SQLIJHS : SQL Injection Attack Handling System

Prof. Sumitra Pundlik
Assistant Professor
Computer Engineering Department
MITCOE,Pue 38

Rajnish Kumar
UG Student
Computer Engineering Department
MITCOE,Pue 38

Bhagyashree Gaikwad
UG Student
Computer Engineering Department
MITCOE,Pue 38

Aarti Aadhale
UG Student
Computer Engineering Department
MITCOE,Pune 38

Vaishali Waghmare
UG Student
Computer Engineering Department
MITCOE,Pune 38

ABSTRACT

The Database Security is major and important part of any business. The business must provide a non-vulnerable interface and assurance of security of customer's data stored on their server.

SQL Injection is of the type of attack used to access information illegally from stored data. In this paper we have given in detail description of SQL Injection attack. We have proposed SQL Injection Handling System (SQLIJHS) which will act as middle ware between user and database. SQLIJHS will help to Detect Mechanism, Avoid Mechanism and Recover Mechanism to secure database form theft.

1. Introduction

Data [1] is qualitative and quantitative representation of real time information. It can be used by storing in the form of database. The data can be stored in the table as columns and rows where column name indicates attribute and row indicates values related to that attribute. The Database Management System (DBMS) is combination of database models, supportive data structure, transaction and query processing mechanisms.

The Structure Query language (SQL) [1] is a language, using which we can perform operation on stored database. It allows us to perform operations such as insert, update, delete, and select on stored data which in the form of table.

Security [2] is concerned about protection of valuable information from misuse or theft from illegal users or resources. There are various types of security such as network security, computing security, data security, and information security etc. The Data security [2] is concerned about security of stored data in the form of database. Now a day's data theft is a serious issue for example theft of credit card information, theft of bank's user id and password and perform illegal online banking transaction etc.

The key of goal for moving toward database systems are: reliability, higher availability and, most importantly, the security that can be implemented more effectively in a database environment. With the increased deployment of such database applications there has been an increase in the number of attacks targeting such applications. One class of attacks targeted by database applications that is particularly dangerous is SQL injection attack. The SQL injection attack (SQLIJA)[3] is nothing but injection of malicious information in the query by unauthorized person or an attacker to change the data stored in the tables or to retrieve secure data from the tables without any permission. This could result in the attacker gaining access to the backend database and potentially sensitive information.[3]

In this paper we present a model handle SQL injection in a database system. The section 2 is about details of SQLIJ whereas section 3 is discussing about on related work. We are also giving in detail comparison of various tools which useful for detection, avoidance and recovery from SQLIJ attack in section 4.

2.1 The Basic Types of Attacks

2. Details of SQL Injection Attack

In this section we are discussing about various types of SQL Injection attacks along with common patterns.

2.2Common Patterns of attacks

The attacker can inject malicious information by using various patterns. From various papers [4][5][6] we have observed some common patterns discussed by various authors while talking about SQLIJA. The patterns are as follows,

1. UPDATE users SET password=˝newpwd˝ WHERE userName= ˝admin˝–˝ AND password=˝oldpwd˝

2. SELECT status FROM users WHERE user name=‘‘‘ or 1=1 – AND pass=‘‘‘
3. SELECT * FROM user WHERE username = 'admin'–' AND password = ‘’
4. SELECT * FROM users WHERE login=’ ’ UNION SELECT password from user_info where user_name=’abc’– AND pass= ‘ ’ .
5. SELECT accounts FROM users WHERE login= 'doe' AND pass='; SHUTDOWN;
6. SELECT accounts FROM users WHERE login=’doe’ AND pass=’’; drop table users –’ AND pin=123

Sr.No.	Name Of Attack	Description	Example
1	**Tautology attacks[6]**	In tautology attack the conditional statement is inserted in the query which will be true all time i.e. 1=1	Select * from staff where st_no=9 or **1=1;**
2	**UNION Attacks[6]**	In union attack user can inject code by using UNION key word in the query to perform illegal operations in the database.	SELECT accounts FROM users WHERE login=’’ **UNION SELECT** cardNo from CreditCards where acctNo=10032 -- AND pass=’’ AND pin= ”
3	**Logically incorrect query attacks[6]**	In this type of attack logically incorrect query can be used to identify type of data or gather overall information about the database or its tables. In this case attacker can take advantage if error messages for further injection.	SELECT accounts FROM users WHERE login=’’ AND pass=’’ AND pin= **convert (int,(select top 1 name from sysobjects where xtype=’u’))**
4	**Piggy Back attack[6]**	The injected query is added to the original query.	SELECT accounts FROM users WHERE login=’doe’ AND pass=’’; **drop table users -- ’ AND pin=123**

