International Journal of Computer Applications Technology and Research
Volume 4– Issue 4, 308 - 311, 2015, ISSN:- 2319–8656

A Detailed Study on Prevention of SQLI attacks for Web Security

Navjot Verma
Center of IT and Management
Punjabi University Regional
Mohali India

Amardeep Kaur
Center of IT and Management
Punjabi University Regional
Mohali India

Abstract: *SQL injection is the major susceptible attack in today's era of web application which attacks the database to gain unauthorized and illicit access. It works as an intermediate between web application and database. Most of the time, well-known people fire the SQL injection, who is previously working in the organisation on the present database. Today organisation has major concern is to stop SQL injection because it is the major vulnerable attack in the database. SQLI attacks target databases that are reachable through web front. SQLI prevention technique efficiently blocked all of the attacks without generating any false positive. In this paper we present different techniques and tools which can prevent various attacks*

Keywords: SQL Injection, SQL injection Prevention, web application, database, vulnerable.

1. INTRODUCTION

Web applications are being in a much wider area these days, online shopping, online banking and social networking is some of the key users of these [21]. All these users have the utmost priority for their privacy and security and these are the most vulnerable while being online. To secure these applications two phases are implied. These phases are:

1.1 Data base layer:
The *database layer* provides an object vision of database information by applying schema semantics to database records, so isolating the upper layers of the directory service from the underlying database system. The database layer is an inner boundary that is not exposed to users. No database admission calls are made directly to the Extensible Storage Engine; as an alternative, all database right to use is routed through the database layer [23].

1.2 Application layer:
refers to techniques of shielding Web applications at the application layer (last layer of the seven-layer OSI model) from nasty attacks that may picture private information. Protection is applied to the application layer especially to protect against illegal access and attacks.

Advantages of Web Security:

1. Internet sites are well-liked targets for crackers, and even without mean forces security holes can permit accidents happen.
2. A secure network is a good network.
3. This doesn't make it easy to take part in a web site. Scripts may require access to sensitive information, or at least information you don't want in the public domain before you are prepared.

SQL injection is one of the most serious threats to the data security of all web applications. SQL injection attack allows attackers to gain control of the original query, illegal access to the database and extract or transform the database [1]. The main cause of SQL injection vulnerabilities is: attackers use the input support to attack strings that contains special database commands. An SQLIA occurs when an attacker change the SQL control by inserting new keywords [2]. A successful SQLI attack hinder privacy integrity and availability of information in the database. In most of cases, SQL Injection is used to initiate the denial of service attack on web applications. The strictness of the attacks depends on the role or account on which the SQL statement is executed.
An attacker needs to know loop holes in the application before launch an attack. Attackers use: input format, timing, performance and error message to decide the type of attack suitable for an application. Database is the heart of many web applications, basis for which database more and more coming under great number of attacks. SQLIAs occur when data provided by the user is incorporated directly in the query and is not appropriately validated.

1.3 Vulnerability
Table 1 present the most common security vulnerabilities found in web programming languages [21].

Table 1: Types of Vulnerabilities

Vulnerability Types	Description
Type I	No clear distinction between data types received as input in the programming language used for the web application development

Type II	Delay of operation analysis till the runtime phase where the current variables are considered rather than the source code expressions.
Type III	The validation of the user input is not definite. Attacker taking advantages of insufficient input validation can utilize mean code to conduct attacks.
Type IV	Puny concern of type specification in the design. A number can be used as a string or vice versa.

2. TYPES OF SQLI ATTACK

The SQL injection attacks can be performed using a variety of techniques. Some of them are specified as follows:

First Order Attack: Attackers aim the database with strings attached to an input field and receives the answer immediately. Such attacks which exploit the lack of validation in the input field parameter are known as first order attacks [21].

Second Order Attack: An attacker attacks the database with inserting mean queries in a table but implement these queries from other actions [21].

