*IOSR Journal of Computer Engineering (IOSRJCE)*
*ISSN : 2278-0661 Volume 1, Issue 1 (May-June 2012), PP 31-37*
*www.iosrjournals.org*

# Token Sequencing Approach to Prevent SQL Injection Attacks

ManveenKaur[1],Arun Prakash Agrawal[2]
[1]*(Computer Science and Engineering, Amity University, India)*
[2]*(Asst Prof, Computer Science and Engineering, Vivekanand Insitute of Technology and Science, Ghaziabad, U.P, India)*

**ABSTRACT :** *Internet, the network of networks represents an insecure channel for exchanging information leading to a high risk of intrusion or fraud. Many web applications remain under the attack of hackers who intentionally try to access secret information stored at the backend database by circumventing its security system. One of the major approaches followed to perform these attacks is with the help of SQL Injection (SQLI) or (SQLIA). SQL injection is a technique used to attack databases through a website. It is a form of attack that comes from user input that has not been checked to see if it is valid. SQL injection is the subset of code which is not verified by the backend server and the aim is to run that code to derive the secret information.[1].*

*In this papera method is proposed in which two approaches, one static in which the database is created and another dynamic in which the query structure against the previously stored query structure is compared. If the two structures match then search is stopped and query is regarded as a valid query. The Algorithm has been developed in JAVA.*

***Keywords-****Group, Malicious, SQLIA, Token, Vulnerability*

## I. INTRODUCTION

SQL injection is a very serious threat to web applications. In recent years, with the emergence of internet, databases have become even more important than ever before and are a critical part of network security. Database is the storage brain of a website.[1] A hacked database is the source for passwords and the other important information like credit card number, account number etc. Importance should be given for preventing database exploitation by SQL injection. [1] [6].Already some work has been done to prevent this attack. The methods used earlier are cumbersome, as they need to modify the source code which is an overhead, also it minimizes the runtime responsetime .In this approach there is no need to modify the source code and use of modern age processor architecture is done in a multithreaded way to minimize response time.

There are different types of web applications that are present at the server. To run the application, itmust be properly configured first. The core part of any Web application is stored on the server site within the application server. The different browser supportedlanguages like PHP,ASP,J2EE,Java/JSP,Perl,CGI contain the core in the form of software program.

As the J2EE web applications have good compatibility therefore they have been used in this work.

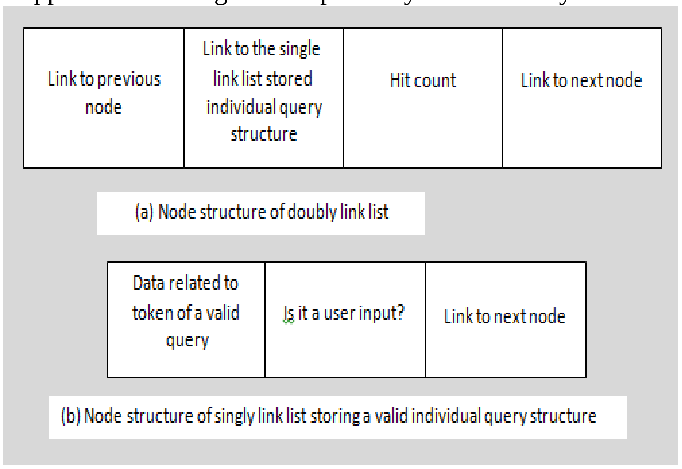
Link to previous node
Link to the single link list stored individual query structure
Hit count
Link to next node
(a) Node structure of doubly link list
Data related to token of a valid query
Is it a user input?
Link to next node
(b) Node structure of singly link list storing a valid individual query structure

*Figure 1 Node structure*

## II. THE GRAPHICAL REPRESENTATION OF LINKED LISTS

The figure below represents how Singly Linked Lists with same number of Tokens are grouped together in a Doubly Linked List. In order to keep track of all the doubly linked lists while searching, we store their reference in Array, that points to these Lists. This makes our Searching task simple. We have made the use of dynamic approach to searching, in order to make our search more efficient and time saving.

