SQL Injection Attacks: Technique and Prevention Mechanism

Gaurav Shrivastava
Mahakal Institute of Technology, Ujjain

Kshitij Pathak
Mahakal Institute of Technology, Ujjain

ABSTRACT

In today's era where almost every task is performed through web applications, the need to assure the security of web applications has increased. A survey held in 2010 shows web application vulnerabilities and SQL Injection attack ranked among top five[1]. SQL Injection attack (SQLIA) is performed by those persons who want to access the database and want to steal, change or delete the data which they do not have permission to access [1]. In SQLIA adversary requests through a malicious query which shows some confidential data [2]. In research, it is also proved that when a network and host-level entry point is highly secured, the public interface provided by an application is the one and only source of SQL injection attack. SQLIA can't be applied without using space, single quotes or double dashes [3]. So to prevent SQLIA, these options are taken in observation. Previous model [10] used JDBC-LDAP library which did not support instances, alias and set operations (UNION and UNION ALL). If a query with injection is accepted by any database which is based on relational approach, then it will be accepted by all databases that are based on relational approach. This paper is focused on SQLIA and its techniques and encounters the shortcoming of previous models. This paper proposed a model which uses two databases one relational and other hierarchical to ensure about injection in a query, compare the results by applying tokenization technique on both databases. If the results are same, there is no injection, otherwise it is present. The proposed model uses a tokenization technique so; query containing Alias, Instances and Set operations can also be blocked at the entry point.

Keywords

SQLIA; Classification of SQLIA; Query Tokenization.

1. 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:

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].

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

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 becomes 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 SQLIA. Section II shows prevention techniques and operations in the previous work done in this field. Section III contains proposed solution using tokenization approach section IV shows the comparison between existing and proposed model. Section V contains the conclusion part of this paper and future research directions to prevent SQLIA.

2. BACKGROUND 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.showes 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.

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.

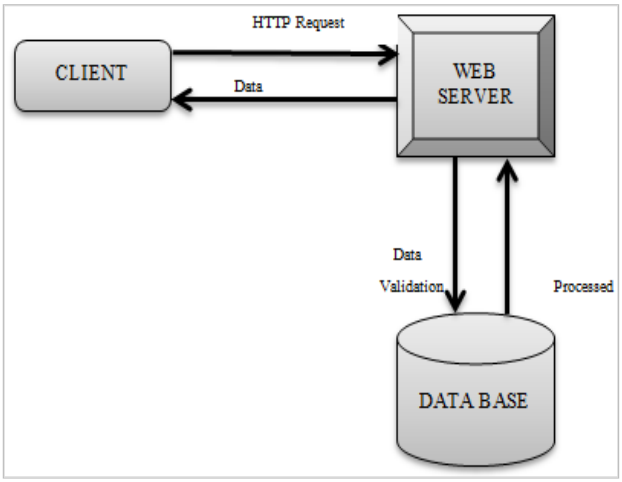


Figure 1. Web Achitecture

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), a no. of executive statements (SQL servers) and no. of INTO OUT FILE 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:

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.

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].

2.1 Classification of SQLIA: SQLIA can be classified into five categories:

1) Bypass Authentication
2) Unauthorized Knowledge of Database
3) Unauthorized Remote Execution of Procedure
4) Injected Additional Query
5) Injected Union Query

1) Bypass Authentication: 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=1is 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].

2) Unauthorized Knowledge of Database: In this type of attack, intruder injects a query which causes a syntax, or logical error in to 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 is logically and syntactically incorrect. The error message can display some information regarding database. Even some error messages display the table name also.

3) 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].

4) 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.

5) 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.

2.2 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:

Injected Query: 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.

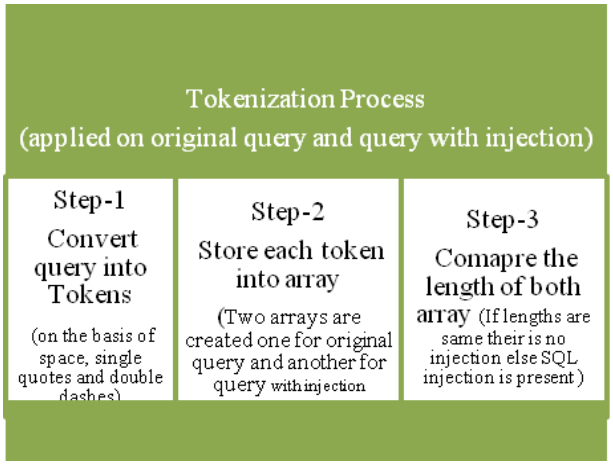
The injection query may contain double dashes (--)

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 query is 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 that it is not applicable for injection queries which contain instances, alias, UNION and UNIONALL [11]. In [9], tokenization method is proposed, which is efficient but applied on original as well as query with injection 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 these rules. 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 though both systems for checking and preventing against SQLIA.

