SQL INJECTION ATTACK, STILL AN UNADDRESSED ISSUE WITH DYNAMIC WEB APPLICATIONS

Dr. AbdulrahmanA. Alghamdi
College of Computing and Information Technology
Shaqra University, Shaqra, KSA
alghamdia@su.edu.sa

Bilal Ahmad
College of Computing and Information Technology,
Shaqra University, Shaqra, KSA
bilal@su.edu.sa

Mohammad Imran
Department of Computer Science, College of Science and Humanities,
Shaqra University, Al-Dawadmi, KSA
mimran@su.edu.sa

Abstract

We all know that most of the dynamic web applications are developed using three tier architecture by dividing it into three layers, but among all security threats, SQLIA is foremost security threat for any dynamic web applications. Using this mechanism, attacker can bypass authenticated & secure functionality of a web application and can inject a special crafted SQL query into application layer to get unauthorized & confidential information from database server. The main approach of this paper recommends a mechanism of using a combination of client & server site scripting to prevent SQLIA.A client-side logical unit coded in client-side scripting language detects most of the special characters which a hacker uses to make a SQLIA. When Next filtered user's input parameters go to the web server, here it again goes through a testing process by a functional unit which is coded using server-side scripting language. The remaining possibilities of SQLIA can stop here; therefore a dual testing approach in the application layer can decrease the probability of SQLIA at development phase of a web application.

Keywords- SQL, SQL Injection, SQLIA, SQL Injection Vulnerability.

I. INTRODUCTION

Today is the era of information technology. Information is playing widely significant role in every aspect of the current modern life. Developers must learn how to secure their applications against SQL injection vulnerability(Sadeghian, Zamani, & Abdullah, 2013) because SQL injection is one of the highest rank security vulnerabilities of dynamic web applications(Patil & Bamnote, 2014; Sadeghian, Zamani, & Ibrahim, 2013).At present web applications can be secured from this attack using various prevention techniques for example parameterized dynamic queries(Sadeghian, Zamani, & Abdullah, 2013), input parameter filtration(Bangre & Jaiswal, 2012) and many more . SQL Injection does not only threat web applications but also all types of applications backed with database, using SQL(Bangre & Jaiswal). Database is considered the most precious assets in any application, and SQL injection is one of the most dangerous threats to the databases (Halfond, Viegas, & Orso, 2006). Serious security threats are posed to application such as allowing the attackers to obtain unauthorized access to the database (Halfond et al., 2006). This paper recommends that a secure coding style at the development phase and secure a web application from SQLIA. Rest of this paper is organized as:-

(1) Section II explains a three tier web application architecture and Structure query Language processing to connection application with the database.

(2) Section III explains the SQLIA.

(3) Section IV explains a literature survey.

(4) Section V explains the proposed mechanism and Section VI ends with the conclusion and future work.

II. Three tier Web Application Architecture and SQL Processing

A. *Three tier Web Application Architecture*

Most of the database driven web applications available on the web implement three tier architecture, containing of a database tier, controller tier and presentation tier(Selfa, Carrillo, & Del Rocio Boone, 2006).

1) Database Tier

The bottom layer of database management system manages storage, retrieval of data. Some other tasks like a database security, data integrity, user privileges & roles and simultaneous access are also managed by this layer. The three tier architecture facilitates for the developer to maintain and evaluate the web application easily because the application is divided into three different layers.

2) Controller Tier or Logic Tier

It's the middle layer that contains most of the application logic. It controls all other layers by processing the inputs received from presentation& database tiers. It consists in the web server, web scripting language (WSL) and the scripting language engine. When Web server handles the HTTP requests and generates responses, scripting language engine is used to execute logic interacting with the database. The resulted data is send to the presentation tier. The server side scripting language which is used to code in controller tier is ASP, JSP, and PHP etc.

3) Controller Tier or Logic Tier

It's a top level layer which receives user's input and displays information related to services available on a web application. This tier communicates with the user by accepting inputs and responding results in the form of HTML, Java Script, and Flash etc.

B. *SQL Processing.*

Most of the web applications are using RDBMS (Relational Database Management System) as their backend process. A Special designed language SQL (Structure Query Language) is used to interact with database system, SQL is an ANSI (American National standard Institute) standard language for relational databases. The SQL statements execute within the database in the form of a query and send the result to the user. Database manipulations (updating of data, altering of data, inserting & fetching of data and modification of data etc.) into RDBMS is achieved by the execution of SQL statements within the database system. It's a user friendly language which allows retrieving the information from database without having much more knowledge of, how it works(Sadeghian, Zamani, & Abdullah, 2013). A SQL injection is made by appending malicious strings into legitimate SQL query(Kiani, Clark, & Mohay, 2008).The below diagram (Figure 1) shows the SQL query processing in the context of simple web application architecture.

