ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 5, Issue 6, June 2016

# SQL Injection Attack

**Ms. Rupam Rai**
**ASM Institute Of Management & Computer Studies (IMCOST) C-4, Wagle Industrial Estate, Near Mulund Check Naka,**
**Thane (W), Mumbai - 400604**
**Department of MCA, University Of Mumbai**

***Abstract: SQL Injection attacks are done on websites. These websites accept user inputs and produce information requested by user. Attacker modifies these input data and adds own SQL statement along with user input. Due to this, attacker gets access to all information stored confidentially in website's database. This information may contain user's credentials, net banking information, and online booking system data etc. Attacker can misuse this information, in addition attacker can execute malicious statements that may turn off database server, resulting unavailability of websites.***
***Many researchers have proposed techniques to detect SQL injection attacks. In this research paper, different types of SQL Injection attacks are mentioned. The stored procedure may be used as counter measure for SQL Injection attack.***

***Keywords: SQL Injection, Database, SQL Query, Stored Procedure***

## I. INTRODUCTION

World Wide Web has experienced remarkable growth in E Commerce, individuals, governments. It is found that web applications can give effective, efficient and reliable solutions to the problems of handling ecommerce. For example, many people pay their bills, book the hotels or give exams online. These websites stored huge data of users.

Attacker tries to get all such information from website database. Attackers can certainly harm many lives by gaining access to it or by making changes to it. For example, if an attacker obtains the bank account details of an individual, he can misuse this information (like account number, account balance etc.) he can also alter the data to cause harm to the concerned individual. Another example is attacker can attack on real time website where information is modified every second such as online auction, sale, booking etc. Attacker can prevent user from booking, temporarily shut down websites. This is some time referred to as denial of service attack. So user cannot access websites.

To access data from database, Structured Query Language is normally used. SQL helps to communicate with database. SQL can insert data, edit data, delete data and retrieve data. User gives input in web application form which is added with SQL statements. Example, user gives his username and password to log in websites. These credentials are added to SQL query. Then this query is executed on database to verify user. If credentials are correct, user can log in. Attacker identifies such input fields and types his SQL code instead of actual inputs. Such altered query is then executed and allows attacker to access user account.

## II. DEFINITION

SQL Injections attack (SQLIA) are attacks by which an attacker alters the structure of the original SQL query by injecting SQL code in the input fields of the web form in order to gain unauthorized access to database.

***A) ATTACK INTENT***

Below is list of various SQL injection attack intent performed to obtain information

***1). Identifying injectable parameters:***

The attacker wants to probe a Web application to discover which parameters and user input fields are vulnerable to SQLIA.

***2). Performing database finger-printing:***

The attacker wants to discover the type and version of database used in Web application

***3). Determining database schema*:**

To correctly extract date from a database, the attacker often needs to know database schema information, such as table names, column names, and column data types.

***4). Extracting data:***

These types of attacks employ techniques that will extract data values from the database. Depending on the type of the web application, this information could be sensitive and highly desirable to the attacker. This is most common type of attacks.

***5). Adding or modifying data:***

In this attack, attacker tries to add or modify information.

***6). Performing denial of service:***

These attacks are performed to shut down the database of a Web application.

***7.) Evading detection:***

This category refers to certain attack techniques that are employed to avoid auditing and detection by system protection mechanism.

## III. ILLUSTRATION OF SIMPLE ATTACK

There are many methods to perform SQLIA. Tautology is conditional statement that evaluates to true if the condition is true. This type of method is used in WHERE clause of SQL query. Attacker inserts condition in SQL query that always evaluates to true. So if WHERE clause contains OR statement, attacker will get result event though other parameter value are not correct.

The SQL query get executed in database is "select * from customer where customerid = 1 or 1=1".

So this query will return not only customer 1 information but also all other customers too. Here 1=1 always evaluates to true.

## IV. TYPES OF SQL INJECTION ATTACK

There are multiple methods by which a Web application can be attacked. Each of these methods are discussed in detail to illustrate how each of them is used to attack the database of the application.

