IWNEST PUBLISHER

Journal of Industrial Engineering Research

(ISSN: 2077-4559)

Journal home page: http://www.iwnest.com/AACE/

Anti SQL IA Vaccine for Detection and Prevention of SQL Injection Attacks

Biji.K.P | Dr. J. Subash Chandra Bose

ME CSE and Head Department of Computer Science & Engineering Professional Group of Institutions Palladam, Tamilnadu, India

ARTICLE INFO

Article history:
Received 20 March 2015
Accepted 25 May 2015
Available online 5 June 2015

Keywords:

ABSTRACT

Anti SQL IA Vaccine is a new concept for Detection and Prevention of SQL Injection Attacks on development phase itself' which helps and manages the important private customer data in a secured manner by mirroring the important database structures into unique secure mirroring tables which is managed in a differently managed secure data management system which runs on same or different servers. An independently managed verification tool is used to inspect and search the possibility of an SQL injection in the source code of the webpages at the development phase itself. This plays an effective medium in the prevention and detection of SQL Injection, which is one of the major web attack terminology which is effectively utilized by various malwares and hackers to steal valuable data from websites of various organizations which manages their transactions through online and web databases. These are unique type of intrusion that takes advantage of improperly managed/amateur coding in the web applications. SQLIA allows intruders to inject SQL commands into access data's from the web forms to allow them to gain access to the data held within your database. In this paper we will discuss several types of SQLIAs, existing techniques and their drawbacks. Finally I have proposed a solution for SQLIA detection using data dictionary and prevention using the intrusion search along with SQL vaccine. I have implemented it using ASP.net with VB.net and SQL Server 2008, although this algorithm can be implemented in any language and for any database platform with minimal modifications.

To Cite This Article: Biji.K.P Dr. J. Subash Chandra Bose., Anti SQL IA Vaccine for Detection and Prevention of SQL Injection Attacks **J. Ind. Eng. Res.,** ***1(4),*** *113-117, 2015*

INTRODUCTION

SQL Injection Attacks (SQLIAs) have emerged as one of the most serious threats to the security of database-driven applications. In fact, the Open Web Application Security Project (OWASP), an international organization of web developers, has placed SQLIAs among the top ten dangerous vulnerabilities that a web application can have. Similarly, software companies such as Microsoft and SPI Dynamics have cited SQLIAs as one of the most critical vulnerabilities that software developers must beware. The problem with SQL injection is that a user input is used as part of the SQL statement. By using calculated and prepared statements you can manipulate the user input to be handled as the content of a parameter (and not as a part of the SQL command). Query parameters help to avoid this risk by separating literal values from the SQL syntax. SQL injection vulnerabilities can be particularly harmful because they allow an intruder to access the database that works as a backbone of an application. Using SQLIAs, an attacker may be able to read, modify, or even delete database information and sometime the schema. In many cases, this information is confidential or sensitive and its loss can lead to problems such as identity theft and fraud. The list of high-profile victims of SQLIAs includes Travelocity, FTD, Creditcards.com, Tower Records, Guess Inc., and the Recording Industry Association of America (RIAA).

A SQL Injection attack is a form of vulnerability attack that comes from user input that has not been verified to see whether it is valid. The prime objective is to make fool the database system into running malicious code that will dig sensitive user information. There are two main types of SQL IA attacks. First-order attacks are those when the intruder receives the desired result immediately, either by direct response from the web application they are in communication with or through some other response mechanism, such as email/sms. Second-order attacks are those when the attacker injects some form of malicious data that will reside in the database, but the same will not be immediately make problems. The vulnerability attack occurs when input from the user is used to directly build a vulnerable query to the database. If the input is not properly encoded and

Corresponding Author: Biji.K.P | Dr. J. Subash Chandra Bose, ME CSE and Head Department of Computer Science & Engineering Professional Group of Institutions Palladam, Tamilnadu, India

validated by the web application, the attacker can inject malicious input that is considered as additional commands by the database. Depending on the brutality of the vulnerability, the intruder can issue a wide range of dangerous SQL commands to the database. In Many interactive database-driven web applications use user data input to query their underlying databases, can be vulnerable to SQLIA. Informal surveys of database-driven web applications have shown that almost 97% are highly potentially vulnerable to SQLIA. Like most security vulnerabilities, SQLIAs can be prevented by using defensive coding. In practice however, this solution is very difficult to implement and enforce.