Tautology Attack: Conditional operators are used by the attackers in the SQL queries such that the query always evaluates to TRUE [1,2,6,10].
For example, SELECT * FROM employee WHERE name = '' OR '1'='1';

Logically Incorrect Queries: An illegal query used by the attacker to glance at the whole database [1,2,6,10].
For example, "SELECT * FROM employee WHERE id =" + name + ";"

Piggy-backed Query: In this attack, attacker tries to add on supplementary queries but terminates the first query by inserting ";" [1,2,7].
For example, SELECT * FROM employee WHERE id=1; DROP TABLE employee;

Inference: The main goal of the inference based attack is to change the activities of a database or application. There are two well-known attack techniques that are based on inference: blind injection and timing attacks

Timing attack: In these types of attack an attacker observe the database delays in database response and gather the information. WAITFOR, IF, ELSE, BENCHMARK [1,2] cause delay in database response.
For example, SELECT * FROM employee WHERE id=1-SLEEP(15);

Blind injection: In this situation an attacker performs queries that have a Boolean result [1].
For example: SELECT * FROM employee WHERE id = '1008' AND 1=1;

Alternate Encoding: Attacker modifies the injection query by using alternate encoding such as hexadecimal, ASCII and Unicode [1,2] .
For example: SELECT * FROM employee WHERE id=unhex('05'); .

Union Query: An attacker makes use of vulnerable parameters and attach injected query to the safe query by the word UNION and get data about other tables from the application.
For example: Select * from company where name=‟ ‟ union select * from employee –„and Password=‟anypwd‟

Stored Procedure [10]: A stored procedure is a cluster of Transact-SQL statements compiled into a single execution plan. As stored procedure could be coded by programmer, attacker can execute these built in procedures**.**

3. RELATED WORK

There are many ways to prevent SQL injection attacks [1,2,7,14,15]. The various types of existing techniques for preventing SQL injection are as follows:

Negative Tainting, [1] Preventing SQL injection attack using negative tainting provide uniqueness by using linked list. This approach works on the untrusted strings and provides good response time for large database programs. This approach consists of (1) Identifying hot spot from the application (2) To find out SQL injection attack using negative tainting. (3) Inserting newly identified SQL injection attacks to get better accuracy.

Positive Tainting, [2] Positive tainting focuses on the recognition and marking of trusted strings. It uses the concept of syntax sensitive estimation. This system works in following manner- (1) Identifying trusted data source. (2) Allowing only data from such sources to suit a SQL keyword or operator in query strings. Trusted data strings can be more readily known. WASP (Web Application SQL injection Preventer) tool have implemented this approach. This approach is defined at the application level and it requires no alteration of the runtime (JVM) system, and it imposes low execution overhead. Positive tainting used to check SQLIA at the runtime. WASP tool works fruitfully but it blocked over 12000 attacks without generating false positives.

Input Filter Technique, [4] An SQL injection attack is interpreted differently on different databases. This technique provides the general solution to solve this problem. Depending on number of space, double dash and single quote the count value of the input value is increased by 1 because default count of the query is 1. Then the fixed count value and dynamically generated count compared to check SQL injection attack. But this technique has limitation that it works on single quote, double dash and space only.

Query Transformation and Hashing, [7] This technique uses a lightweight method to prevent SQL injection and works in two ways. Fist is to convert the query into structural form than parameterized form. Second apply an appropriate hash function to create unique hash key for each transformed query by using suitable hash function. Only the hash keys are stored instead of transformed query. A primary index can be created for fast and proficient searching. As this approach is proficient but it does not prevent second order SQL injection attacks and this approach can neither be applied to prevent XSS attacks.

SQL-ID, [14]

Kemalis and Tzouramanis have suggested novel specification-based methodology for the detection of exploitations of SQL injection vulnerabilities in "Specification based approach on SQL Injection detection" [3]. A Java-based application monitors by this system and identify SQL injection attacks in real time.