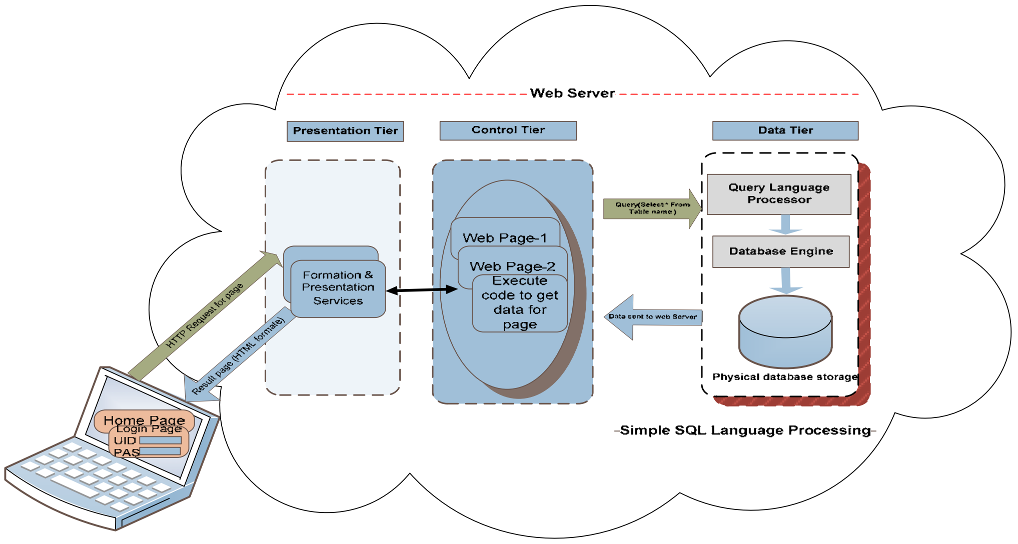
Web Server
Presentation Tier
Control Tier
Data Tier
Formation & Presentation Services
Web Page-1
Web Page-2
Execute code to get data for page
Query(Select* From Table name)
Query Language Processor
Database Engine
Physical database storage
Data sent to web Server
HTTP Request for page
Result page (HTML formate)
Home Page
Login Page
UID
PAS
Simple SQL Language Processing

Figure 1. SQL Processing in web application architecture

III. SQL Injection

A web application suffers from many possible attacks by the hackers like cross scripting, cross site request forgery and SQL attack etc. SQL attack is widely used attack (Patil & Bamnote, 2014).Therefore ,SQLIA is one of the major security issue in web applications(Pinzo et al., 2010). It becomes more serious issue because it attacks the backend database of a web application, which reveals private & confidential information to the hackers(Kiani et al., 2008). Since the SQLIA occur between Presentation & Controller layer of the application, a weak coding style could increase its probability. Considering the following example,

SELECT * FROM user_table WHERE user_id = '$User_ID' AND Pass='$pass';

The above query connects to the database and pulls up the records from User table.

SQL injection attack occurs when a malicious code is inserted by an attacker who exploits weakness in SQL string (Anley, 2002). "The crucial form of SQL injection comprises of direct insertion of code (malicious code) into user-input variables which are concatenated with SQL commands and executed" (Haixia & Zhihong, 2009) . It is speculated that SQL code looks like this:

SELECT * FROM user_table WHERE user_id = '$User_ID'OR 1=1 - - AND Pass='$pass';

After - - (double dash) the rest of the sentence becomes as comment and 1=1 is a true condition that means authentication is bypassed. It is clear from the above example, attacker makes SQLIA by appending some special character to the user's input string. A number of techniques are vailable to stop SQLIA but none of them could completely stop it.However a good coding approach could decrease its possibilities. The below diagram(figure 2) explains this process:

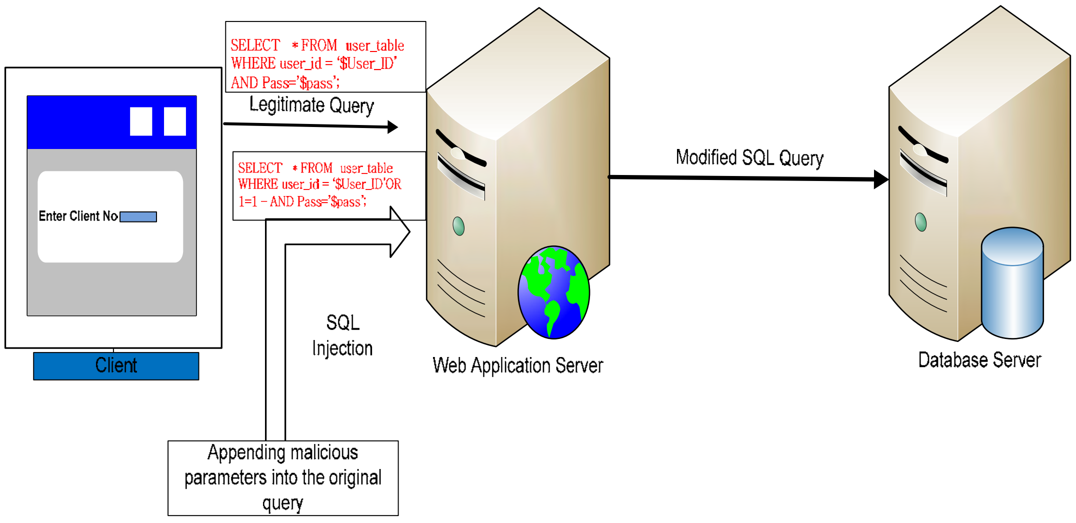
SELECT * FROM user_table WHERE user_id = '$User_ID' AND Pass='$pass';
Legitimate Query
SELECT * FROM user_table WHERE user_id = '$User_ID'OR 1=1 – AND Pass='$pass';
Enter Client No
Client
SQL Injection
Appending malicious parameters into the original query
Web Application Server
Modified SQL Query
Database Server

Figure2. SQL Injection Attack Mechanism

IV. Review of Literature

A literature survey was done to prevent & detect SQL injection attack.Here we present an overview of various prevention techniques which we have gone through in our survey.

A. *Input parameter filtration.*

Due to weak input validation logics into the controller tier of web application, hackers get the chance to attack web application. SQLIA is most famous database attack. Bangre & Jaiswal proposed input parameter filtration technique. In this technique a function is used to compare dynamic generated input parameter with already stored static parameter within the function logic. If both the parameters are not same then user's attempt is recorded and blocked. Here author considers that in SQLIA, attacker only uses some special character like space, single quote and double dash etc. Hence this technique is specific to detect some of the attacks only (Bangre & Jaiswal, 2012).

B. *Black box testing*

Black box testing application is used to protect web application from hacking, threats, XSS, SQLIA and CSRF etc. Mostly used security threat by the hacker is SQLIA. Identifying the vulnerable program code within web applications is too difficult as in a methodical or automatic manner. The writer suggested a set of recommendation which increases the rate of detection. However some drawbacks are found in black box testing technique(Huang, Huang, Lin, & Tsai, 2003).

C. *Hidden web crawling*

SQLIAV is the most widely used web application attacking issue in World Wide Web. A number of vulnerable scanners are formed to detect this attack. Here an approach exists based on hidden web crawling (scanning) for the achievement of destination .This tool is compared with other traditional scanner with the help of testing on a public website. The outcome justifies that the proposed tool is fine over the traditional web scanner(Xin, Luhua, Gengyu, Dongmei, & Yixian, 2010).

D. *Static Analysis Framework (SAFELI)*

It is the static analysis tool which detects SQLIA at compile time of an ASP.Net Web application. SAFELI contains five major components:

1) Microsoft Intermediate Language (Byte code)
2) Symbolic execution engine
3) Library of attack patterns
4) Constraint server
5) Test case generator

Based on the practical results it's a better tool to detect SQLIA than black box testing.

SAFELI checks MSIL code to detect the possible SQLI attacks points with the help of symbolic execution engine .The constraint solver then find out the input string constraint associated with each input string.

Test case generator traces the attacks by analyzing the response from the server after making HTML requests(Xiang et al., 2007).

E. *Network Vulnerability Scanner*

NVS scans all the web pages of a web application and finds out the SQLI vulnerable pages.

This is very fast tool which generates report within 0.01hours. NVS model consists of three parts.

1) *Crawling* -It scans whole web application to find out vulnerable pages.
2) *Attack Simulation*-It has three parts
   - *Payload Setup*-It consists a list of possible SQLIAs
   - *Generating Attack & Response Analysis*–Based on payload list a number of simulation attack performs and SQLIA patterns are studied after analyzing the responses.
   - *Report Generation*-An automatic report is generated after analyzing the SQLI patterns and URLS associated with these vulnerable pages are also stored into the table.
3) *Network Setup*- An ad-hoc network is considered to connect multiple systems. Each system is assigned a Remote Method invocation server to make attacks on systems using the URLs from table. If any vulnerability is found the report will be updated(Singh & Roy, 2012).