II. *Related Work:*

A. Mahima Srivastava (Dept. Of CSERKGITW Ghaziabad, India March 2014) [1] proposed a new approach using the concept of ASCII values and at the same time makes sure not even a single byte of additional storage is used. User can enter any type of data, any number of special characters. No more restrictions are imposed on the user. This approach can be implemented in three different phases. In first phase, discussing what user data should be stored in the database attached and how it can be stored. In second phase discussing how the data should be cross checked in the database and in third phase how data should be retrieved from database.

B .Lwin Khin Shar and Hee Beng Kuan Tan (March 2013) [2] proposed a method to overcome SQL Injection. Defensive coding practices will ensure preventing SQLIA and secure code but are time-consuming and labor-intensive. SQL injection defensive methods can be classified into three different types: defensive coding mechanism, SQLIV detection mechanism, and SQLIA runtime prevention mechanism. Vulnerability detection mechanism approaches can detect most if not all SQLIVs, but there is a possibility to generate many false alarms. Runtime prevention methods can effectively prevent SQLIAs, but they require more dynamic monitoring systems. The most effective strategy is effectively combining all three approaches.

C. Debabrata Kar and Suvasini Panigrahi proposed a lightweight tactical approach to prevent SQL Injection attacks by using a novel query transformation mechanism followed by hashing. The proposed query transformation scheme and hashing mechanism provides full protection from most varieties of SQL injection. This approach cannot able to prevent second order SQL injection attempts through the parameter values (mainly through string values) are removed during the transformation process. While prevention of SQL injection attacks is the primary aim in this work, it is worthwhile to mention that this approach can neither be applied to prevent the XSS attacks.

D. Dr.M.Amutha Prabakar, M.KarthiKeyan, Prof.K. Marimuthu, proposed a detection system and prevention technique for preventing SQL Injection Attack (SQLIA) using Aho–Corasick pattern matching algorithm. This scheme is evaluated by using sample code of well-known attack patterns. Initial stage evaluation shows that the proposed scheme is produce not false positive and false negative.

E. Types Of SQL Injection Attacks:

Once attackers traced and identified an input source that can be used to exploit SQLIA vulnerability, there are many different types of attack techniques that they can employ. Depending on the type and extent of the vulnerability, the results of these attacks can include hack the database, gathering information about the tables in the database schema, establishing covert channels, and open-ended injection of virtually any SQL command. Malicious SQL statements can be introduced into a vulnerable application using many different input mechanisms.

SQL injections can be implemented in the following ways:

i) Tautology
ii) Illegal/logically incorrect queries
iii) End of line comment
iv)Timing attack
v) Union queries
vi) Blind SQL injection attacks
vii) Piggy backed queries

III. Proposed Algorithm:

SQL Injections are intrusions by which an attacker changes the structure of the original SQL query by injecting SQL code in the input fields of the web form in order to gain unauthorized access to the database. [1] Once an attacker gains access through internet then he can perform so many functions which will be a very big danger to our databases. These functions might include for retrieving the passwords of other users, deleting information from the user tables or even deleting the entire tables or database, updating someone's valuable information, etc.

Journal of Industrial Engineering Research, 1(4) July 2015, Pages: 113-117

The proposed system is a simple and effective method to accurately detect and prevent SQLIAs by using a Combination of DDL & DML Mapping along with Vectorization of SQL Queries. In DDL & DML Mapping we generate a partial Sparse Image of the Structure and Data contents of the Database to a Mirror Database which will be stored in parallel to another database if possible in a different server. Along with the Sparse Image the Vectorization of the SQL Queries is also store in tables of the Mirror Database, to include different syntax, we resolve the parse tree of different generated queries. As a result of the output of the different generated queries, we can monitor the detection of abnormalities among the queries within our database structure; we are in forward for a resemblance monitoring for the space of controlled objects, i.e. the space of valid SQL parse tree structure. Thus, we strive with the problem of having to generate a comparison utility for matching trees.