Dynamic Candidate Evaluations Approach, [15]

Bisht et al. propose CANDID. It is a Dynamic Candidate Evaluations method for automatic prevention of SQLInjection attacks. This structure dynamically extracts the query structures from every SQL query location which are intended by the developer (programmer). Hence, it solves the issue of manually modifying the application to create the prepared statements.

AMNESIA,[16] - AMNESIA approach for tracing SQL input flow and generating attack input, JCrasher for generating test cases, and SQLInjectionGen for identifying hotspots. The experiment was conducted on two Web applications running on MySQL1 1 v5.0.21. Based on three attempts on the two databases, SQLInjectionGen was found to give only two false negatives in one attempt. This framework is efficient considering the fact that it emphasizes on attack input precision. Besides that, the attack input is properly matched with method arguments. The single disadvantage of this approach is that it involves a number of steps using different tools.

SQLrand,[17] SQLrand approach is proposed by Boyd and Keromytis. To implement this approach, they use a proof of concept proxy server in between the Web server (client) and SQL server; they de-randomized queries received from the client and sent the request to the server. This de-randomization framework has 2 main advantages: portability and security. The proposed scheme has a good performance: 6.5 ms is the maximum latency overhead imposed on every query.

SQLIA Prevention Using Stored Procedures,[18] Stored procedures are subroutines in the database which the applications can make call to . The prevention in these stored procedures is implemented by a combination of static analysis and runtime analysis. The stagnant analysis used for commands identification is achieved through stored procedure parser and the runtime analysis by using a SQLChecker for input identification. This paper has proposed a combination of static analysis and runtime monitoring to fortify the security of potential vulnerabilities.

Adaptive algorithm,[19]

This method consists of the top features of parse tree validation technique and code conversion method. This technique parse the user input and verify whether its prone, if there is any chance of vulnerability present then code conversion will be applied over the input. This paper had also surveyed various SQL injection methods and techniques against SQL injection. It has also presented the algorithm to apply for the vulnerable code.

4. COMPARISON

The main goal for future improvement is to improve the efficiency of the technique by reducing false positive. Table 2, [1] shows a chart of the schemes and their prevention capabilities against various SQL injections attacks and précis the results of this comparison.

Table 2: comparison of various prevention schemes and various attacks

Schemes	Tautology	Logically Incorrect Queries	Union Query	Stored Procedure	Piggy Backed Queries	Inference Attack
AMNESIA	YES	YES	YES	NO	YES	YES
SQLrand	YES	NO	NO	NO	YES	YES
CANDID	YES	NO	NO	NO	NO	NO
SQLGuard	YES	NO	NO	NO	NO	NO
SQLIPA	YES	YES	YES	NO	YES	YES
Negative Tainting	YES	YES	YES	NO	YES	YES
Positive Tainting	YES	YES	YES	YES	YES	YES

5. CONCLUSION

With above data in place it can be reviewed that database security is major issue in today life. This paper presents a survey report on the SQL injection attacks and how attacks are implemented on the database using SQL queries. We first identified the various types of SQL injection attacks and then we examine various prevention techniques. Finally a comparative analysis of various types of prevention techniques of SQL injection is presented.

6. REFERENCES

[1] A. S. Gadgikar, "Preventing SQL injection attacks using negative tainting approach," in IEEE International Conference on Computational Intelligence and Computing Research, 2013, pp. 1–5.
[2] W. G. J. Halfond, A. Orso, and P. Manolios, "Using positive tainting and syntax-aware evaluation to counter SQL injection attacks," in Proceedings of the 14th ACM SIGSOFT International Symposium On Foundations of Software Engineering - SIGSOFT '06/FSE-14, 2006, pp. 175–185.
[3] S. Roy, A. K. Singh, and A. S. Sairam, "Detecting and Defeating SQL Injection Attacks," International Journal of Information and Electronics Engineering., vol. 1, no. 1, pp. 38–46, 2011.
[4] S. Bangre and A. Jaiswal, "SQL Injection Detection and Prevention Using Input Filter Technique," International Journal of Recent Technology and Engineering (2012), vol. 1, no. 2, pp. 145–150, 2012.
[5] E. Bertino, A. Kamra, and J. P. Early, "Profiling database applications to detect SQL injection attacks," in Conference Proceedings of the IEEE International Performance, Computing, and Communications Conference, 2007, pp. 449–458.