***A) Tautologies:***

The general goal of a tautology-based attack is to inject code in one or more conditional statements so that they always evaluate to true. The consequences of this attack depend on how the results of the query are used within the application. The most common usages are to bypass authentication pages and extract data. In this type of injection, an attacker exploits an injectable field that is used in a query WHERE conditional. Transforming the conditional into a tautology causes all of the rows in the database table targeted by the query to be returned.

Original query : select * from customer where customerid=1

Altered Query: select * from customer where customerid=‟ ‟ or 1=1

***B) Illegal/Logically Incorrect Queries***

This attack lets an attacker gather important information about the type and structure of the back-end database of a Web application. The attack is considered a preliminary, information-gathering step for other attacks. The vulnerability leveraged by this attack is that the default error page returned by application servers is often overly descriptive. In fact, the simple fact that an error messages is generated can often reveal vulnerable/injectable parameters to an attacker .When performing this attack, an attacker tries to inject statements that cause syntax, type conversion, or logical error into the database.

Query: select * from customer where customername=‟ ‟+ convert(int,(select top 1 name from sysobjects where xtype=‟u‟))

In the attack string, the injected select query attempts to extract the first user table (xtype=‟u‟) from the database‟s metadata table (assume the application is using Microsoft SQL Server, for which the metadata table is called sysobjects). The query then tries to convert this table name into an integer. Because this is not a legal type conversion, the database throws an error.

For Microsoft SQL Server, the error would be:"Microsoft OLE DB Provider for SQL Server (0x80040E07) Error converting nvarchar value 'users' to a column of data type int."

***C. Union Query***

In union-query attacks, an attacker exploits a vulnerable parameter to change the data set returned for a given query. With this technique, an attacker can trick the application into returning data from a table different from the one that was intended by the developer. Attackers do this by injecting a statement of the form: UNION SELECT <rest of the injected query>.Because the attackers completely control the second/injected query, they can use that query to retrieve information from a specified table. The result of this attack is that the database returns a dataset that is the union of the results of the original first query and the results of the injected second query.

Original Query: select customername from customer where customerid=1

Injected Query: select customername from customer where customerid=1 union select customerphoneno from customer

Here query will return all customers phone number along with 1st query.

***D) Piggy-Backed Queries***

In this attack type, an attacker tries to inject additional queries into the original query. in this case, attackers are not trying to modify original intended query; instead, they are trying to include new and distinct queries that piggy back on the original query. As a result the database receives multiple SQL queries. Attacker tries to retrieve, insert, and modify data.

Vulnerability to this type of attack is often dependent on having a database configuration that allows multiple statements to be contained in a single string.

Example: If the attacker inputs ; "drop table users--" into the customerid field, the application generates the query:: SELECT * from customer where customerid =‟‟‟‟;drop table users This query will drop table users so no one will be able to login further.

***E) Alternate Encodings:***

In this technique, attackers modify the injection query by using alternate encoding, such as hexadecimal, ASCII, and Unicode. Because by this way they can escape from developer's filter which scan input queries for special known "bad character".
For example, attacker can use char() instead of actual character. So application code may not identify such encoded string and they will treat it as normal input. By this technique, different attacks could be hidden in alternate encodings successfully.

Select * from customer where customerid=‟‟‟‟; exec(char(0x73687574646f776e)) –

In the above query, char code for "SHUTDOWN" command is added.so database server will shut down after execution of this query.

***F) Inference***

In this attack type, attacker tries to change the behaviour of database response. There are two types of inference.

***1).Blind Injection:***

Attacker adds code that always evaluates to true/false. Blind injection is useful when attacker cannot view error message from database. So attacker checks response from website.

First Query : select * from customer where customerid=1 and 1=2. Here if there is no input validation, query will not return anything since condition is false.

Second Query: select * from customer where customerid=1 and 1=1. Here query display customer information which is different response from previous query.so attacker can find such injectable fields and use them other attack.

