Intrusion Protection Against SQL Injection Attacks Using Reverse Proxy

Monika Ghodake Gitanjali Dabhade Gayatri Aher Mr.Sachin N.Wandre

Abstract- **Keeping in mind the increasing volume of real time transactions on the internet, security in the web applications is vital to protect the value and usability of assets. The level of security has neither grown as fast as the internet applications nor evolved as fast as the attacks and intrusion, exposing various vulnerabilities inherent in the internet bases services of the era.**
Internet has eased the life of human in numerous ways, but the drawbacks like the intrusions that are attached with the Internet applications sustains the growth of these applications. One such intrusion is the SQL Injection attacks (SQLIA). Since SQLIA contributes 25% of the total Internet attacks, much research is being carried out in this area. Here we propose a method to detect the SQL injection. A SQL Injection Attack usually starts with identifying weaknesses in the applications where unchecked users" input is transformed into database queries. Reverse Proxy is a technique which is used to sanitize the user's inputs that may transform into a database attack. In this technique a filter program redirects the user's input to the proxy server before it is sent to the application server. At the proxy server, data cleaning algorithm is triggered using a sanitizing application.

Index Terms: **SQL Injection, SQL attack, Security threats, Web application vulnerability.**

I. INTRODUCTION

The data security is an important aspect to the organizations which are developing web applications. It is a great challenge to the organizations to protect their valuable data against intruders, corruptions and malicious accesses. The main aim of database security is to provide the protection to the database from unauthorized access. The unauthorized or illegal action may execute the malicious query; because of this vulnerability database security may break. Researchers have developed the Intrusion Detection System (IDS) for enhancing database security against malicious access. Database security can be categorized into external attack and insider attack. In external attack sensitive data is compromised by authorized attempts, whereas insider attack is executed by an authorized user with the help of various malicious techniques. Attacks have increased as the large number of information systems is connected.

SQL injection is one of the most serious security threats for web applications and databases. The attacker can submit a SQL query or command through the web application And can gain access to the underlying confidential information, by bypassing authentication mechanisms, to modify the database and lead to execution of illogical code. This topic focuses on different types of vulnerability in database and the scheme used for detecting the Structure Query Language Injection Attack (SQLIA) and Cross Site Scripting attacks and prevention of the same using reverse proxy from the web applications. The main objective of this document is to illustrate the requirements of the project Security system. The document gives the detailed description of the both functional and non-functional requirements proposed by the proxy server. The purpose of this project is to provide an environment to secure the access of database and security for web applications. This is a project which focuses on the inherent aspects of Security system and attempts to develop a proper Security system to secure database access as well as web applications.

II. SQL INJECTION ATTACK:

SQL Injection Attackers include user data in the SQL query and arbitrary code is added in such a way that a part of the input is understood as SQL code. The

Monika R..Ghodake *, B.E Final year, Sinhgad Institute Of Technology,*
Lonavala-410401, Maharashtra,India
Gitanjali N.Dabhade *, B.E Final year, Sinhgad Institute Of Technology,*
Lonavala-410401, Maharashtra,India
Gayatri A. Aher *, B.E Final year, Sinhgad Institute Of Technology,*
Lonavala-410401, Maharashtra,India
Mr.Sachin N.Wandre, Assistant Professor *Computer Engg. Dept. Sinhgad Institute Of Technology,*
Lonavala-410401, Maharashtra,India

ISSN: 2278 – 1323

attackers always use this malicious SQL codes for input fields of a web form to gain access and update the data. These types of vulnerability allow an attacker to flow commands directly to a web application's underlying database and destroy functionality or confidentiality. There are two techniques of SQLIA i.e. access through input fields and access through URL. In first technique attacker always bypass the authentication of user i.e. password. Attacker can perform this technique through multiple queries, extended stored procedure and „or „condition etc. In second technique attacker manipulates the query string in URL. This vulnerability can be represented as:

*SELECT * FROM admin WHERE*
username ='admin123' AND
password = 'secret'

If in this query the attacker can enter [, , OR1=1– –] and [ ] instead of [admin123] and [secret], the query would take the form:

*SELECT * FROM admin WHERE*
username ='admin123' OR '1=1 – –
' AND password = ' '

Working of a SQL injection, as shown in the Fig.2, the condition equation of [1=1– –] and an empty password are mentioned in the query. After checking this condition equation and empty password, query would found a valid row in the table. An attacker may bypass all authentication measures, make changes to the database and gain unlimited access to the sensitive information on the server.

A. SQL INJECTION TYPES:
Following are the various types of attacks:

1. Tautology Attacks :
The tautology attack always uses conditional statements to inject the code. This attack bypasses the authentication of the web page and extracts the important data. The tautological attacker exploits where clause in the query.