F. *SQLIA Prevention Using Stored Procedures.*

Due to insecure coding practice attack rate of web application is increased. SQLIA is one of the widely used attacks among them. Stored procedures are the most important database subroutines to stop SQLIA due to the benefits like encapsulation, strong input validation, faster execution and exception handling.Muthuprasanna & Kothari proposed a mechanism to stop SQLIA by performing in the application's code after its static analysis with runtime validation(Ke, Muthuprasanna, & Kothari, 2006).

G. *Using Parameterized Query.*

SQLIA is the most important security threat for dynamic web applications. Sadeghian, Zamani, & Abdullah described different type of SQIAs and eventually recommended that using parameterized query is best coding practice to avoid SQLIA during application development phase. Parameterized query compile within the databases without considering user's input, however place holders are created to hold input parameters .These parameterized queries again compile within the database systems considering users input. This approach opposes the string concatenation to the user's input string which may cause the SQLIA(Sadeghian, Zamani, & Abdullah, 2013).

H. *Smart database configuration.*

A smart database configuration includes user privileges and roles. Configuring database system with different privileges & roles may decrease the probability of SQLIAs. A user privilege is a right to run a particular SQL statement within the database system, or accessing a particular database object belonging to other users. Role is a way to assign multiple privileges to a single user .Database administrator has the rights to create the roles of the users(Sadeghian, Zamani, & Ibrahim, 2013).

I. *Database Security Testing*

Web database security is a serious issue in web applications .SQLIA is one of the prominent vulnerability .Author proposed a technique to increase database security against SQLIAs.

Firstly it finds SQLIA points by scanning source code of web application consequently designs SQLIA rules library, Then it makes simulation attack to the running application to find out the vulnerability of database. At the end it will make a report and make database secure against these attacks(Haixia & Zhihong, 2009).

V. RECOMMENDEDAPPROACH TO AVOID SQLIA

The proposed approach to prevent SQLIA is based on client & server-side scripting of a three tier web applications. Usually dynamic web application are developed by using html, client site scripting languages (java script, VB script), server site scripting languages (Asp, JSP, PHP) and database servers etc. When a user request for a page which is handled by the web server and the request page is passed through its process (page lifecycle at server), during page lifecycle the server site script executes at server and the last event of this life cycle is to render the page and send it back to the client. This rendered page is the combination of HTML, client site script language and Flash etc. This approach is to educate the web developers to stop SQL injection attack by making prevention logics in both client side and server site scripting because this vulnerability may occur due to the weak coding style at development phase(Bangre & Jaiswal, 2012). As showing Figure 3Using Clint site scripting, developers may decrease the probability of SQLIA by making strong Input validation logics. Here developer may send only filtered input string to the server. In the next step, remaining possibilities of SQLIA can be stopped by making strong right access/user permissions logical units (functions) in the business tier of web application architecture. As a result this mechanism, client & server side scripting is a best approach to develop a secure web application.