3. Related Work

Various authors handled different types of SQL Injection attacks. They have proposed detection, avoidance and recovery approaches to handle SQL Injection attack. The author [6] discussed about SCMAS is an architecture for Distributed hierarchical multi-agent system for blocking attacks to databases. This strategy helps to detect and prevent SQL injection attacks consisting of a multi-agent based architecture. SCMAS presents a hierarchical structure and this hierarchical structure distributes roles and tasks for the detection and prevention of SQL injection attacks. A Solution based on Multi-agent System. The agents handle capacities such as autonomy, social abilities, reasoning, learning, and mobility, among others. One of the main features of agents is their ability to carry out cooperative and collaborative work when they are grouped into multi-agent systems to solve problems in distributed manner.

The authors Xing Wang et.al. [7] proposed detection methods based on hidden web crawling. They have implemented an algorithm for purpose of raising web page coverage and enhance it as web scanner to detect SQL Injection attack where as various methods such as Query Tokenization [8] by query parser method, recurrent neural network trained by back propagation through time algorithm [9], obfuscation-based analysis [10] for static and dynamic analysis of submitted queries, Sunday pattern matching algorithm to implement data security gateway [11], Detection Defense Log model for client server technology[12], Encoding techniques and finger prints of SQL queries [13], AIIDA-SQL using code Based Reasoning [14],SQL Query removal method using combined static and dynamic analysis[15],Firewall for Anomaly detection[16] proposed to detect SQLIJA.

The Signature based Method proposed by [17] used Hirschberg algorithm, Finite state automata and SQL graph representation[18], Hybrid Encryption (PSQLIA-HBE) for authentication scheme[19], protective adaptive shell located between application and backend database[20] to prevent system from SQLIJA. The recovery approach is proposed by [21] as reconstruction using network recording solutions. So there is need to propose a system which can help to perform all three operations to provide more security to the data.

4. Tools Comparison:

In this section we are discussing about the comparison of available tools which are handling SQLIJA as shown in Table 3.1.

We have considered parameters such as open source, able to perform detection, recovery and avoidance/prevention from SQLIJA for comparison of tools. From above comparison we conclude that Pangolin [22], Havij [22] are able to perform detection, avoidance and recover for web application and those are open source also. But as we are considering SQLIJA at query level i.e. at database level where user is firing queries from SQL prompt. This an extension for DBCrypto [23] which is doing data security at query level by using encryption and decryption algorithm.

5. Proposed Model

In this we have proposed SQLIJ Handling System (SQLIJHS) as a middleware in between application (normal query window, web application etc.) and database as shown in Figure 5.1 .It will handle SQL injection attack patterns. The SQLIJHS will have three mechanisms

1. Avoidance Mechanism
2. Detection Mechanism
3. Recovery Mechanism

Table 3.1. Comparison of SQLIJA Handling Tools

Name of tools	**Features**	**Open source**	**Detection**	**Prevention**	**Recovery**
Pangolin[26]	Its goal is to detect and take advantage of SQL injection vulnerabilities on web applications.	yes	Yes	Yes	yes
Havij[26]	This tool that helps penetration testers to find and exploit SQL Injection vulnerabilities on a web page.	yes	Yes	Yes	yes
Sqlninja[26]	Sqlninja is a tool targeted to exploit SQLInjection vulnerabilities on a web application	No	Yes	Yes	partially
CANDID[26]	CANDID's natural and simple approach turns out to be very powerful	No	Partially	partially	Partially
WAVES[26]	The tool identify all points a web application that can be used to inject SQLIAs.	No	Partially	partially	Partially
JDBC-Checker[26]	It was not developed with the intent of detecting and preventing general SQLIAs	No	Partially	partially	partially
SecuriFly[26]	SecurityFly tries to sanitize query strings that have been generated using tainted input	Yes	partially	partially	partially
WebS SARI[26]	WebS SARI [use static analysis to check taint flowsagainst preconditions for sensitive functions.	No	Yes	Yes	yes
V1p3R [27](‘Viper”)	Used for Web application Penetration testing	No	Yes	No	No