SELECT accounts FROM users
WHERE Login ='nil' OR 1=1---
AND password = 'nil'

The conditional statement (OR 1=1) transforms the where clause into tautology. The injected query returns on null value that means the query is successfully executed.

2. UNION Attacks:
This technique is combine two queries using UNION keyword. First query is original and second is injected query. In this attack with the help of original query injected query will retrieve the important data of the user.3

SELECT accounts From user
WHERE login=' 'UNION SELECT
credit card WHERE accno=02220 –
– AND Password=' '

The first/original query returns the null value and second/injected query returns the important data from the “credit card” table.

3. Piggybacked Query:
In this type of attack additional query will be added along with original query. The additional query is a injected query which is also called piggy-back onto the original query. Because of additional query database receives multiple SQL queries for execution.

SELECT accounts FROM user
WHERE login='sumit' AND
pass=''; drop table user – – 'AND
pin=231

After executing the first query, database would find injected/additional query. The result of the above query is to drop the user table and destroy important information.

4. Blind Injection:
In this technique, attacker asks true and false question to the server through the web page. If the injected statement evaluates to the true, the web site works normal. If the statement evaluates to false, although there is no descriptive error message, the page differs significantly from the normal functioning page.

5. Logical Incorrect Query Attack:
This type of attack gathers important information about the type and structure of the back end database in the web application. When the attacker uses this logical incorrect query, the application server displays error page, which can serve to expose sensitive information about the databases to the attacker.

SELECT accounts FROM users
WHERE login=''AND pass='' AND
pin= convert (int,(selecttop 1 name
from sysobjects where type='u'))

ISSN: 2278 – 1323

The select query extracts the user table and then converts this table name into an integer. Because of this illegal conversion, database gives an error. In this technique, attacker has two useful tricks. First, the attacker can see that the database is an SQL Server database, as the error message explicitly states this fact. Second, the error message reveals the value of the string that caused the type conversion to occur.

III. SYSTEM DESIGN

A. SYSTEM ARCHITECTURE:

The architecture of the system is illustrated in Figure. In a client server model, a reverse proxy server is placed, in between the client and the server. The presence of the proxy server is not known to the user. The sanitizing application is placed in the Reverse proxy server.

