ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 2, Issue 3, pp: (33-38), Month: July - September 2014, Available at: www.researchpublish.com

# SQL Injection Attack- Types and Classification A Review

Sitharthan. S, Sankaran. S

Department of Computer Science and Engineering
Saveetha School of Engineering, Thandalam, Chennai, India

***Abstract:*** **SQL injection is a technique where the attacker injects an input in the query in order to change the structure of the query intended by the programmer and gaining the access of the database which results modification or deletion of the user's data. In the injection it exploits a security vulnerability occurring in database layer of an application.**

**SQL injection attack is the most common attack in websites in these days. Some malicious codes get injected to the database by unauthorized users and get the access of the database due to lack of input validation. Input validation is the most critical part of software security that is not properly covered in the design phase of software development life-cycle resulting in many security vulnerabilities. This paper presents the techniques for detection and prevention of SQL injection attack. There are no any known full proof defences available against such type of attacks. In this paper some predefined method of detection and the some modern techniques of preventions are discussed. This paper also describes countermeasures of SQL injection.**

***Keywords:*** **SQL injection attack,**

## I. INTRODUCTION

As per the Open Web Application Security Project (OWASP), the Structured Query Language Injection Attack (SQLIA) is regarded as number one web application vulnerability in the past years. In this modern computer era database has become very essential in any web applications. All the web applications that are being developed has database connectivity in some form, hence making it database dependent. In an average about 71 attempts of attack is performed on an application. Some applications experience about 1300 times in an hour at the peak **[1].**

A successful SQLIA can

- Read/Extract sensitive or confidential data from database.
- Modify data in database (Insertion/Updating/Deletion).
- Executing operations such as shutdown of the DBMS and other harmful operations.

The insertion or injection of an SQL Query via the input data from the client to the application results in an SQL injection attack. Malicious input statements are given in an SQLIA. This malicious inputs when executed at the database result in unpredicted behaviour and thus the security of the database is compromised. The input given by the attacker is executed as it is. The database cannot differentiate between an input given by the clients/attacker and a malicious input **[2].**

## II. BACKGROUND OF SQLIA'S

### *A. Background of SQLIA's*

An SQL injection attack is occurred when a client/attacker manipulates the expected effects of an SQL query by inserting or injecting new keywords into the query. This manipulated query is used to gain unauthorized access to the database by inserting the query through the input field of a web form, of a web application. The SQL query that is given as input is converted to form an SQL code **[1] [2].**

The two important characteristics of SQLIA‘s:

Injection Mechanisms and Attack Intent

***a. Mechanisms of SQL Injection***

Different mechanisms are used for input of malicious statements or code into an application. The most general and common mechanisms of injection are focused in this section.

1. *Injection through user inputs:* In this type, attacker injects malicious SQL commands into the user input query. Depending on the development and deployment of a web application in an environment, there are several methods by which an input can be read from a web application. The most commonly used input methods include the web form which is transmitted via an HTTP GET OR POST request to a web application **[3].** Using this type of injection, attacker can gain unauthorized access of web application and its underlying database.

2. *Injection through server variables:* This is the most common type of SQL injection. Server variables are a collection of variables that defines the HTTP request, environment, and various other network parameters. These include the two most common form submission methods, GET and POST, as well as other intricate injection avenues such as HTTP header variables and sessions. The bulk of injection attempts are made through these server variables, either through entering malicious input into the client end of the application or by crafting their own request to the server.

3. *Injection through cookies:* Web applications gather the files which contain state information stored on the client machine, known as cookies. When the client returns to the web application, the state information can be restored by these cookies. Since these cookies are stored on the client machines, the client has total control of the contents of the cookies. These cookies can be modified by malicious clients to build SQL queries to attack a web application. **[20].**

4. *Second Order Injection:* In this type, attacks are performed directly on the database or the system, sending malicious input that can be used at a later time. A second order injection attack differs significantly from regular SQLIA’S (first-order) based on the objectives. Like a first order injection attacks, the second- order injection attacks are not performed at the time input are given to the database, but the attacker uses the knowledge of when and where the inputs are stored and uses this knowledge to plan an attack that is executed when the database or application is being used.