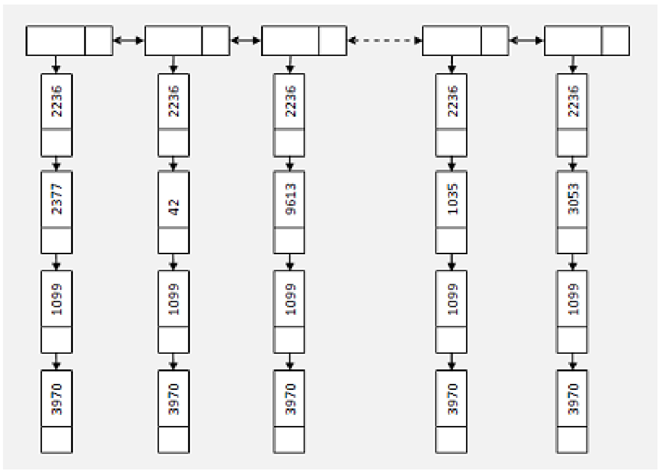

*Figure 2Singly Linked Lists with same number of tokens.*

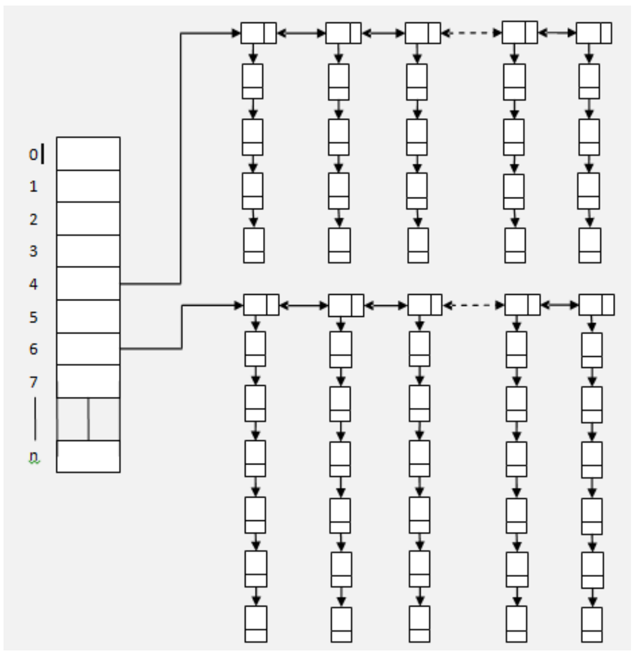

*Figure 3Array pointing to Groups.*

## III. CATEGORIES OF SQLIA

**SQL manipulation -** It means modifying SQL statements by the use of several operations like UNION. Another way is by changing Where clause to get different output.

**Code Injection -** IT is the technique of inserting inserting new SQL statements into the vulnerable SQLstatement Example Append EXECUTE command to the code.

**Function call injection -** Process of inserting numerous database function calls into SQL statement.

**Buffer overflows -** Caused by using function call injection. This attack is possible when Server is un –

patched.

**SQL manipulation -** It means modifying SQL statements by the use of several operations like UNION. Another way is by changing Where clause to get different output.

**Code Injection -** IT is the technique of inserting inserting new SQL statements into the vulnerable SQL statement. Example Append EXECUTE command to the code.

**Function call injection -** Process of inserting numerous database function calls into SQL statement.
**Buffer overflows -** Caused by using function call injection. This attack is possible when Server is un – patched.

### 3.1 Sql Injection attacks examples

Suppose a user wants to access the email-id in the form "email me my password" form .[4] He writes the query

```
SELECT data FROM  table WHERE Emailinput
= '$email-input';
```

The "$email-input" contains what the user types in the email address form field.[4] [2]

Suppose that the attacker knows that the database is vulnerable to attacks, he can type any malicious code on the form field to get more information

```
Y'; UPDATE table   SET email ='hacker @
Ymail.com' where email = 'Smith@Ymail.com';
```

Here the extra quote which is followed by semicolon that allows the close the statement & run another statement of his own.[2] [4]

When the malicious code is executed by the application,

```
SELECT data FROM table WHERE Emailinput = 'Y';
UPDATE table SET email = 'hacker @ Ymail.com'
WHERE email = 'Smith @ymail.com';
```

This code resets the email- id of "Smith @Ymail.com" to "hacker @ Ymail .com".

