

SQL Injection

[< Previous](#)[Next >](#)

SQL Injection

SQL injection is a code injection technique that might destroy your database.

SQL injection is one of the most common web hacking techniques.

SQL injection is the placement of malicious code in SQL statements, via web page input.

SQL in Web Pages

SQL injection usually occurs when you ask a user for input, like their username/userid, and instead of a name/id, the user gives you an SQL statement that you will **unknowingly** run on your database.

Look at the following example which creates a SELECT statement by adding a variable (txtUserId) to a select string. The variable is fetched from user input (getRequestString):

Example

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

COLOR
PICKER



HOW
TO

The rest of this chapter describes the potential dangers of using user input in SQL statements.

SQL Injection Based on 1=1 is Always True

Look at the example above again. 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:

Then, the SQL statement will look like this:

```
SELECT * FROM Users WHERE UserId = 105  
OR 1=1;
```

The SQL above is valid and will return ALL rows from the "Users" table, since **OR 1=1** is always TRUE.

Does the example above look 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 hacker might get access to all the user names

Tabs
Dropdown
Accordion
Side
Navigation
Top
Navigation
Modal
Boxes
Progress
Bars
Parallax
Login
Form
HTML
Includes
Google
Maps
Range
Sliders
Tooltips
Slideshow
Filter
List
Sort
List

SHARE



CERTIFIC

HTML
CSS
JavaScript
SQL
Python
PHP

and passwords in a database, by simply inserting 105 OR 1=1 into the input field.

jQuery
Bootstrap
XML

[Read More](#)

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

Here is an example of a user login on a web site:

Username:

Password:

Example

```
uName = getQueryString("username");  
uPass =  
getQueryString("userpassword");  
  
sql = 'SELECT * FROM Users WHERE Name  
=' + uName + ' ' AND Pass =' + uPass +  
' '
```

Result

```
SELECT * FROM Users WHERE Name ="John  
Doe" AND Pass ="myPass"
```

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

User Name:

Password:

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

Result

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

The SQL above is valid and will return all rows from the "Users" table, since **OR ""=""** is always TRUE.

SQL Injection Based on Batched SQL Statements

Most databases support batched SQL statement.

A batch of SQL statements is a group of two or more SQL statements, separated by semicolons.

The SQL statement below will return all rows from the "Users" table, then delete the "Suppliers" table.

Example

```
SELECT * FROM Users; DROP TABLE  
Suppliers
```

Look at the following example:

Example

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

And the following input:

User id:

The valid SQL statement would look like this:

Result

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

Use SQL Parameters for Protection

[HTML](#)[CSS](#)[JAVASCRIPT](#)[SQL](#)[PYTHON](#)[PHP](#)[BOOTSTRAP](#)

use SQL parameters.

SQL parameters are values that are added to an SQL query at execution time, in a controlled manner.

ASP.NET Razor Example

```
txtUserId = getRequestString("UserId");  
txtSQL = "SELECT * FROM Users WHERE  
UserId = @0";  
db.Execute(txtSQL, txtUserId);
```

SQL Null Functions
SQL Stored Procedures
SQL Comments

SQL Database

SQL Create DB
SQL Drop DB
SQL Backup DB
SQL Create Table
SQL Drop Table
SQL Alter Table
SQL Constraints
SQL Not Null
SQL Unique
SQL Primary Key
SQL Foreign Key
SQL Check
SQL Default
SQL Index
SQL Auto Increment
SQL Dates
SQL Views
SQL Injection
SQL Hosting

SQL References

SQL Keywords
MySQL Functions
SQL Server Functions
MS Access Functions
SQL Operators
SQL Data Types
SQL Quick Ref

SQL Examples

SQL Examples
SQL Quiz
SQL Exercises

Note that parameters are represented in the SQL statement by a @ marker.

The SQL engine checks each parameter to ensure that it is correct for its column and are treated literally, and not as part of the SQL to be executed.

Another Example

```
txtNam =  
getRequestString("CustomerName");  
txtAdd = getRequestString("Address");  
txtCit = getRequestString("City");  
txtSQL = "INSERT INTO Customers  
(CustomerName,Address,City)  
Values (@0,@1,@2)";  
db.Execute(txtSQL,txtNam,txtAdd,txtCit);
```

Examples

The following examples shows how to build parameterized queries in some common web languages.

SELECT STATEMENT IN ASP.NET:

```
txtUserId = getRequestString("UserId");  
sql = "SELECT * FROM Customers WHERE  
CustomerId = @0";  
command = new SqlCommand(sql);  
command.Parameters.AddWithValue("@0",txt  
UserID);  
command.ExecuteReader();
```

INSERT INTO STATEMENT IN ASP.NET:

```
txtNam =  
getRequestString("CustomerName");  
txtAdd = getRequestString("Address");  
txtCit = getRequestString("City");  
txtSQL = "INSERT INTO Customers  
(CustomerName,Address,City)  
Values (@0,@1,@2)";  
command = new SqlCommand(txtSQL);  
command.Parameters.AddWithValue("@0",txt  
Nam);  
command.Parameters.AddWithValue("@1",txt  
Add);  
command.Parameters.AddWithValue("@2",txt  
Cit);  
command.ExecuteNonQuery();
```

INSERT INTO STATEMENT IN PHP:

```
$stmt = $dbh->prepare("INSERT INTO  
Customers (CustomerName,Address,City)  
VALUES (:nam, :add, :cit)");  
$stmt->bindParam(':nam', $txtNam);  
$stmt->bindParam(':add', $txtAdd);  
$stmt->bindParam(':cit', $txtCit);  
$stmt->execute();
```

[< Previous](#)[Next >](#)[REPORT ERROR](#)[PRINT PAGE](#)[FORUM](#)[ABOUT](#)[Top Tutorials](#)[Top](#)[Top](#)[Web](#)

	References	Examples	Certificates
HTML Tutorial			
CSS Tutorial			
JavaScript Tutorial	HTML Reference	HTML Examples	HTML Certificate
How To Tutorial	CSS Reference	CSS Examples	CSS Certificate
SQL Tutorial	JavaScript Reference	JavaScript Examples	JavaScript Certificate
Python Tutorial	SQL Reference	How To Examples	SQL Certificate
W3.CSS Tutorial	Python Reference	SQL Examples	Python Certificate
Bootstrap Tutorial	W3.CSS Reference	Python Examples	jQuery Certificate
PHP Tutorial	Bootstrap Reference	W3.CSS Examples	PHP Certificate
jQuery Tutorial	PHP Reference	Bootstrap Examples	Bootstrap Certificate
Java Tutorial	HTML Colors	PHP Examples	XML Certificate
C++ Tutorial	jQuery Reference	jQuery Examples	
	Java Reference	Java Examples	
	Angular Reference	XML Examples	
			Get Certified »

W3Schools is optimized for learning, testing, and training. Examples might be simplified to improve reading and basic understanding. Tutorials, references, and examples are constantly reviewed to avoid errors, but we cannot warrant full correctness of all content. While using this site, you agree to have read and accepted our terms of use, cookie and privacy policy. Copyright 1999-2019 by Refsnes Data. All Rights Reserved.

Powered by W3.CSS.