***b. Attack Intent***

Based on the intention or the goal of an attacker the attacks can also be classified **[1].** Hence all types of attacks that are described has one or more intention or goals as given below.

1. *Parameters that are Injectable:* The attacker uses the vulnerable input fields and parameters to perform an SQLIA on the web application.

2. *Fingerprinting the Database:* Different databases respond differently to queries and attack. The attacker gains knowledge of the type and version of the database and uses this information for fingerprinting the database. Database specific attacks are performed if the type and version of the database are known to the attacker **[1].**

3. *Data Extraction:* This is the most common objective of an SQLIA. In this type, different methods are used to extract data from the database. To an attacker, these information could be of sensitive or highly desirable depending on the type of web application.

4. *Modifying the Data:* In this type, the data in the database can be modified or new data can be added.

5. *Denial of Service:* In this type, the database is shutdown. Hence, the services to a user are denied. Attacks such as dropping a table and locking the table come under this type.

6. *Detection Evading:* In this method, the attacker uses some attack methods to evade from being detected by the security mechanisms of the system **[1].**

7. *Bypassing Authentication:* In this type of attack, the attacker gains access to the database bypassing the

authentication of the database and web applications. Hence an attacker has control of the access privileges and the rights to the database and the application.

8. *Remote Command Execution*: In this type of attack, the functions and stored procedures that are available to the database users are executed using remote commands.

9. *Escalating Privileges:* These types of attacks are intended to escalate the privileges of an attacker taking advantages of errors in the code and the logical flaws.

## III. TYPES OF SQLIA'S

In this section, the different types of SQL Injection Attack are discussed in a detailed manner. Each of the attack type is given a name, attack intents, description of the attack, an example of the attack and the references to these attack types that discuss the attack types in detail [1].

Most of the attacks are not performed separately, one or more of them are combined depending upon the attacker and the aim of the attacker.

### *A. Tautologies*

*Attack Intent:* Identifying injectable parameters bypassing authentication, extracting data.

***Description:*** In this type of attack, a code is injected into the input fields of the web application and one or more conditional statements are executed that are always true when executed. This is one of the most widely and commonly used method to bypass authentication and data extraction. If the attack was performed successfully, it returns a set of record, or a result if some action is performed.

***Example:***

The query for login is:
SELECT * FROM user_info WHERE logID='' or 1=1 --AND pass1=''
The conditional statement OR **(1=1)** makes the entire WHERE clause to a tautology. The query evaluated true for all records in the table and returns them. In the above example, there is a value returned and hence a not null value is evaluated, making the authentication process successful allowing the user to login successfully without valid credentials. This method is also employed to extract data from the database. **[9, 4]**

### *B. Illegal/Logically Incorrect Queries*

*Attack Intent:* Extraction of data, performing database finger-printing, identifies injectable parameters.

***Description***: When a logically or wrong SQL Query is sent to the database, error messages are sent from the databases which may contain sometimes useful information that can be used for debugging. This information may sometimes be helpful for the attacker in finding vulnerabilities in the system and hence in the database of the application also.

***Example:***

Select * from <table name> where userId = <id> and password = <wrongPassword> or 1=1;
This is a query for sending an error message for wrong password.
In this type of attack, the attacker gets to know the information of the table such as the name of the table and the field names of the fields in the table, which helps him for a more coordinated attack at the later stage. Credit cards are a classic example for this type of an attack in which the name of the table and details are gathered. Similarly target specific attacks can be performed if the schema of the database is known. **[4, 10]**

### *Union Query*

*Attack Intent:* Extraction of data, Authentication bypassing.

