Overview on SQL Injection Attacks
Ankita Kushwah¹ Gajendra Singh²
¹²SSSIST, Sehore

Abstract— SQL injection attack is the most common and difficult to handle attacks now days. SQL injection attack is of five types. In these paper details of SQL injection is mentioned.
Key words: SQLIA, Types of SQLIA, Query Tokenization.

I. INTRODUCTION
SQLIA is an attack on web-applications. SQLIA occurs when adversary changes the logic, semantics or syntax of an SQL query [1]. The query which is generated dynamically based on user input, maliciously crafted with SQL keywords, operators, strings or literals, executes in the database server. The aim of the intruder for the SQLIA is to access database for which he is unauthorized [2]. So, accessing information beyond limitations intruder applies SQLIA in the form of queries which are syntactically correct [3]. The results of SQLIA are as below:

A. Bypass Authentication
It is a serious type of attack. Intruder can access the sensitive information about another user and can access the information available in his account. This attack is applied when intruder bypasses the validation (checking of username and password) phase and can access the authorized area/space of victim [4].

B. Confidentiality Loss
When the confidential data from the database is extracted by the intruder, confidentiality is lost.

C. Integrity Loss
The major effect of SQLIA is integrity loss. When intruder access the database as well as he apply SQLIA in such a way that he can have authority of altering the database, it has a major impact on the integrity of the system as he can alter the database as he thinks. It be comes dangerous if this is done in banking web applications, as intruder can get privilege to the accounts also. Unwanted operations: intruder can perform unwanted operations such as shutting down the database, change it, upload files or delete files from database [5].

This paper emphasizes on various aspects of SQL Inspection. Section II shows prevention techniques and operations in the previous work done in this field. Section III contains proposed solution using tokenization approach as well as conclusion part of this paper and future research directions to prevent SQLIA.

II. BACKGROUND AND RELATED WORK
The attacker’s objective for using the injection technique is lies in gaining control over the application database. In a web based application environment, most of the web based applications, social web sites, banking websites, online shopping websites works on the principle of single entry point authentication which requires user identity and password. A user is identified by the system based on his identity. This process of validation based on user name and password, is referred as authentication. Web architecture illustrated in Fig.1 shows general entry point authentication process. In general client send a HTTP request to the web server and web server in turn send it to the database layer. Database end contains relational tables so queries will be proceeding and result will be send to the web server. So entire process is database driven and each database contains many tables that are why SQLIA can be easily possible at this level.

Fig. 1: Web Architecture

SQL Injection is a basic attack used for mainly two intentions: first to gain unauthorized access to a database and second to retrieve information from database. Function based SQL Injection attacks are most important to notice because these attacks do not require knowledge of the application and can be easily automated [6].

Oracle has generally aware well against SQL Injection attacks as there is are multiple SQL statements that support (SQL server and Postages SQL), a no. of executive statements (SQL servers) and no. of INTO OUTFILE functions (MYSQL) [7]. Also use of blind variables in Oracle environments for performance reasons provides strong protections against SQL Injection attack.

There are two types of SQLIA detection:

A. Static Approach
This approach is also known as pre-generating approach. Programmers follow some guidelines for SQLIA detection during web application development. An effective validity checking mechanism for the input variable data is also requires for the pre-generated method of detecting SQLIA.

B. Dynamic Approach
This approach is also known as post-generated approach. Post-generated technique are useful for analysis of dynamic or runtime SQL query, generated with user input data by a web application. Detection techniques under this post-generated category executes before posting a query to the database server [2, 7].

C. Classification of SQLIA
SQLIA can be classified into five categories:
- Bypass Authentication
- Unauthorized Knowledge of Database
- Unauthorized Remote Execution of Procedure
- Injected Additional Query
- Injected Union Query
D. Bypass Authentication

It is already discussed in Section I. Researchers have proved that query injection can’t be applied without using space, single quotes or double dashes (--). In bypass authentication, intruder passes the query in such a way which is syntactically true and access the unauthorized data [8]. For example:

```
SELECT SALARY_INFO from employee where username='' or 1=1 - - 'and password='';
```

This SQL statement will be passed because 1=1 is always true and - - which is used for comments, when used before any statement, the statement is ignored. So the result of this query allows intruder to access into user with its privileges in the database [3].

E. Unauthorized Knowledge of Database

In this type of attack, intruder injects a query which causes a syntax, or logical error into the database. The result of incorrect query is shown in the form of error message generated by the database and in many database error messages, it contains some information regarding database and intruder can use these details. This type of SQLIA is as follows:

```
SELECT SLARY_INFO from employee where username = 'rahul' and password =convert(select host from host);
```

This query logically and syntactically incorrect. The error message can display some information regarding database. Even some error messages display the table name also.

F. Unauthorized Remote Execution of Procedure

SQLIA of this type performs a task and executes the procedures for which they are not authorized. The intruder can access the system and perform remote execution of procedure by injecting queries. For example:

```
SELECT SALARY_INFO from employee where username=''; SHUTDOWN; and password ='';
```

In above query, only SHUTDOWN operation is performed which shuts down the database [2].