6.Algorithm:

6.1.1 Exact Matching

Input : T, P

Output: Safe Query, Attack Alarm I

[a] match_count b 0;

[b] For i= 1 to n do

Begin

[d] If (P=Ti) then {

- Add 1 to match_count;

- Declare 'Safe Query';

- Exit; }

[e] End if;

[f] End For Loop;

[g] If (match_count=0) then {

- Declare 'Attack Alarm I';

- Call Approximate Matching; }

[h] Stop;

6.1.2 Approximate Matching:

Input : T, P, W

Output: Safe Query, Attack Alarm Final

[a] k = element_count(P);

[b] For i = 1 to n do {

[c] For j = 1 to k do {

[d] If (P[j] c T[i][j]) then

[e] D[i] b D[i] + 1 ;

[f] Enf if ; } }

[e] Edit_Distance b 0 ;

[f] F

or i = 1 to n do {

[g] Edit_Distance = MIN (D[i]); }

[h] If (Edit_Distance < W) then {

- Declare 'Safe Query' ;

- Execute P; }

[h] Else {

[i] - Declare 'Attack Alarm Final' ;

[j] - Block P; }

[k] End if;

[l] Stop;

6.2 Steps for implementing various pattern

There are different implementation algorithm for different patterns. So we used total 6 algorithm for 6 different types of injection which is given below:

1 .tautology
2. piggy backing
3. union attack
4. stored procedure
5. logically incorrect query

6.2.1 Steps for tautology

1. Capture Query as String. Ex.: query1;query2;query3
2. Split the queries by segregating the ";" as separator
   if n is number of ";" then no of queries is n+1
3. for each query split the words by removing white spaces and store in string array
4. get index of "=" in the array of words. If index of "=" is I
5. separate the strings at positions I-1 and I+1 from the array
6. perform equality check on the two strings separated
7. if found equal the result = "Tautology Detected" else "Valid query"
8. Repeat steps 3 to 6 for the next query till all queries are verified for Tautology

6.2.2 Steps for piggy backing

1.take a array variable
2.split query into subparts by ; and store all the query into array variable

3.Use for loop from 0 to n(no of queries)

a. convert it into lowercase and store it into another variable
b. then check in each query 1st word should be

either drop table or shut down for piggy backing attack.

c. if yes then return true else return false

6.2.3 Steps for union attack

1. Split the queries by searching the keyword 'UNION'.
2. Search the 'ALL' keyword in the query. If found remove 'ALL' keyword it will give us only query.
3. Search for 'SELECT' keyword in the query if not found then mark as 'UNION' attack.
4. If 'SELECT' keyword found in all queries then separated the column list from each query. i.e. The string after 'SELECT' and before 'FROM'.

5. From the column list separate each column by splitting string by commas.

6. Compare the no. of on each query. If not equal then mark it as 'UNION' be set.

7. If no. of columns also same then compare each column from the column list.

If no. of columns does not match then it is 'UNION' be set otherwise it is valid query.

6.2.4 Steps for Stored procedure

1. start , declare the two variables actualQ, execQuery as null

2. Convert query into lowercase

3. If create procedure or alter procedure are there in query then check there is exec or execute command or not

4. It checks after calling procedure there is tautology or not

5. If true then detected stored procedure attack

6. Else this is valid query

7. End

6.2.5 Steps for Logically incorrect logic

1. When we fire logically incorrect query, it checks that with all the patterns that we have implemented.

2. If that query is not found in that patterns then it is valid query for our application.

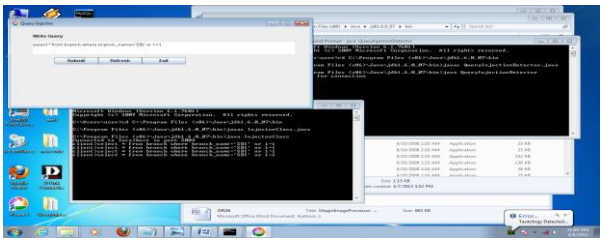
3. If query is valid then it is submitted to MYSQL database.