2.3 Query Tokenization: The query tokenization technique is implemented by query parser method. In this method, the original query and query with Injections are considered differently. Table 1 shows the overall process of tokenization.

Table 1. Tokenization Process



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:

Query 1: Original Query SELECT * from Employee where emp_name= 'Rahul';

Query 2: Query with Injection SELECT * from employee where emp_name='Rahul' or '1'= '1';

Table 2. Tokenization Result of Original Query

0	1	2	3	4
SELECT * from	Employee	Where	emp_name=	Rahul

Table 3. Tokenization Result of Query with injection

0	1	2	3	4	5	6	7	8
SELECT * from	Employee	Where	emp_name=	Rahul	or	1	=	1

When index of table 2 and table 3 are compared the length of both array are unequal. So, it is sure that the second query has injection.

2.4 Evaluative Study of Previous Models: Table 4 shows all the pros and cons of previous model in the field of SQL Injection Attack. Table illustrated below compares five models Sania:[7], SBSQLID[8], RDUD[9], TransSQL[11] and Tokenization [10] on the basis of their advantages and drawbacks.

3. PROPOSED MODEL

SQLIA is a server type of web vulnerability, which impacts badly on web applications. In this section, a novel model for SQLIA prevention is proposed. As mentioned in previous section, several models are proposed for prevention of SQLIA, but they are not applicable for all type of injection attacks. SQLIA prevention via double authentication through tokenization is an approach to control SQLIA. This paper proposes double authentication process on both relational and hierarchical databases by applying tokenization approach on both databases. This task is performed via three steps.

Step 1: Query Forwarding

Step 2: Tokenization process on query

Step 3: Comparison of array index

Figure 2 shows the proposed architecture of SQLIA prevention through double authentication via tokenization by using above three essential steps.

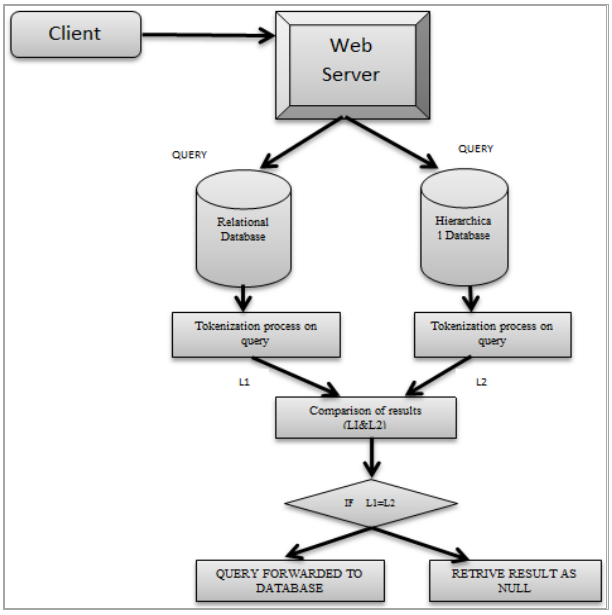


Figure 2. Proposed Architecture

Step 1: Query Forwarding- When a query comes from a user via user interface, the input query is forwarded to both databases, one which is created by relational approach and other based on hierarchical approach.

Step 2: Tokenization Process on Query- the input query is divided into various tokens on the basis of space, single quotes and double dashes between them. Once the tokens are decided, they are stored in array. Tokenization process is applied on both databases.

Step 3: Comparison of Array Index- In this step, the array length of both the arrays are compared. If the length of L1 and L2 are same, there is no injection present in the query and the query is proceed further to main database for retrieving result. But if the lengths L1andL2are different, then injection exists and query is not forwarded to the database. The result is a NULL value.

Table 4. Comparative Analysis of all existing models

Model	Advantage	Disadvantages
Sania: [7]	1) It can detect SQL vulnerabilities during the development and debugging phase of a web application. 2) It identifies vulnerable spots by analyzing SQL queries .	1) It requires knowledge of database in lack of knowledge attack can not be handled.
SBSQLID: [8]	1) The main advantage of this approach is that, error massage generated does not contain any Mata-data. (Information about the database which could help the attacker)	1) Web service is not integrated with the web application. Any modification that should be done to the system should be done in such a way that it should be supported by the web service.
RDUD[9]	1) It uses supervised learning approach using SVM to learn and to classify a query at run time. 2) It is based on classification task.	1) Special care is taken for maintaining the integrity of the web profile files to avoid poisoning of web profiles. 2) Not applicable for By pass Authentication
Trans SQL [11]	1) It is a server side application. So, it does not change the legacy web application. 2) Query is checked twice before retrieving information.	1) This model is unable to prevent against set operations, instances, alias directly.
Query Tokenization [10]	1) It converts query into tokens which contains between space, single quotes and double dashes. .2) Applied for all types of SQLIA.	1) The original query of input query which contain injection, must be stored.