G. Injected Additional Query

When an additional query is injected with main query and if main query generates Null value, even though the second query will take place and the additional query will harm the database. For example:

```
SELECT SALARY_INFO from employee where username='rahul' and password=''; drop table user';
```

First query generates Null because the space is not present between ‘and’ and password, but the system executes the second query and if the given table present in database, it will be dropped.

H. Injected Union Query

In this type of attack, the intruder injects a query which contains set operators. In these queries, the main query generates Null value as a result but attached set operators data from database. For example:

```
SELECT SALARY_INFO from employee where username='' and password='' UNION SELECT SALARY_INFO from employee where emp_id='10125';
```

In above query, the first part of query generated Null value but it allows the intruder to access the salary information of a user having id 10125.

I. Major Elements of SQLIA

It is shown in various research papers that SQLIA can’t be performed without using space, single quotes and/or double dashes. These are the major elements of SQLIA. SQLIA is occurred when input from a user includes SQL keywords, so that the dynamically generated SQL query changes the intended function of the SQL query in the application.

When user input types a number, there is no need to use single quotes in the query. In this case SQL Injection is injected by using space. This query can be done on original query.

Original Query:  
```
SELECT * from employee where emp_id=10125;
```

The injection query can be of this form using space:

```
SELECT * from employee where emp_id=10125 or 1=1;
```

The injection query shown below is a query which uses single quotes:

```
SELECT*from employee where emp_name='rahul'or1=1;
```

In this case if an employee with name rahul is present in database, information is retrieved. But if the name is not present, even then the query is executed because the statement 1=1 is always true.

The injection query may contain double dashes (--) 
```
SELECT * from employee where emp_name='rahul'--' and SALARY_INFO>25000;
```

SQLIA is a prominent topic and lots of research work has been done for the detection and prevention of SQLIA. In [3] the author proposes the TransSQL model. In this model author proposes a model for SQLIA prevention. TransSQL is server side application so, it does not changes legacy of web application. This model uses the idea of database duplication and run time monitoring. The proposed model is fully automated and the result shows the effectiveness of system. TransSQL propose to use two data bases, one is original relational database and another (LDAP) is copy of the first one, But data is arranged in hierarchical form. When a query is paused by the user, the system checks if the query contains the injection or not. The queries inserted in both original database and LDAP. If result of both databases is same, it shows the input query is free from injection, but if results are different, it means, the query contains injection. So the system shows the result as Null. The major shortcoming of this models not applicable for injection queries which contain instances, alias, UNION ad UNIONALL [11].

In [9], tokenization method is propose, which is efficient but applied on original as well as query with injection. It is not possible for all queries that their original query is already stored. In [2], the author proposes rule-based detection technique, which is based on classification task. For a particular query, rule dictionary is generated and query is replaced with the serules. If another query is present, the rules are applied in new entry and using classification approach, identify that new query contains the SQL injection or not.[2] proposes, two levels of
authentication: SQL authentication and XML authentication, and every query is passed through both systems for checking and preventing against SQLIA.

J. Query Tokenization

It is a SQLIA prevention technique proposed in [10]. The query tokenization technique is implemented by query parser method. In this method, the original query and query with injections are considered differently. Figure 2 shows the overall process of tokenization.

Tokenization is performed by detecting space, single quotes (‘ ’) or double dashes (- -) and all strings before each symbol constitute of token. Tokens are decided on the basis of spaces between them. All the tokens are stored as an element of the array. Two arrays resulting from both original and a query with injection are obtained with their lengths. If the length of both arrays is same, there is no injection. If lengths are different there is injection. Table 1 and Table 2 shows resulting arrays after tokenization for query 1 and query 2 which are as follows:

1) Query 1: Original Query

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT * from Employee where emp_name='Rahul';</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2) Query 2: With Injection

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT * from employee where emp_name='Rahul' or '1'='1';</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Tokenization Result of Original Query

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT * From Employee Where Emp_Name= Rahul</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Tokenization Result of Query with Injection

When index of table 1 and table 2 are compared the length of both array are unequal. So, it is sure that the second query has injection.

III. CONCLUSION AND FUTURE WORK

Now–a-days, when web applications have become popular and many companies rely on them, the need of security of web application increases. SQLIAis the topmost threat to web applications. In SQLIA, intruder passes an injected query in the system and access the unauthorized data. If an injected query is accepted by any relational database, it will be accepted by all databases which are based on relational approach, for example, SQL,MySQL, MS Access. So, if input query will be checked by two different databases, using different approaches (relational and hierarchical approaches), then the proper checking of injection can be done. This paper is focused on theSQLIA, its classification and its prevention techniques. This research paper proposes introduction of a new system which issued for the prevention of SQL injection and also accepts and checks the query which contains instances, alias, UNION or UNION ALL, etc. set operators, by applying tokenization on hierarchical and relational databases.

REFERENCES