4. MYSQL returns the result of the user & user gets the idea about the table of our database.

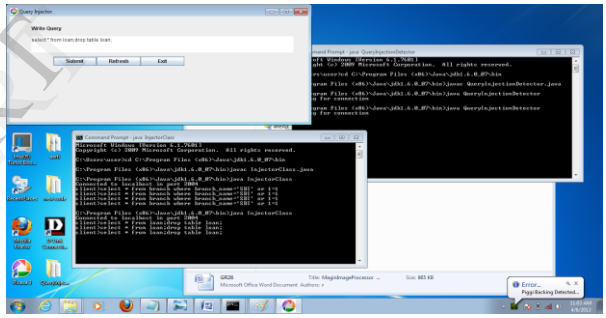
5.But before user gets that result our application detects that is logically incorrect query pattern attack.

7.Result

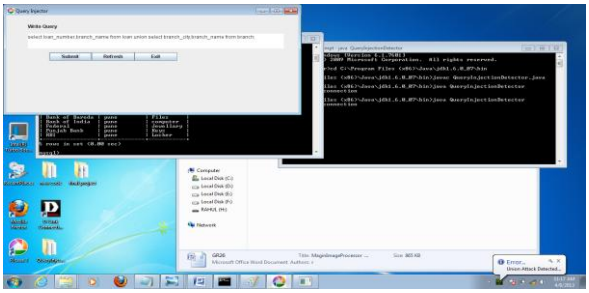
7.1 tautology



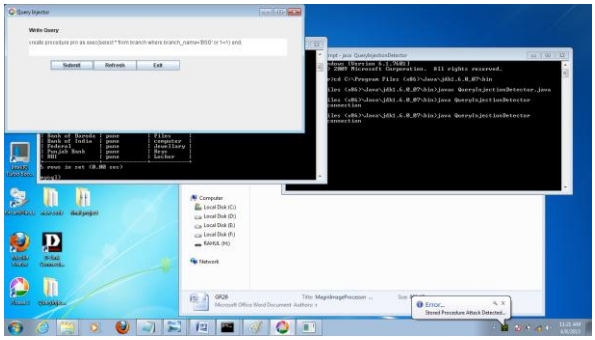
7.2. piggy backing



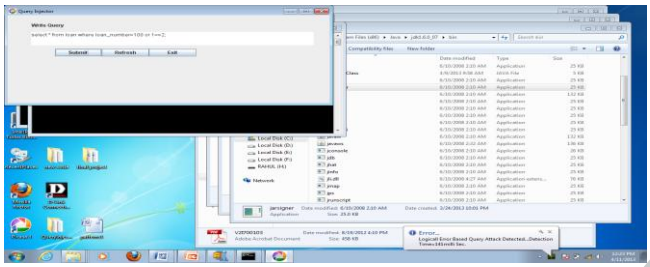
7.3. union attack



7.4 Stored procedure



7.5 logically incorrect query



8. Conclusion

SQL injection is a common technique hackers employ to attack these web-based applications. In this paper we have discussed various types of SQLI Attacks and different common patterns. We have also discussed about various approaches proposed by authors and comparison of working tools. We have discussed about our proposed SQLIJ handling system will be used to detect, avoid and recover injected query.

9. Future Scope

In future this mechanism can be used for distributed environment to handle injection from internet user, web applications etc. It can be implement in various places such as banking sector, airline reservation etc. to avoid illegal theft of data.