***2) Time Based:***

This attack uses if then statement to inject code. Majority times "WAITFORDELAY" is used to pause database response. If the condition is true then database response in delayed.so attacker gain information about applications.

Query: select * from users where username=‟abc‟ and substring(„password‟,1,1)=‟0‟ waitfor delay ‟00:00:03‟. So if the response is delayed for 3 seconds attacker will come to know about user password information.

## V. PREVENTING SQL INJECTION ATTACK

Main cause for SQL injection attack is no input field validations. Developer and database administrator should implement all input validations to prevent SQL injection attack. Here we explain some of the way to minimize vulnerability for SQLIA.

***1) Input type checking:***

If the input field is alphabetic then only alphabets should be used. Allowing no alphabetic character may cause to input field vulnerable.

***2) Hide Url Extensions:***

We should hide extension of website address. Attacker gets knowledge of database from website extension. Following table shows websites and database generally used with them

Websites and Database

| Sr.No | Website | Database |
|---|---|---|
| 1 | Aspx | MS SQL Server |
| 2 | Php | Mysql |
| 3 | Jsp | Oracle |

***3) User input validation against anomaly tokens:***

All inputs entered by user should be checked against SQL operators (+,-, =) , special characters („„ ; , -- ) and keywords (delete , drop ,shutdown ).

For this separate tables in database are stored which defines all special characters, keywords and operators.

When user enters input , this input is matched against these anomaly If any match is found Log entries in database will be created.

Log entry table contains columns like

- IP address
- Page name
- Input
- Date time

This log information will help us to identify what sort of input is passed. So according to input, more validations can be implemented.

## VI. STORED PROCEDURES

Majority of SQL injection attacks are performed on inline query that construct query with user input.

Inline query used in above section:

```
"Select * from customer where customername = "" + textbox1.text + " " "
```

Here textbox input is added in query which then executed on database server. As discussed earlier , various attack are possible on this inline query.

We proposed that use of stored procedure instead of inline query can minimize SQL injection attack.

A stored procedure is an operation set that is stored. Typically, stored procedures are written in SQL. Since stored procedures are stored on the server side, they are available to all clients. Once the stored procedure is modified, all clients get new version of stored procedure. Store Procedure hides business logic.

They provide better performance than inline query. Reusability is another advantage of Stored Procedure

Following syntax explains stores procedures in MS SQL SERVER.

```
Create Procedure Procedure – Name
 (
Input parameterName1 datatype,
Input parameterName2 datatype,
 .
 .
 Input parameterNameN datatype
 )
 AS
 BEGIN
  Sql statement used in the stored procedure
 END
```

So our inline query can be re written using stored procedure as follows :

```
Create Procedure GetCustomerDetails
(
  @Customername varchar(50)
 )
 AS BEGIN
  Select *    FROM    customer
 WHERE    customername = @customername
 END
```

Here user enter input customer name in textbox ,so this input is passed as parameter in stored procedure.

So all text entered in textbox field will be considered as single input in parameter. So even if attacker add SQL injection code in textbox, query will run against those customer that has name given in parameter.

So stored procedures are more efficient than inline query.

## VII. CONCLUSION

SQL injection Attacks are a serious hazard to the growing popularity of web applications. The main target of these attacks is the database of the Web application and attackers have developed various methods for the same. All the common attack methods are explained with simple examples for each of them. Also the counter measure that can be taken is described that how stored procedures are useful to minimize SQL injection Attack.

## VIII. REFERENCES

[1] *https://www.Google.com*

[2] *https://en.wikipedia.org/wiki/SQL*

[3] *http://www.merriam-webster.com/dictionary/artificial%20intelligence*

[4] *http://www.codeproject.com/Articles/11020/SQL-injectionattacks/*

[5] *W. G. Halfond, J. Viegas and A. Orso, "A Classification of SQL Injection Attacks and Countermeasures," College of Computing Georgia Institute of Technology IEEE, 2006.*