[6] A. Sadeghian, M. Zamani, and A. A. Manaf, "A Taxonomy of SQL Injection Detection and Prevention Techniques," in 2013 International Conference on Informatics and Creative Multimedia, 2013, pp. 53–56.

[7] D. Kar and P. Suvasini, "Prevention of SQL Injection Attack Using Query Transformation and Hashing," in Proceedings of the 2013 3rd IEEE International Advance Computing Conference, IACC 2013, 2013, pp. 1317–1323.

[8] R. Dharam and S. G. Shiva, "Runtime Monitors for Tautology based SQL Injection Attacks," IEEE Int. J. Cyber-Security Digit. Forensics, vol. 53, no. 6, pp. 253–258, 2012.

[9] P. Kumar and R. Pateriya, "A Survey on SQL injection attacks, detection and prevention techniques," in Computing Communication & Network Technologies, 2012, no. July, pp. 1–5..

[10] X. Fu, X. Lu, and B. Peltsverger, "A static analysis framework for detecting SQL injection vulnerabilities," in 31st Annual International Computer Software and Application Conference, 2007, no. Compsac, pp. 87–96.

[11] S. Thomas, L. Williams, and N. Carolina, "Using Automated Fix Generation to Secure SQL Statements [Short presentation paper]," 2007.

[12] K.-X. Zhang, C.-J. Lin, S.-J. Chen, Y. Hwang, H.-L. Huang, and F.-H. Hsu, "TransSQL: A Translation and Validation-Based Solution for SQL-injection Attacks," in 2011 First International Conference on Robot, Vision and Signal Processing, 2011, pp. 248–251.

[13] K. Kemalis and T. Tzouramanis, "SQL-IDS : A Specification-based Approach for SQL-Injection Detection," pp. 2153–2158, 2008.

[14] P. Bisht, "CANDID : Dynamic Candidate Evaluations for Automatic Prevention of SQL Injection Attacks," ACM Int. J. Comput. Sci., vol. V, no. 2, pp. 1–38, 2010.

[15] G. T. Buehrer, B. W. Weide, and P. A. G. Sivilotti, "Using Parse Tree Validation to Prevent SQL Injection Attacks," no. September, pp. 106–113, 2005.

[16] S. W. Boyd and A. D. Keromytis, "SQLrand : Preventing SQL Injection Attacks," IEEE Appl. Cryptogr. Netw. Secur.,2004,vol. 3089, pp. 292–302..

[17] W. G. J. Halfond and A. Orso, "Preventing SQL injection attacks using AMNESIA," in Proceeding of the 28th international conference on Software engineering - ICSE '06, 2006, p. 795.

[18] Z.Su and G. Wassermann, "The Essence of Command Injection attacks in Web Applications", 33rd ACMSIGPLAN, SIGACT Symposium on Principles of Programming Languages, Charleston, South Carolina, USA, 2006,pp. 372-382.

[19] Ashish John "SQL Injection Prevention by Adaptive Algorithm," IOSR Journal of Computer Engineering, 2015,Vol 17,pp. 19-24

[20] Diksha Upadhya "A Survey on SQL Injection - Vulnerabilities Attacks and Prevention Techniques".

[21] K. S. Chavda, "International Journal of Advance Engineering and Research," Sci. J. Impact Factor, vol. 1, no. 12, pp. 173–179, 2014.

[22] A. John, A. Agarwal, and M. Bhardwaj, "An adaptive algorithm to prevent SQL injection," *An Am. J. Netw. Commun.*, vol. 4, pp. 12–15, 2015.

[23] https://technet.microsoft.com/en-us/library/cc961800.aspx