IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A REVIEW ON SQL INJECTION DETECTION AND PREVENTION TECHNIQUES

Manpreet Kaur*, Anand Kumar Mittal
M. Tech IT Guru Kashi University Talwandi Sabo (Punjab).
Assistant Professor Guru Kashi university, Talwandi Sabo (Punjab).

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.

KEYWORDS— SQL Injection, SQL injection Prevention.

INTRODUCTION

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 changes 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.

Phase used in web application security:
There are two types of Phase:

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.

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.

There are 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 good network is a secure 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.

Types of SQL Injection-
Some of the attacks are:
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.

Second order attack
An attacker attacks the database with inserting mean queries in a table but implement these queries from other actions.

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].

SELECT * FROM student 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].
"SELECT * FROM student 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].

SELECT * FROM student WHERE roll_no=1;DROP TABLE student;

Timing attack
In this types of attack the IF ELSE statement is used for injecting queries [1 2]. Example WAITFOR, IF, ELSE, BENCHMARK etc .
SELECT * FROM student WHERE roll_no=1-SLEEP(15);

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

LITERATURE SURVEY
A. S. Gadgikar *et al* SQL injection attack has become a major threat to web applications, which gives unauthorized access [1]. This paper has used negative tainting approach with linked list structure. This approach is implemented between application program and database server. All the symptoms of SQL injection attack are stored in database. The future goal is to improve the efficiency by reducing false positives. Multithreading can be used to reduce the time requirements. In this paper SQL database is used for testing.

W. G. J. Halfond *et al* An SQLIA occurs when an attacker changes the developers SQL command by inserting new SQL keywords or operators [2]. In this paper their approach works by identifying trusted strings and allowing only those trusted strings to be used to create sensitive part of the SQL query strings. WASP (web application SQL injection prevented) tool implements this technique. It stops all the attacks without generating false positives. In the Future work- the proposed work can be used for binary programs and further improve the efficiency of technique to reduce the amount of information required.

S. Roy *et al* SQL injection is most vulnerable to web applications. In this paper CSR scanner used that is light weighted, fast and low false positive rate [3]. CSR scanners work in three steps- crawling the web page, scanning for vulnerable points and generate attack and report. Then the later part includes security mechanism by using the filter with any web application. CSR scanner is able to identify higher number of vulnerabilities. Efficiency of filter tested by using CSR scanner.

S. Bangre *et al* SQL injection target databases that are accessible through a web front end and take advantage of flaws in the input validation logic of web components such as CGI script [4]. In this paper input filter technique has been used which checks the attribute value for single quote, double dash and space provided by the user through input fields. This paper proposes simple and effective method by using SQL query parameter counter to prevent SQLIA. This paper utilizes both static and dynamic analysis to detect SQL injection attack. Future work is also needed on this paper because research work is not related to web applications of SQL but also other web application attack such as XSS.

Existing Techniques-
Some of the existing techniques to prevent SQLIA are:
Positive Tainting- 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.

Negative tainting- 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.

Query Transformation and Hashing- 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.

Input Filter Technique- 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.

SAFELI - Static Analysis Framework for discover SQL Injection approach is anticipate to classify the SQL Injection attacks at the compile-time. This static analysis tool has two main advantages. Firstly, it does a White-box Static Analysis and secondly, it uses a Hybrid-Constraint Solver. For the White-box Static Analysis, the proposed approach considers the byte-code and essentially with strings. For the Hybrid-Constraint Solver, the method equipment an efficient string analysis tool which is able to deal with Boolean, integer and string variables.

SQL-ID –
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]. The proposed query-specific detection allows the system to perform focused analysis with a small computational overhead and produces no false Positives or false negatives.

AMNESIA - 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. The proposed 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 only disadvantage of this approach is that it involves a number of steps using different tools.

SQLrand Scheme - SQLrand approach is proposed by Boyd and Keromytis. For the implementation, 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 – 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 static analysis used for commands identification is achieved through stored procedure parser and the runtime analysis by using a SQLChecker for input identification. Proposed a combination of static analysis and runtime monitoring to fortify the security of potential vulnerabilities.

Dynamic Candidate Evaluations Approach
Bisht et al. propose CANDID. It is a Dynamic Candidate Evaluations method for automatic prevention of SQLInjection attacks. This framework 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.

Comparison
The following is a comparison of various SQL injection prevention techniques along with the types of attack they can prevent.

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

REFERENCES

[1] 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] 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.
[10] P. Kumar and R. K. Pateriya, "A Survey on SQL Injection Attacks , Detection and Prevention Techniques," no. July, 2012.
[11] X. Lu, B. Peltsverger, S. Chen, G. Southwestern, K. Qian, and S. Polytechnic, "A Static Analysis Framework For Detecting SQL Injection Vulnerabilities," no. Compsac, 2007.
[12] S. Thomas, L. Williams, and N. Carolina, "Using Automated Fix Generation to Secure SQL Statements [Short presentation paper]," 2007.
[13] 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," 2011 First Int. Conf. Robot. Vis. Signal Process., pp. 248–251, Nov. 2011.