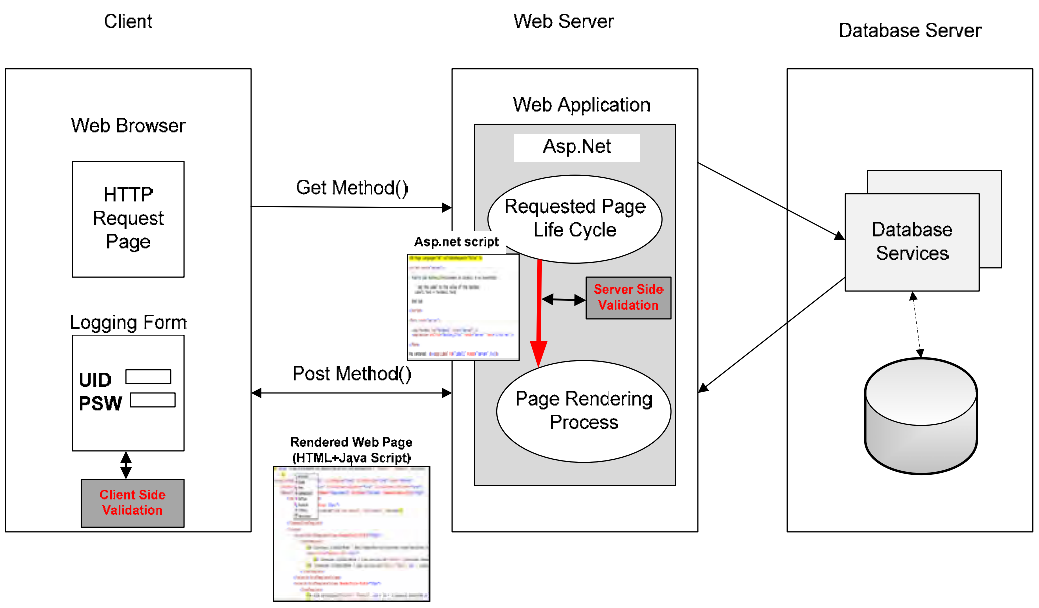

Figure3. CLIENT & SERVER SIDE PREVENTION

VI. CONCLUSION

This paper recommends a coding mechanism at development phase which comprises client-side scripting and server-side scripting. A combination of secure coding style using client-side scripting & server-side scripting language can decrease the probability of SQLIA.

This approach provides a dual validation of web application which means validation at client side as well as server site to stop SQLIA. Although a number of tools are available to protect web application against this vulnerability but none of them is completely protected. However we rely on secure coding style at development phase because SQL injection attack occurs due to improper validation or authentication weakness of the code of the web applications. Our future work is to design a tool which will detect not only SQLIA but also other web application vulnerabilities.

REFERENCES

[1] Anley, C. (2002). Advanced SQL injection in SQL server applications.Bangre, S., & Jaiswal, A. (2012). SQL Injection Detection and Prevention Using Input Filter Technique. International Journal of Recent Technology and Engineering (IJRTE), 1(2).
[2] Haixia, Y., & Zhihong, N. (2009). A database security testing scheme of web application. Paper presented at the Computer Science & Education, 2009. ICCSE'09. 4th International Conference on.
[3] Halfond, W., Viegas, J., & Orso, A. (2006). A classification of SQL-injection attacks and countermeasures. Paper presented at the Proceedings of the IEEE International Symposium on Secure Software Engineering, Arlington, VA, USA.
[4] Huang, Y.-W., Huang, S.-K., Lin, T.-P., & Tsai, C.-H. (2003). Web application security assessment by fault injection and behavior monitoring. Paper presented at the Proceedings of the 12th international conference on World Wide Web, Budapest, Hungary.
[5] Ke, W., Muthuprasanna, M., & Kothari, S. (2006, 18-21 April 2006). Preventing SQL injection attacks in stored procedures. Paper presented at the Software Engineering Conference, 2006. Australian.
[6] Kiani, M., Clark, A., & Mohay, G. (2008, 4-7 March 2008). Evaluation of Anomaly Based Character Distribution Models in the Detection of SQL Injection Attacks. Paper presented at the Availability, Reliability and Security, 2008. ARES 08. Third International Conference on.

[7] Patil, V. S., & Bamnote, G. (2014). An Overview to SQL Injection Attacks and its Countermeasures. International Journal of Innovative Research and Development, 3(1).8.Pinzo, x, n, C., De Paz, J. F., Bajo, J., Herrero, A., & Corchado, E. (2010, 23-25 Aug. 2010). AIIDA-SQL: An Adaptive Intelligent Intrusion Detector Agent for detecting SQL Injection attacks. Paper presented at the Hybrid Intelligent Systems (HIS), 2010 10th International Conference on.

[8] Sadeghian, A., Zamani, M., & Abdullah, S. M. (2013). A Taxonomy of SQL Injection Attacks. Paper presented at the Informatics and Creative Multimedia (ICICM), 2013 International Conference on.

[9] Sadeghian, A., Zamani, M., & Ibrahim, S. (2013). SQL Injection Is Still Alive: A Study on SQL Injection Signature Evasion Techniques. Paper presented at the Informatics and Creative Multimedia (ICICM), 2013 International Conference on.

[10] Selfa, D. M., Carrillo, M., & Del Rocio Boone, M. (2006, 27-01 Feb. 2006). A Database and Web Application Based on MVC Architecture. Paper presented at the Electronics, Communications and Computers, 2006. CONIELECOMP 2006. 16th International Conference on.

[11] Singh, A. K., & Roy, S. (2012). A network based vulnerability scanner for detecting SQLI attacks in web applications. Paper presented at the Recent Advances in Information Technology (RAIT), 2012 1st International Conference on.

[12] Xiang, F., Xin, L., Peltsverger, B., Shijun, C., Kai, Q., & Lixin, T. (2007, 24-27 July 2007). A Static Analysis Framework For Detecting SQL Injection Vulnerabilities. Paper presented at the Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International.

[13] Xin, W., Luhua, W., Gengyu, W., Dongmei, Z., & Yixian, Y. (2010, 26-28 Oct. 2010). Hidden web crawling for SQL injection detection. Paper presented at the Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference on.