A. Detection and Prevention of SQL Injection Attacks:

AMNESIA, (**A**nalysis for **M**onitoring and **NE**utralizing **S**QL **I**njection **A**ttacks) is a fully-automated and general technique for detecting and preventing all types of SQLIAs. The approach works by combining static analysis and runtime monitoring. Our two key insights behind the approach are that (1) the information needed to predict the possible structure of all legitimate queries generated by a web application is contained within the application's code, and (2) an SQLIA, by injecting additional SQL statements into a query, would violate that structure. In its static part, our technique uses program analysis to automatically build a model of the legitimate queries that could be generated by the application. In its dynamic part, our technique monitors the dynamically generated queries at runtime and checks them for compliance with the statically-generated model. Queries that violate the model represent potential SQLIAs and are reported and prevented from executing on the database. The technique consists of four main steps.

B .The AMNESIA Approach:

Identify hotspots:

Scan the application code to identify *hotspots*—points in the application code that issue SQL queries to the underlying database.

Build SQL-query models:

For each hotspot, build a model that represents all the possible SQL queries that may be generated at that hotspot. A *SQL-query model* is a non-deterministic finite-state automaton in which the transition labels consist of SQL tokens (SQL keywords and operators), delimiters, and placeholders for string values.

Instrument Application:

At each hotspot in the application, add calls to the runtime monitor.

Runtime monitoring:

At runtime, check the dynamically-generated queries against the SQL-query model and reject and report queries that violate the model.

C. Algorithm for SQL Injection Detection:

Step 1: Gathering the information about the Database Schema.
Step 2: Store the Schema values into the Sparse Tables.
Step 3: Gather the Output of Randomly generated SQL Queries from various tables.
Step 4: Store the Queries and Output into Tables of Mirror Database.
Step 5: Search for the different SQL queries which is coded inside the web forms.
Step 6: Store the SQL Query detected along with the position information.
Step 7: Confirm the Gathered information.

D) Runtime SQLIA prevention:

Researchers have proved that query injection can't be applied without using space, single quotes or double dashes (--). Researchers have developed tools and techniques that could prevent all SQLIAs by checking actual runtime against legitimate queries.

Static Analysis:

AMNESIA (Analysis for Monitoring and NEutralizing Sql Injection Attacks) uses static analysis to deduce valid queries that might appear at each database access point in Web programs via isolation of tainted and untainted data. Another runtime SQLIA prevention technique uses a query learning approach similar to AMNESIA, but, instead of targeting query statements in a server program, it targets stored procedures in a database and also crosscheck with the data dictionary generated.

Journal of Industrial Engineering Research, 1(4) July 2015, Pages: 113-117

Dynamic Analysis:

Statically inferred legitimate query structures might not be accurate, and attackers could exploit this weakness to conduct SQLIAs. Researchers have thus proposed dynamic-analysis-based approaches to provide more accuracy. SQL Check tracks tainted data at runtime by marking it with Meta characters and store the SQL queries inside the Data dictionary. When a Web application invokes a query, SQL Checker learns the query's legitimate structure by verifying it with the data dictionary.

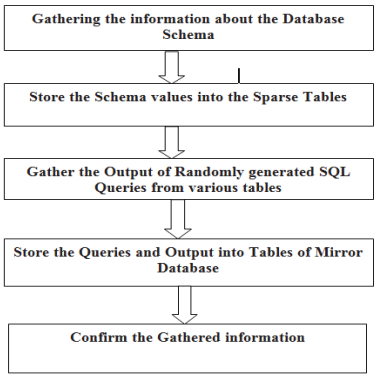

Fig. SQL Injection detection.

1V. Result And Evaluation:

The proposed algorithm for the detection of SQL Injection can be implemented successfully on real web applications. This paper compares the detection rate of the proposed method with other researches under the same conditions. It also compares other major detection and prevention methods of particular attack forms. It detects attacks by comparing the structure and the grammar of the query with the available methods along with the data dictionary. If the dynamically generated query has a different structure or if it uses different grammar, it will be detected. The algorithm proposed in this paper does not use complex analysis methods such as Parse trees. It uses a very simple method which compares the queries after the removal of the attribute values. Therefore, it can be implemented into any type of DBMS.