This model uses two types of databases, pone hierarchical and other relational. The aim of using two databases having different representations is that when one query with injection applied on relational database is accepted, it can be accepted by all databases based on relational databases and important information can be disclosed. While using different databases having different storage strategies, shows different results for same query. And if the results are different, it shows the presence of injection.

4. COMPARATIVE ANALYSIS

Table 5 shows the comparison between existing SQLIA prevention technique and proposed technique on the basis of different SQL injection type like Bypass authentication, Unauthorized Knowledge of Database, Unauthorized Remote Execution of Procedure, injected additional query, Injected Union & Union ALL Query, Injected Alias query and Injected Instance query. Existing TransSQL technique uses LDAP database which may trap in some of the case like UNION,UNION ALL etc. because the part of SQL commands, UNION, UNION ALL are not supported by JDBC-LDPA library[11]. This paper proposed technique which uses XML-Authentication and Query Tokenization technique to eliminate the drawbacks of existing model. Table below shows the outcomes of existing TransSQL model and the expected outcomes of proposed model.

Table5 .SQLIA Techniques Countermeasures

SQL Injection Types	**SQLIA prevention technique**	
	Existing Model's Outcomes	**Proposed Model's Expected Outcomes**
1. Bypass Authentication	Prevented	Prevented
2. Unauthorized Knowledge of Database	Prevented	Prevented
3.Unauthorized Remote Execution of Procedure	Prevented	Prevented
4.Injected Additional Query	Prevented	Prevented
5. Injected Union & Union ALL Query	Not Prevented	Prevented
6. Injected Alias query	Not Prevented	Prevented
7. Injected Instance query	Not Prevented	Prevented

5. 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. SQLIA is the top most 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 the SQLIA, its classification and its prevention techniques. This research paper proposes introduction of a new system which is used 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.

6. REFERENCES

[1] R. Ezumalai, G. Aghila, "Combinatorial Approach for Preventing SQL Injection Attacks", 2009 IEEE International Advance Computing Conference (IACC 2009) Patiala, India, 6-7 March 2009.

[2] Asha. N, M. Varun Kumar,Vaidhyanathan.G of Anomaly Based Character Distribution Models in the,"Preventing SQL Injection Attacks", International Journal of Computer Applications (0975 – 8887) Volume 52– No.13, August 2012

[3] Mehdi Kiani, Andrew Clark and George , "Evaluation e Detection of SQL Injection Attacks".The Third International Conference on Availability, Reliability and Security,0-7695-3102-4/08, 2008 IEEE.

[4] V.Shanmughaneethi, C.EmilinShyni and Dr.S.Swamynathan, "SBSQLID: Securing Web Applications with Service Based SQL Injection Detection" 2009 International Conference on Advancesin Computing, Control, and Telecommunication Technologies, 978-0-7695-3915-7/09, 2009 IEEE

[5] Yuji Kosuga, Kenji Kono, Miyuki Hanaoka, Hiyoshi Kohoku-ku, Yokohama, Miho Hishiyama, Yu Takahama, Kaigan Minato-ku, "Sania: Syntactic and Semantic Analysis for Automated Testing against SQL Injection" 23rd Annual Computer Security Applications Conference, 2007, 1063-9527/07, 2007 IEEE

[6] Prof (Dr.) Sushila, MadanSupriyaMadan, "Shielding Against SQL Injection Attacks Using ADMIRE Model", 2009 First International Conference on Computational Intelligence, Communication Systems and Networks, 978-0-7695-3743-6/09 2009 IEEE

[7] A S Yeole, B BMeshram, "Analysis of Different Technique for Detection of SQL Injection", International Conference and Workshop on Emerging Trends in Technology (ICWET 2011) – TCET, Mumbai, India, ICWET'11, February 25–26, 2011, Mumbai, Maharashtra, India. 2011 ACM.

[8] Ke Wei, M. Muthuprasanna, Suraj Kothari, "Preventing SQL Injection Attacks in Stored Procedures".Proceedings of the 2006 Australian Software Engineering Conference (ASWEC'06).

[9] Debasish Das, Utpal Sharma, D. K. Bhattacharyya, "Rule based Detection of SQL Injection Attack", International Journal of Computer Applications (0975 – 8887) Volume 43– No.19, April 2012.

[10] NTAGW ABIRA Lambert, KANG Song Lin, "Use of Query Tokenization to detect and prevent SQL Injection Attacks", 978-1-4244-5540-9/10/2010 IEEE.

[11] Kai-Xiang Zhang, Chia-Jun Lin, Shih-Jen Chen, Yanling Hwang, Hao-Lun Huang, and Fu-Hau Hsu, "TransSQL: A Translation and Validation-based Solution for SQL-Injection Attacks", First International Conference on Robot, Vision and Signal Processing, IEEE, 2011