***Description:*** In this attack type, a dataset is returned that is a result that is an UNION of the original query and the result of the injected query that is inserted into the vulnerable parameter. The UNION query combines two or more queries together and gives the result which gives the fetched rows from the queries participating in the UNION.

***Example:*** An attacker could inject the text "' UNION SELECT cardNo from Credit Cards where acctNo=**10032** --" into the login field, which produces the following query:

SELECT accounts FROM users WHERE login='' UNION SELECT cardNo from CreditCards where acctNo=**10032 --** AND pass='' AND pin=
The original first query returns a null set since there is no logical equal to "", but the second query returns information from the table. In this example, it returns the column "CardNo" for the account "**10032". [5, 6]**

### *C. Piggy-backed Queries*

*Attack Intent:* Extracting data, adding or modifying data, performing denial of service, executing remote commands.

***Description:*** In this type of attack, the statement after the ";" is executed. It is a very dangerous type of attack which could damage the database, sometimes may destroy it also. It could bring large loss of data if it is successfully executed.

***Example:***

SELECT account FROM user WHERE login='abc' AND pass=''; drop table user --' AND pin=123
The above query is generated if the attacker inputs "'; drop table users --" into the *pass* field.
The execution of the query is done, and after the first query the delimiter is read and the second injected query is executed. After the successful execution of the second injected query the table user is dropped, destroying some information that may be valuable. Other queries can be used to insert new values, execute stored procedures, etc. **[5, 6]**

### *D. Stored Procedure*

*Attack Intent:* Escalating privileges, denial of service, remote command execution.

**Description:** The database engine runs the routine that are stored, these are known as Stored Procedures. These stored procedures may be user defined or that are provided by the database by default. The different ways of attacking the database are dependent on the type of stored procedure.

***Example:***

CREATE PROCEDURE DBO.is Authenticated
@username varchar2, @pass varchar2, @pin int
EXEC("SELECT account FROM user
WHERE login='" +@username+ "' and pass='"
+@password+ "' and pin=" +@pin); GO
In the example, the constructed query string at the line 5 is replaced by a call to the stored procedure. The authentication of the user's credentials is indicated by a true/false value that is returned by the stored procedure. The stored procedure starts executing if the attacker inserts the code "'; SHUTDOWN; --" into the username field or the password field. This generates the query:
SELECT accounts FROM users WHERE login='abc' AND pass=' '; SHUTDOWN; --AND pin=
This is similar to a Piggy backed attack where the statement after ";" is executed resulting in the shutdown of the database. Successful execution of this attack results in huge amount of loss. [**5, 11, 7, 6]**

### *E. Blind Injection*

*Attack Intent:* Extraction of data, Theft of data.

*Description:* The error messages that are displayed by the database are hidden by the developers for security reasons and only the generic error pages are displayed to the user when it is accessed, making difficult for an attacker to get information from the database [19]. It is when an attacker sends true/false questions to steal/theft data.
***Example*:** SELECT name FROM <tablename> WHERE id=<username> and 1 =0 -- AND pass = SELECT name FROM <tablename> WHERE id=<username> and 1 = 1 -- AND pass =

After execution of the queries, error messages are returned. If the web application is secure, there is a chance of injection if the inputs are not validated in advance. After the insertion of first query, if the attacker receives error, he does not know whether it was due to input validation or query formation error. But, the id field is vulnerable if there is no error message after the submission of second query. **[5, 7]**

### *F. Timing Attacks*

*Attack Intent:* Shutting down the server.

***Description:*** The timing delays of the response from the database are observed from the database which helps for an attack on the database to gather information. Depending on the logic that has been injected, an if then statement is used to run a time delay statement forced by the SQL engine. Depending upon the time to load page the condition such as true or false is determined. A response delay can be given using the WAITFOR time clause.