Table 1: Structure of legitimate query.

Sl.No	File Name	Query
1	Login.vb	Select username, password from tbl_Login
2	Home.vb	Select category from tbl_category
3	Home.vb	Select Date from tbl_datareport where category_id='2'
4	Vouchure.vb	Select * from tbl_vouchure where category_id='2' and date='date'

V. Conclusion:

This paper proposes a new SQL injection attack detection method that utilizes both Static and Dynamic Analysis along with a data dictionary in parallel for data verification. SQL injection is a common technique hackers employ to attack underlying databases. These attacks reshape the SQL queries, thus altering the behavior of the program for the benefit of the hacker. In this paper, we present a fully automated technique for detecting, preventing and reporting SQLIA incidents in databases. In DDL & DML Mapping we generate a Sparse Image of the Structure and Data contents of the Database to a Mirror Database which will be stored in parallel. The proposed algorithm for the detection of SQL Injection can be implemented on real web applications. It can store total number of parameters and queries in web applications and make a list to compare with the real time generated value of parameter. Also, it can profile legitimate dynamic query generated by normal users between the web application and the database server and compare it with the dynamically generated query to detect attacks.

REFERENCES

[1] Mahima Srivastava, 2014. "Algorithm to Prevent Back End Database against SQL Injection Attacks", Published in: Computing for Sustainable Global Development (INDIACom), International Conference. 978-93-80544-12-0/14/$31.00_c IEEE, Date of Conference: 5-7, Page(s): 754 - 757

[2] Lwin Khin Shar and Hee Beng Kuan Tan, Nanyang, 2013. "Defeating SQL Injection", 0018-9162/13/$31.00 © IEEE Published by the IEEE Computer Society.

[3] Dr Amutha Prabakar, M. KarthiKeyan, Prof. K. Marimuthu, 2013. "An efficient technique for preventing sql injection attack using pattern Matching algorithm". IEEE International Conference on.M. Emerging Trends in Computing, Communication and Nanotechnology (ICECCN 2013).

[4] Rajesh, M., Lomte1, Prof. S.A. Bhura2, 2013. "Survey of Different Web Application Attacks & Its Preventive Measures", IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p- ISSN: 2278-8727. 14(5): 46-51. www.iosrjournals.org
[5] Debabrata Kar, Suvasini Panigrahi, 2013. "Prevention of SQL Injection Attack Using Query Transformation and Hashing", 3rd IEEE International Advance Computing Conference (IACC), 1317-1323.
[6] Anjugam, S., A. Murugan, 2014. "Efficient Method for Preventing SQL Injection Attacks on Web Applications Using Encryption and Tokenization", ©, IJARCSSE All Rights Reserved Page, 173-4(4). ISSN: 2277 128X, International Journal of Advanced Research in Computer Science and Software Engineering.
[7] Nataraj, J. and P. Muralikrishnan, "DVGAR: Distraction of Web Vulnerabilities to Provide Grasp Accessing Capability of The Web Resources", | ISSN: 2321-9939, International Journal Of Engineering Development And Research | IJEDR
[8] Sadeghian, M., Zamani, S. Ibrahim, 2013. "SQL Injection Is Still Alive: A Study on SQL Injection Signature Evasion Techniques", Date of Conference: 4-6. Publisher : IEEE.
[9] Joshi, V., Geetha, 2014. "SQL Injection detection using machine learning", Date of Conference: 10-11. Publisher : IEEE.
[10] http://www.4 halfond07springer William G.J. Halfond and Alessandro Orso Georgia Institute of Technology .
[11] Shruti Bangre, Alka Jaiswal, 2012. SQL Injection Detection and Prevention Using Input Filter Technique. International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, 1-2.
[12] Mihir Gandhi, Jwalant Baria, 2013. "SQL INJECTION Attacks in Web Application", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, 2(6).