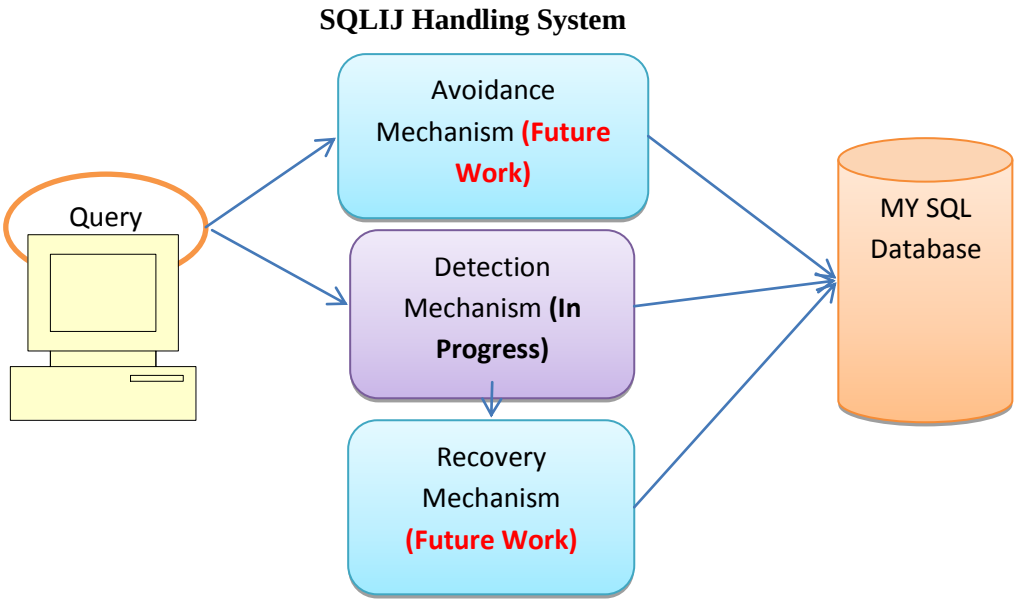


Figure 5.1 Proposed Diagram for SQL IJA handling system

10. References

[1] Korth Henry F., Siberschatz Avi, Sudarshan S., Database System Concepts 5th Edition

[2] Stallings William Cryptography and Network Security

[3] Evaluation of Anomaly Based Character Distribution Models in the Detection of SQL Injection Attacks.

[4] A Survey On Sql Injection: Vulnerabilities, Attacks, And Prevention Techniques By Diallo Abdoulaye Kindy and Al-Sakib Khan Pathan Department of Computer Science, International Islamic University Malaysia, Malaysia

[5] J. M. Corchado, M. Glez-Bedia, Y. De Paz, J. Bajo and J. F. De Paz, Replanning Mechanism for Deliberative Agents in Dynamic Changing Environments. Computational Intelligence, vol.24, pp.77-107, 2008.

[6] A. Damba and S. Watanabe, Hierarchical Control in a Multiagent System, International Journal ofInnovative Computing Information and Control, vol.4, no.12, pp.3091-3100, 2008

[7] Xin Wang, Luhua Wang, Gengyu Wei, Dongmei Zhang, Yixian Yang, HIDDEN WEB CRAWLING FOR SQL INJECTION DETECTION Proceedings of IC-BNMT20 10

[8] NTAGWABIRA Lambert, KANG Song Lin, Use of Query Tokenization to detect and prevent SQL Injection Attacks.

[9] Jaroslaw Skaruz, Franciszek Seredynski, Recurrent neural networks towards detection of SQL attacks.

[10] Raju Halder, Agostino Cortesi, Obfuscation-based Analysis of SQL Injection Attacks.

[11] Xu Ruzhi, Guo jian, Deng Liwu, A Database Security Gateway to the Detection of SQL Attacks, 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)

[12] Qian XUE, Peng HE, On Defense and Detection of SQL SERVER Injection Attack.

[13] Elisa Bertino, Ashish Kamra, James P. Early, Profiling Database Application to Detedct SQL Injection Attacks

[14] AIIDA-SQL: An Adaptive Intelligent Intrusion Detector Agent for Detecting SQL Injection Attacks

[15] Jeom-Goo Kim, Injection Attack Detection using the Removal of SQL Query Attribute Values.

[16] Ke Wei, M. Muthuprasanna, Suraj Kothari, Preventing SQL Injection Attacks in Stored Procedures.

[17] Jaroslaw Skaruz, Franciszek Seredynski, Recurrent neural networks towards detection of SQL attacks.

[18] Ke Wei, M. Muthuprasanna, Suraj Kothari, Preventing SQL Injection Attacks in Stored Procedures.

[19] Indrani Balasundaram, E. Ramaraj, An Authentication Scheme for Preventing SQL Injection Attack Using Hybrid Encryption (PSQLIA-HBE), European Journal of Scientific Research ISSN 1450-216X Vol.53 No.3 (2011), pp.359-368.

[20] Li Shan, Dong Xiaorui, Rao Hong, An Adaptive Method Preventing Database from SQL Injection Attacks, 2010 3rd International Conference on Advanced Computer Theory and Engineering(1CACTE)

[21] Allen Pomeroy and Qing Tan, Effective SQL Injection Attack Reconstruction Using Network Recording, 2011 11th IEEE International Conference on Computer and Information Technology.p

[22] Atefeh Tajpour, Maslin Masrom, Mohammad Zaman Heydari, Suhaimi Ibrahim, SQL Injection Detection and Prevention Tools Assessment

[23] Asavari deshpande (2012), 'DBCrypto: A Database Encryption System using Query Level Approach '