Hacker now has a mail – id & password to the application which is under someone else account.

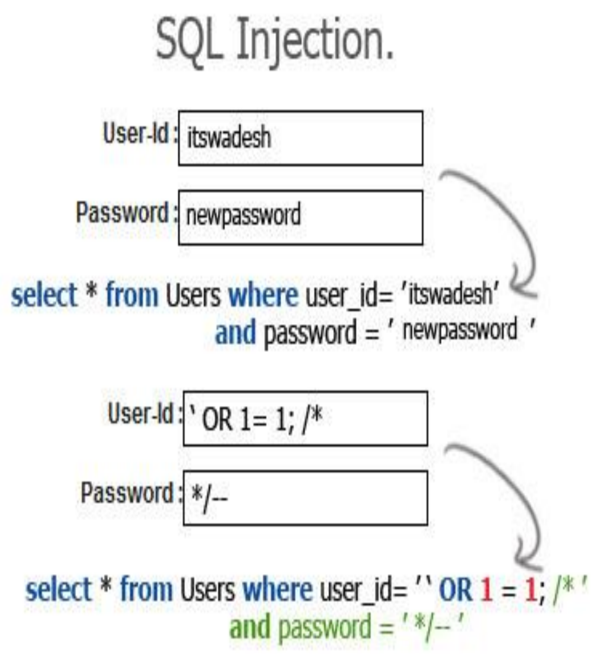
SQL Injection.
User-Id : itswadesh
Password : newpassword
select * from Users where user_id= 'itswadesh' and password = ' newpassword '
User-Id : ' OR 1= 1; /*
Password : */--
select * from Users where user_id= '' OR 1 = 1; /* ' and password = ' */-- '

Figure 4. An SQL Injection attack.

## IV. HOW THE WEB SITES CAN BE ATTACKED

**Tautology**

Tautology means a formula that is true in every case.Here the code is injected in such a way that it always evaluates to true.If a condition is transformed to a tautology, it can return all the rowsof the database. The main aim of the attacker is to make all the conditional statements evaluate to true by injecting code in one or more statements.

```
"SELECT * FROM student

WHERE name = ' "

+request.getParameter("name")+

" 'AND password =' " + request.getParameter("pass")+

 " ' " ;
```

```
SELECT * FROM Student

WHERE name = 'Amit'

AND Password = ' 'OR 'a' LIKE '%a%'
```

**End of Line comment**

```
SELECT * FROM Student

WHERE Studname = ‘admin’--‘AND password = ‘’
```

This statement logs the hacker asadmin user.[2] [4]

**Logically Incorrect query**

Sometimes it is the intentional approach of the attacker to write incorrect query. The application server of the backend database returns error messages that contain enough information for the attacker to make good attempt.

**Union Query**

An attacker sometimes causes information to be derived from a desired table that was not actually intended by the developer.In order to do this, he first exploits the vulnerable parameter so that the dataset which is returned by the query can be changed.This is done by making use of UNION SELECT. [4] E.g.

```
SELECT * FROM Student

WHERE login” =

“UNION SELECT  password from

Stud_info where stud_name =’Amit’ – ‘AND pass=”
```

The database will return column “Password” for stud_name “Amit”.

**Piggy backed Query**

In this approach additional queries are added to the original query. There is no need to change the original query, instead more are added which “piggy- back” the original query
[4]
E.g. If the user inputs “; drop table student—

```
SELECTRollno FROM Student

WHERE login = ‘Amit’ AND Pass = ”;

drop table Student --’
```

Database will recognise (“;”) & and execute the second injected query. As a result table is dropped.

