CSE-545 Software Security

SQL INJECTION PREVENTION PROJECT REPORT

Submitted by

NIKHIL MEKA, - nmeka@asu.edu- 1209311086
NIKHIL VEMENTALA – nvementa @asu.edu- 1209370509

Table of Contents

1. Introduction:

In this project, we developed a library to prevent SQL-Injection attacks. This library provides a layer for the application programmer by providing a set of functions to make sure that the query passed to the database cannot be changed by the user input.

2. Approach for Project:

Sanitizing inputs that pass onto the query will prevent first –order SQL injection attacks but do not prevent Second-Order attacks. Hence to prevent n-order attacks, the user-input should always be sanitized while sending data to the database and retrieving data from database. We provide tag for the user input before storing in the database. The tag is useful to identify the user input from rest of the data and provides an easy way to sanitize the special characters while retrieving the data from database. The n-order SQL injection will be successfully prevented using this approach since the tags are never removed from the user input and the user-inputs from which the tags are removed will not be stored in the database unless they are tagged again.

Tag used : "&|\"

1. **Functions provided to the developer:**
    a. **tag_str($input):** To tag any special characters that might potentially change the SQL query. This function replaces any special character 'c' with '&|\c'.
    b. **mysql_iquery($query):** This function executes the query and stores or retrieves the data from/to database. The function results an object of the class **mysql_iresult** that extends the behavior of the class **mysqli_result**.

c. **mysql_inum_rows():** This function returns the number of rows in the retrieved dataset.

d. **mysql_ifield_tell():** This function gets the current field offset of result pointer.

We extended the mysqli_result in procedural and object-oriented methods.

e. mysql_ifield_count() : This function returns the number of columns of the query.

f. mysql_ifield_seek(): This function sets the field cursor to the given field offset.

g. mysql_ifield_tell(): This function returns the position of the field cursor.

2. The class mysql_iresult contains the following methods:

a. **fetch_object():** This method fetches the object and retags the object to escape the special characters. The special characters will be identified based on the tag "&|". Thus, any special character superseding "&|" will be replaced with "&|\\". These objects can be used internally in the application and can also be used in sub-queries or correlated queries. This method should not be used for variables that are displayed to the user.

b. **fetch_row():** This method fetches row and retags the object to escape the special characters. This method works same as above method.

c. **fetch_field_direct():** This method fetches the metadata for a single field. The field can be specified as an argument.

d. **fetch_field():** This method fetches the next field in the result set.

e. **free():** Frees the memory allocated to **iresult** object.

f. **fetch_assoc()**: This method fetches the data set and retags it to escape the special characters. This method also works as the same as fetch_object().

g. **fetch_array():** This method fetches the data into an array and retags the values in the array.

h. **fetch_all():** This method also re-tags the data to escape special characters.

The below functions should be used only to display the result to the user. The data obtained by these methods should be tagged again to store in the database.

i. **fetch_object_toDisplay()**

j. **fetch_row_toDisplay()**

k. **fetch_assoc_toDisplay()**

l. **fetch_array_toDisplay()**

m. **fetch_all_toDisplay()**

The above functions mimic the behavior of respective functions in the mysqli_result object. They fetch the data from the database but don't retag them. Instead, the function untags the input so that the user can use the data set to display to the user. The user can also use that data to build a JSON object or CSV file. It is the responsibility of the application programmer to take care that this data is **not** pushed to the database without tagging them.

The procedural functions available for the application programmer are:

a. **mysql_ifetch_all()**

b. **mysql_ifetch_array()**

c. **mysql_ifetch_assoc()**

d. **mysql_ifetch_field_direct()**

e. **mysql_ifetch_field()**

f. **mysql_ifetch_fields()**

g. **mysql_ifetch_lengths()**

h. **mysql_ifetch_object()**

i. **mysql_ifetch_row()**

3. Changes that must be done by Developer:

The developer should take care of the following conditions before executing the application.

3.1 Condition 1:

All the user input (except passwords; passwords can be hashed to escape special characters) that is used in SQL query should be tagged using tag_str() function.

3.2 Condition 2:

All the queries should be called using mysql_iquery() function.

3.3 Condition 3:

All the data that are retrieved using fetch_[object|row|assoc|array|all]_toDisplay() should be re-tagged using tag_str() ,if the data have to be stored in to the database again.

4. Limitations of the API:

a. Since tagging input changes the data in the database, the developer should be very careful while using the data. The data should be retrieved or stored using only mysql_iquery function.

b. The application programmer has to take care where and how the data will be used. The programmer has to use different methods to retrieve same data if it is used for different purposes.

c. We don't recommend tagging passwords. The passwords will be changed if there are any special characters. Thus passwords should be hashed or sanitized using some other methods.

d. Untagged data in database breaks the security. We developed a robust library but it doesn't mean that application developers will not make mistakes. Misuse of functions can cause untagged input to enter into database and cause n-order sq. injection.

5. Future Work

This library can be extended as a functionality of the database to prevent special characters changing the query. For example, the database can change the datatype of the user input data, that automatically adds/ removes the tags. The database can append metadata instead of tag to specify that the data has to be sanitized before using it again in a query.

This SQL injection library can be implemented for PHP interpreter which will take care of tagging and untagging of inputs. The tag used here is "&|\". In future, the PHP interpreter can use metadata instead of tags to hook input and output functions. The data that is tagged by interpreter can be sanitized and used.