uPass = getRequestString("UserPass");  
  
sql = "SELECT \* FROM Users WHERE Name ='" + uName + "' AND Pass ='" + uPass + "'"

A smart hacker might get access to user names and passwords in a database by simply inserting " or ""=" into the user name or password text box.

The code at the server will create a valid SQL statement like this:

Result

SELECT \* FROM Users WHERE Name ="" or ""="" AND Pass ="" or ""=""

The result SQL is valid. It will return all rows from the table Users, since **WHERE ""=""** is always true.

SQL Injection Based on Batched SQL Statements

Most databases support batched SQL statement, separated by semicolon.

Example

SELECT \* FROM Users; DROP TABLE Suppliers

The SQL above will return all rows in the Users table, and then delete the table called Suppliers.

If we had the following server code:

Server Code

txtUserId = getRequestString("UserId");  
txtSQL = "SELECT \* FROM Users WHERE UserId = " + txtUserId;

And the following input:

User id:  


The code at the server would create a valid SQL statement like this:

Result

SELECT \* FROM Users WHERE UserId = 105; DROP TABLE Suppliers