SQLIAD – A Hazard to Web Applications

Fyrooz Nidhal, Hiba Naser R N, Nahida Abdul Latheef, Shahsitha Siddique V, Yomna Kalam A V
Mentor – Neethu Prabhakaran P

Department of Computer Science, IES College of Engineering, Calicut University, Thrissur, Kerala, India

Abstract – SQLIAs has been now a major threat to the growing popularity of web application. The main target of this attack is the database. This allows attackers to obtain unauthorized access to database. In this paper we survey different types of SQLIAs and prevention methods. To address this problem we propose a mixed approach for prevention of SQLIAs. This paper ensures that the untrusted data are validated against a list of allowable values. Least privilege principle is applied to SQL account used by web application. We avoid query concatenation at almost all costs and use parameterized queries wherever possible.

Keywords — SQLIA, Database, Vulnerabilities, attacker, web security, detection, prevention, web application.

I. INTRODUCTION

SQL injection attack are very effective system attack that can be used to gain or manipulate data in data driven systems, which is a common problem for web application data published on the internet. SQLIAs are simple to be learnt and simple to be executed. So they can be executed as inexperienced hackers. So they have been many researches that have developed various methods to detect and prevent SQLIAs. Each of these methods covers an objective or set of objectives related to this type of attack but there is no method that can cover the whole system from being attacked by SQL injection.

The risk of SQLIAs is that when they are performed through the victim back end system, they will be running with the same privileges that the system have on the database, that means if the system have the power user or administrator permission then the injection code could be executed on the disaster effect on the victim machine. This paper proposes a system of mixed approach for preventing this kind of SQL injection attacks by data validation, principle of least privilege, avoiding query concatenation and the use of parametrized queries.

II. LITERATURE SURVEY

A. CSR Scanner

The tool mainly used for web application developer. CSR Scanner checks vulnerability of SQL injection using the following steps: (1) Crawling the whole web application. (2) Scanning for vulnerable points. (3) Generate attack and report.

For checking a web application have a home page as input designed like a tree for visiting all pages. Find each vulnerable spots on all pages and scan that spots then generate attack on that vulnerable points and report the result of attack. Attack generation consist of payload setup and generating attacks. In payload setup attack is generated base on existing SQL injection attacks and created list of attacks which is used by attackers. Depending on the database the response of attack will be different. Combining attack payload with original query of client performed in Generating attack phase. The Crawling, scanning and attacks are implemented using several functions. By comparing CSR Scanner with other vulnerability scanners, it gives good and full coverage of any web application. Thus can visit all pages for checking vulnerabilities of SQL injection.

B. URL Filter

To prevent SQLIAs, we will block the request which include SQL commands send to the application server. We should make sure that queries and statements are not blocked, at the time of blocking SQL commands. Client sends request that pass through URL filter and processed by application server. If proposed filter contain combinations of some other character URL filter prohibits SQL Meta character in the input. In server side using a browser, user invokes services of application server. User sends input to application server as parameter string and this uses to make SQL query, to insert information to database or delete from it. Between a user and application server has a Meta filter which passes parser input into SQL Meta character tokens. The injected input is considered as an attack if input from user contain any type of attack else application server will process the input directly. Since Meta filter checks the presence of SQL expressions, URL filter will not block legal inputs.

C. Proxy Server model

Proxy server is used between two communicating devices which avoid SQL injections and filtering of their attempts. SQL query commands are analysed from a list of common SQL injection commands. To check current patterns of SQL statements, create a parser. Proxy server is created to prevent SQLIA to a database. Now we can prove SQL injection can be prevented using this proxy server filter.

From input parameters queries are generated in web application and send to database server using a TDS
proxy. TDS proxy parse incoming request and clean the query and send to database server. The database sends a response of processed request through TDS proxy. For getting correct view of client TDS proxy sends that respond to the client. The system receives data from client and analyze the data and if data contains a SQL query, it will be logged and filtered. An attack is logged if filter process any kind of attack and wrong query to the destination and response back to the client. If filter does not contain an attack, query send to database and response to client. Data directly passed to destination if sending data does not contain a query.

D. Crypto graphical Hash Function

Defence mechanism and username and password validation using crypto graphical hash function is proposed in this model. A crypto graphical hash function is a hash function which is considered practically impossible to invert, that is to recreate the input data from its hash value alone. The input data is often called the message and the hash value is often called the message digest. The ideal crypto graphical hash function has four main properties: (1) It is easy to compute the hash cache for any given message (2) It is infeasible to generate a message from its hash. (3) It is infeasible to modify a message without changing the hash. (4) It is infeasible to find two different messages with the same hash.

This model has two aspects: (1) SQLIA prevention using data validation. (2) SQLIA prevention using crypto graphical hash function. Following approaches are proposed: (a) escape single quotes (b) reject input that is known to be bad (c) accept only input that is known to be good. In the second aspect, we provide a hash function mechanism. We need to add two additional attributes in the login database in this mechanism, one attribute for hash value for username and another for password. When an admin creates an account and assign password the hash value is automatically generated by using hash function algorithm and stored in the database along with the login information of the user which is stored in the database in encrypted form.