**System stored procedures**

System stored procedures are as vulnerable as other queries. In order to attack the database, attackers first determine the database and then make queries against the stored procedures. E.g.

```
CREATE PROCEDURE EMPLOYEE_SEARCH

@EmpName Varchar(200) = NULL AS

DECLARE @Sql nVarchar(2000)

SELECT @Sql = ‘SELECT

EmpId,EmpName,Category,Price’+’FROM Employee

WHERE’

IF @Empname IS NOT NULL

SELECT @Sql + ‘EmpName  LIKE” ’ +@EmpName +

“ “EXEC(@sql)
```

Now if the input is 1’ or’1’ = ‘1’; exec mastr.dbo.xp_cmdshell ‘dir’--, then the query executed at the server is [4]

```
SELECT  EmpId, EmpName FROM Employee

WHEREEmpName LIKE ‘1’ or ‘1’ = ‘1’;

exec master.dbo. xp_cmdshell ‘dir’--‘
```

It returns all the rows from table and executes command DIR.

**Inference**

In this technique when the injection attack becomes successful, no error message is returned by the database server. So therefore attacker has to insertcommands in thewebsite and observe the behaviour of that website. When the response of the website changes, he can deduce different values from the database.

**Blind injection**

In this type of attack, true/False questions are asked and information is derived from the behaviour of the page. If the result of injection is true, the site functions normally otherwise if it is false the page behaves differently.

**Timings attacks**

In this technique, the attacker uses If-then statement to derive information. The attacker makes note of the timing delays in the response of a database. E.g WAITFOR keyword causes the database to delay its response by a specific time.

## V. SOME WAYS TO AVOID THESE ATTACKS

1. Minimum use of dynamic SQL queries should be made if there is some alternate way.

2. The Stored procedure must be executed usinga safe interface such as callable statements in JDBC or command object in ADO

3. All the input from the user must be validated thoroughly.

4. In order to run the database, use of low privilege account must be made.

5. Proper roles & privileges must be given to the stored procedure that is used in the application.

6. Use of parameterized stored procedures with embedded parameters must be made

## VI. Algorithm

I. Input a query that probably contains the SQLIA code.

II. Convert the query into tokens & separate the tokens.

III. Convert the tokens into corresponding integer values.

IV. Search the query structure that matches with the input query structure by using appropriate Searching technique.

V. If a match is found, search is successful and the query structure is a valid one

VI. Else

VII. The search is not successful and the query is regarded as invalid.

VIII. EXIT.

## VII. Conclusion

Sql injection is a common technique that the hackers use to attack databases to extract their confidential information. These attacks have thus made it essential to develop methods to disable such attacks so that no invalid query can exploit the database. In this paper a method has been developed to differentiate between the valid queries and invalid queries. In the dynamic approach developed a query is first examined to see if it is valid , if it is so by comparing its structure only then the query is allowed to execute. This is an efficient method as it minimizes the searching time and response time because searching is performed in a multithreaded way as well as it makes use of modern age processor.There are also some complexities involved such as Token separation, Token to integer conversion and the searching process in link list. Complexity of Token to integer conversion is 0(n) where n is the total number of literals in all Tokens of query.

Therefore it can be said that ,[5]

1. SQLIAs have

a) .Many Sources

b)Many Goals

c)Many types

2. Detection techniques can be effective but are limited by lack of automation.

3. Prevention techniques can be very effective but should move away from developer dependence. [5]

## References

[1] http://en.wikipedia.org/wiki/SQL-injection

[2] http://www.unixwiz.net/techtips/sql-injection.html

[3] http://cwe.mitre.org/documents/vuln-trends.html

[4] http://ferruh.mavituna.com/sql-injection-cheatsheet

[5] Preventing sql injection attacks Using AMNESIA William G.J Halfondand Alessandro Orso Georgia Institute of Technology. www.cc.gatech.edu/orso/papers/halfond.orso.ICSEDEM006.Presentation.pdf