**Example-** Declare @s varchar(500) select @s = db_nameO if (ascii(substring(@s, I, I)) & ( power(3, 0))) > O waitfor delay '0:0:20'

If the first bit of the first byte of the name in the database is 1 then there is a delay in the response time of 20 seconds. **[7, 8]**

### *G. Inference*

*Attack Intent:* Data extraction, Identify injectable parameter, determine database schema.

***Description:*** In this type of attack, the query is modified to recast it in the form of an action that is executed based on the answer to a true/false question about data values in the database **[8].** In this type of injection attack the attacker has difficulty in attacking the site that has been secured so much that there is no detailed error messages that are displayed. Even after a successful injection attack there is no useful feedback or response from the database. Hence, the attacker uses different methods for extracting details from the database. The attacker inserts commands into the site and then observes how the response of the site is. By this method, the attacker comes to know the parameters that are vulnerable to attack and also additional information about the database. Most commonly there are two attacks possible, they are extracting the data, detection of vulnerable parameters. **[7, 1]**

### *H. Alternate Encodings*

*Attack Intent:* Vulnerability detection evasion.

***Description:*** In this type, the injection query is modified by alternate encoding, changing characters to some other characters in the queries. By this way, the attacker evades filters for “wrong characters”.

All different kinds of SQL injection attack can be hidden using this method.
Example- SELECT name FROM <tablename> WHERE id=’’ and password=O; exec (char (O x73687574646j776e))

The hexadecimal encoded character are taken as input that is used as char function that returns actual character. This encoded string, shutdowns the database when the command is executed. **[7, 1, 4]**

## IV. CONCLUSION

It is obvious from above description that SQL injection attacks are one of the largest classes of security problems. Most existing technique either require developers to manually specify the interfaces to an application or, if automated, are often inadequate when applied to modern, complex web applications. In this paper we have reviewed the most popular existing SQL Injections attack issues. And also we have presented a review on various types of SQL Injection attacks and their working methods.

## REFERENCES

[1] William G.J. Halfond And A. Orso, A Classification Of SQL Injection Attacks And Counter measures,‖ 2006.

[2] Tajpour; M. Masrom; M. Z. Heydari.; S. Ibrahim; "SQL injection detection and prevention tools assessment " Proc. Of ICCSIT 2010, vol.9, no., pp.518-522, 9-11 July 2010.

[3] N. W. Group. RFC 2616 – Hypertext Transfer Protocol – HTTP/1.1. Request for comments, The Internet Society, 1999. http://msdn.microsoft.com/library/ default.Asp?url=/ library/en-us/iissdk/ html/9768ecfe-8280-4407-b9c0-844f75508752.asp

[4] Indrani Balasundaram. , Dr. E. Ramara. An Approach to Detect and Prevent SQL Injection Attacks in Database Using Web Service. IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.1, YEAR 2012

[5] Anley. Advanced SQL Injection In SQL Server Applications. White paper, Next Generation Security Software Ltd., 2002.

[6] S. Labs. SQL Injection. White paper, SPI Dynamics, Inc.,

[7] Abhishek Kumar Baranwal. Approaches to detect SQL injection and XSS in web applications. EECE 571B, TERM SURVEY PAPER, APRIL 2012

[8] Gould, Z. Su, and P. Devanbu. JDBC Checker: A Static Analysis Tool for SQL/JDBC Applications. In Proceedings of the 26th International Conference on Software Engineering (ICSE 04) – Formal Demos, pages 697–698, 2004.

[9] M. Dornseif. Common Failures in Internet Applications May 2005. http://md.hudora.de/presentations

[10] Litchfield. Web Application Disassembly with ODBC Error Messages. Technical document, @Stake, Inc.,2002.http://www.nextgenss.com/papers/webappdis.doc. 2002.http://www.spidynamics.com/assets/documents/WhitepaperSQLInjection.pdf.

[11] F.Bouma.StoredProceduresareBad,O‘kay?Technicalreport,Asp.NetWeblogs,November2003.http://weblogs.asp.net/fbou ma/archive/2003/11/18/38178.aspx.