This model filters or validate the entered data by a user on a webpage hence the chance of vulnerability can be avoided.

E. Client side SQLIA detection

The client side is the starting point of SQLIA. In this model attacks are detected early at the browser side and thus prevents the forwarding of the malicious inputs to the server side for further processing. This model uses the concept of measuring deviation of shadow queries with dynamic query based on conditional entropy metrics. A shadow query is an identical query of the original query except column name. Table name and input variable are replaced with arbitrary symbolic values. Three PHP application containing SQL vulnerabilities are used to evaluate this approach. This evaluation result indicates the possibility of detecting SQLIA early at the client side.

SQLIA detection approach is mainly based on SQLIA detection approach and evaluation. SQLIA detection approach consist of (a) shadow query generation and form modification (b) conditional entropy calculation. (c) SQLIA detection with conditional entropy. Conditional entropy is a measurement technique in which we are measuring the information contents of SQL queries as part of determining the deviation between shadow query and an original query with malicious inputs. Evaluation of the proposed conditional entropy is done using three open source PHP applications available from sourceforge.net.

This approach detects malicious inputs causing SQLIA at the client side early as well as relieving the server side for additional checking and acts as a complimentary solution to other existing approaches.

III. PROPOSED SYSTEM

Over the years the SQL Injection threat has grown to the point where now we are seeing weaponed SQL Injection attacks perpetrated by state actors. Organizations are continually being breached via SQL Injection attacks that slip seamlessly through the firewall over port 80 for HTTP or 443 for SSL to soft internal networks and vulnerable databases. Objective of this research paper is to see and analyse threat of SQL injection in our web applications. There are many methods of preventing and avoiding this attack but here we will see latest and most effective method to counter this SQLIA.

The gaining popularity of SQL Injection attacks is attributed to the fact that many Web Applications do not perform any validations on the data input by the user. This enhances the chances of the attacker to gain unauthorized access to the database of the application. We will be developing a specific mechanism in .NET in order to prevent any of such SQL injection attacks during user interactions through the forms. A stringent check against such possible insecure keywords and data patterns will be implemented. Also, parameterized dynamic SQL approach in ADO.NET will be adapted in order to prevent attack attempts, where end user can even enter suspicious keywords, but the same will be encoded and send to database as a field value instead of special instructions. We will also study the functional requirements of the project and identify the areas where SQLIA prevention fixes can be made. An agile based scrum methodology will be followed for this engineering activity.

We propose a system of mixed approach for preventing SQL injection attacks. Our approach on SQL injections are as follows
1. The untrusted data needs to be validated against a white list of allowable values. The .NET code should define a regex of the characters it allows and that probably shouldn’t include equal signs and semicolons.

2. The principle of least privilege should be applied to the SQL account used by the web app. While it won’t stop the 1=1 sort of scenario, least privilege would mean thinks like not allowing the account to be used by the public users to write to the Users in Role table. Give it only the bare minimum it needs.

3. Avoid query concatenation at almost all costs. Even though there will always be edge cases, but concatenating untrusted data directly into the query itself is the last thing you want to do anywhere. Most folks already know this is the case in the web tier, don’t forget it applies to stored procedures as well.

4. Use of parameterized queries (prepared statements) wherever possible.

   The most important advantage of the new approach against existing analogous mechanisms are that, it prevents all forms of SQL injection attacks and also the current technique doesn’t allow the user to access database directly in database server.

IV. IMPLEMENTATION

Fig.1 Overview design of SQLIAD

Fig.2 System Architecture design of SQLIAD

V. Conclusion

SQL injections are attacks by which an attacker can alter the structure of original SQL query. This is performed usually by injecting SQL codes into the web form at its input field. This allows the attacker to gain unauthorized access to the database. We have surveyed different types of SQLI and have proposed techniques for prevention and detection and thus have provided a better solution to this problem. This paper has developed mixed approach for preventing SQLIAs. The proposed system is more advantageous than the existing system as it prevents all forms of SQLIA and secondly it allows the user access database directly in the database server. The future work is to implement this technique and thus ensure a secure database by detecting SQLIA and provide a better performance for web application.

ACKNOWLEDGMENT

We are grateful to the head of the department of computer science and engineering, Dr. S Brilly Sangeetha for her support. We also acknowledge Miss Neethu Prabhakaran P for her guidance as our mentor.

REFERENCES

[1] Evaluation of SQL Injection Detection and Prevention Techniques By Atefeh Tajpour , Centre for Advanced Software Engineering (CASE) ,University Technology Malaysia ,Kuala Lumpur, Malaysia ; Mohammad JorJor zade Shooshhtari, Centre for Advanced Software Engineering (CASE) ,University Technology Malaysia, Kuala Lumpur, Malaysia


