# SQL INJECTIONS – THE BASICS

## INTRO

Today's websites become more and more advanced and a common service that they use is **SQL-server**. But when you implement a **SQL server** and start to write code that inserts and selects data from the **SQL** database you must be sure that nobody can send a malformed request that will result in that sensitive information such as users passwords and emails gets into wrong hands. It's also common that credit cards gets stolen using ***SQL Injections***.

If you are going to write secure code you must know what to expect from the users. A good rule to think about when coding a website is: *"never trust the user"* since they can malform any data sent from their computer.

To make it easier for you to read this article I suggest that you read up on SQL basics at: ***http://www.w3schools.com/sql/default.asp***

## HOW TO FIND A VULNERABILITY

Online stores are a huge target for today's hackers. Not only cause many stores store customers credit card information, no, hackers can also use customers login to buy things and let the customer pay for something to hacker bought. This is a serious issue. But how do you prevent this from happening? Well you must know what to look for.

An example of a vulnerable website could look like this: "***www.explample.com/product.php?id=5***". And why is this vulnerable? I will explain; when you visit this site it will ask the **SQL-server** a question, it will send a ***query*** to the server that it may want price, name and description about a product with the **ID 5**. And that query could look like this:

```
SELECT price,name,description FROM products WHERE id='5'
```

As long as the user don't manipulate this it's going to be fine. But what happens if the user want to see a product with the **id 5'**. Well the query would look like this:

```
SELECT price,name,description FROM products WHERE id='5''
```

Notice that the id now looks like this: *'5"* instead of *'5'*. This will, if vulnerable, return a syntax error or a custom error-message written by the systemadmin.

## WHAT CAN YOU DO WITH THIS?

If a error should appear then we know that we are able to manipulate the query using **UNION**. What we want is to display information that may be sensitive so lets say that there is a table that named "**customers**" and we want the content in the columns username, password and email. Then we create our own query that may look like this:

```
SELECT user,password,email FROM customers--
```

And if we combine this query with the other one we get:

```
SELECT price,name,description FROM products WHERE id='5' UNION SELECT user,password,email
FROM customers-
```

This will result in that the customers username, password and emails are printed on the page.

## HOW CAN YOU PREVENT THIS?

When you are writing code, let's say **PHP** it's very very important that you never trust the users. You have to make sure that you expect input that can look like anything and have protocols that takes care of unwanted input.
A great way to prevent attacks such as the one above is to use the function intval(..) in the PHP-language which will extract everything from a variable that is an integer.

Another great thing is to limit how many characters in length that you allow and thereby prevent long querys like the one used to extract user-info.

## SUMMARY

This is an article that summarizes the basics in SQL Injections. There is much more to it than this like extract table names and column names, get the exact information from the tables, shutdown the server and many other things. So this is basically a wakeup-call for you guys that is working with programming in PHP, ASP and many other languages related to the web.