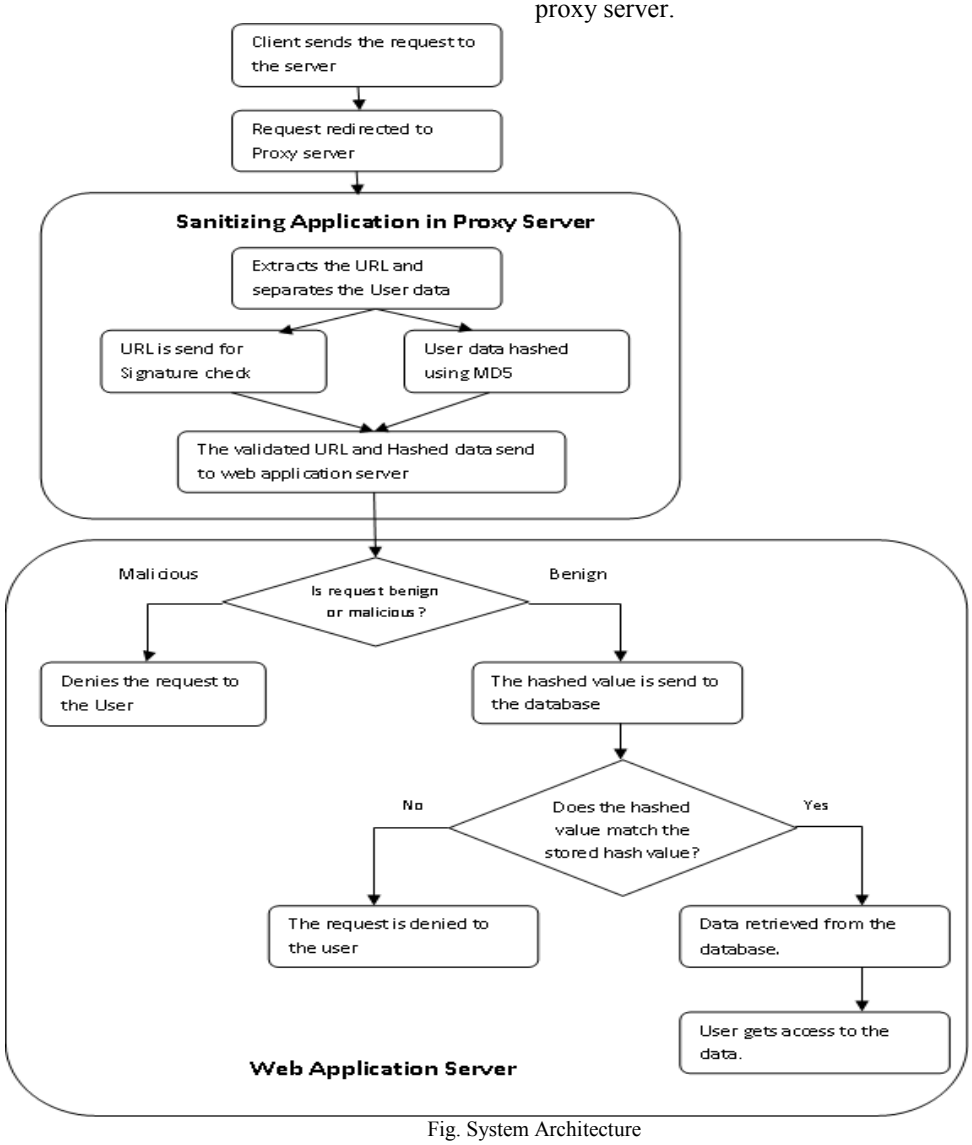

Fig. System Architecture

ISSN: 2278 – 1323

A reverse proxy is used to sanitize the request from the user. When the request becomes high, more reverse proxies can be used to handle the request. This enables the system to maintain a low response time, even at high load. The general work of the system is as follows:

1. the client sends the request to the server.
2. The request is redirected to the reverse proxy.
3. The sanitizing application in the proxy server extracts the URL from the HTTP and the user data from the SQL statement.
a. The URL is send to the signature check
b. The user data (Using prototype query model) is encrypted using the MD5 hash.
4. The sanitizing application sends the validated URL and hashed user
data to the web application in the server.
5. The filter in the server denies the request if the sanitizing application had marked the URL request malicious.
6. If the URL is found to be benign, then the hashed value is send to the database of the web application.
7. If the hashed user data matches the stored hash value in the database, then the data is retrieved and the user gains access to the account.
8. Else the user is denied access.

IV. THE SANITISING APPLICATION

A. DATA CLEANSING ALGORITHM (DC ALGORITHM):

Step 1:
Parse the user data into Tokens-*tok*;
While (not empty of *tok*)
Check if *tok* _ reserved SQL Keyword
Move *tok* to User data Array-*UDA*;
Step 2:
For (every data in *UDA*)
Convert to Corresponding MD5 and store in *MD5-UDA*.
Step3:
Extract the URL from HTTP; Parse the URL into Tokens-*toks;* While (not empty of *toks*)
Checkif (URL = Benign using the signature check) Set the *flag* to *continue*;
Else
Set the *flag* to *deny*;
Step 4:
Send the *MD5-UDA* and *flag* to Web Application Server.

B. DATA CLEASING ALGORITHM DETAILS EXTRACTING USER DATA FROM URL STATEMENTS

The SQL statement is extracted from the HTTP request and the query is tokenized. The tokenized query is then compared with the prototype document. A prototype document consists of all the SQL queries from the Web application. The query

tokens are transformed into XML format. The XSL" s pattern matching algorithm is used to find the prototype model corresponding to the received Query. This method has been adapted in the previous work COMPVAL. *XSL's Pattern Matching:* The query is first analyzed and tokenized as elements. The prototype document contains the query pertained to that particular application.

V. TECHNICAL SPECIFICATION

A . ADVANTAGES :

1. Lower cost of ownership.
2. Easier to deploy.
3. Detects attacks related to SQL injection.
4. Real time detection and quick response.
5. Detection of failed attacks.
6. This system does no change the source code application.
7. The detection and mitigation of attacks are fully automated.
8. By increasing the number of proxy servers the web application can handle any number of requests without obvious delay in time and still can protect the application from SQL injection attacks.
9. Does not require additional hardware.

REFERENCES

[1] David Litchfield, (2005) "Datamining with SQL Injection and Inference", Next Generation Security software Ltd., White Paper.
[2]Chip Andrews, "SQL Injection FAQs" http://www.sqlsecurity.com/FAQs/S QLInjectionFAQ/tabid/56/Default.aspx Computer Science & Information Technology (CS & IT) 143
[3] Y.Huang, F. Huang, T.Lin and C.Tsai, (2003) "Web Application Security Assessment by Fault Injection and Behavior Monitoring", Proc. International World Wide Web Conference " 03, pp. 148 -159.
[4] C.Gould, Z.Su and P.Devanbu, (2004) "JDBC Checker: A Static Analysis Tool for SQL/JDBC Application", Proc. International Conference on Software Engineering „04, pp.697-698.
[5] W. G. Halfond and A. Orso, (2005) "AMNESIA: Analysis and Monitoring for Neutralizing SQL Injection Attacks", Proc. ACM International Conference on Automated Software Engineering " 05.

ISSN: 2278 – 1323