International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 09 | Sep 2019 www.irjet.net p-ISSN: 2395-0072

An Efficient Technique for finding SQL Injection using Reverse Proxy Server

Raj Agarwal[1], Sumedha Sirsikar[2]

[1]Student, Computer Science and Engineering, MIT-World Peace University, Pune, Maharashtra India
[2]Prof., Computer Science and Engineering, MIT-World Peace University, Pune, Maharashtra India

---

Abstract - One of the most serious threats to the data driven applications is SQL Injection. Web applications that are vulnerable to SQL injection may permit an invader to gain ample access to their underlying databases. A SQL Injection Attack sometimes starts with identifying weaknesses in the applications where unrestricted users' input is transformed into database queries. A successful SQL Injection attack interfere privacy, Integrity and Availability of information in the database. There are several ways of detecting and preventing SQLIA such as Hybrid Method, Decision Tree Classification, Hidden Markov Model, Removing of parameter values, Dynamic SQL, Stored Procedure. For each Technique it is not possible to detect and prevent all the types of SQL Injection attack. By exploiting vulnerabilities in web application, an invader can pass through security system even when custom firewall and IDS systems are placed to secure the application. Reverse Proxy could be a technique which sanitize the user's inputs. In this technique a filter program redirects the user's input to the proxy server before it is directed to the application server and data cleansing algorithm is triggered using a sanitizing application. The data cleansing algorithm uses sanitization to check whether the user input contains malicious code or not. If malicious patterns are found, then the user input request is rejected otherwise it is been passed to application server.

Keywords— SQL Injection, SQL attack, Security threats, Run time monitoring.

I. INTRODUCTION

Definition of SQLIA

SQL injection is a type of attack in which the invader adds malicious code to user input box to gain access or to modify the data [2, 5, and 6]. SQL injection vulnerability grants an attacker to insert commands directly to a web application's underlying database and terminate functionality, confidentiality and privacy.

SQL Injection is one of the attacks that are caused by the attackers to alter the structure of the original SQL query. In the user input field by injecting SQL code in order to gain unauthenticated access to the database. The access over the internet than can perform many functions which becomes a huge threat to the database [4,7]. These functions might include retrieving the other users' passwords, table deletion, updating someone's valuable information, or even deletion of entire database etc.

SQL injection attacks are classified under several categories:

i) **Tautology: -** This type of attack represents the SQL manipulation category in which an attacker can inject malicious code into the query, it is based on the conditional statement.

ii) **Illegal/logically incorrect queries: -** This type of attack represent the SQL manipulation category in which the attacker can get advantage from error message which is generated from the database server.

iii) **Timing attack: -** In this type of attack, attacker collect information from the database by observing timing delays.

iv) **Union queries: -** This type of attack represents Code Injection and SQL manipulation category. Union Query adds malicious code or injects code with a safe query to get another table information from the database server. With the help of this type of attack, the attacker can extract data type information of the column.

v) **Blind SQL injection attacks: -** In this type of attacks attacker can steal data from database asking true and false questions through SQL statement.

vi) **Piggybacked queries: -** This type of attack represents to Code injection category. In this type of attack, attackers inject malicious code with traditional queries and perform data manipulation operation like INSERT, UPDATE and DELETE clause for manipulating a record.

Currently there are wide range of SQL injection detection and prevention techniques are proposed and used by developers and application owners. We can divide these techniques based on the nature of their defense to three main categories of:

• **Best code practices:** They are set of guidelines and policies for developers to improve the quality of their code by following them. [10] Using of the best practices can highly decrease the potential chance of SQL injection vulnerabilities.

• **SQL injection detection:** This technique detects the SQL injection attacks.

• **SQL injection runtime prevention:** This technique prevents SQL injection attack in the execution time and compares them against the legitimate query.

II. LITERATURE REVIEW

TECHNIQUES USED FOR SQL INJECTION DETECTION AND PREVENTION.

Hybrid varieties and combination of id-based detection mechanisms are used for detecting SQL injection attacks in Web applications. Machine Learning Classifier and pattern matching inspection engine is combined to prevent SQL Injection attack. Samantha et al proposed a recognition mechanism using regular expressions which allows only the trusted data and syntax aware SQL queries to be executed in a Web application.

Various methods are available to overcome XSS. But the issue still occurs in many applications as the methods are difficult to adopt and implement. Moreover, the difficulty of XSS problem has further added to it. A code auditing approach is provided which comprises of two phases. First phase is the extraction of all features that would stand against XSS attacks. The second phase does the tasks for preventing XSS attacks using the information in the first phase. Using this, a tainted information flow graph is modeled. This is implemented using XSS defense extractor using seven test subjects. But the DOM based XSS attack is not considered in this model and only few applications are examined.

Parse Tree Validation Technique: -

This technique performs comparison at runtime between parse tree of the particular statement and original statement, execution does not begin before the parse tree intended SQL query as well as the resulting SQL query generated after attacker input do not match.

Analyzing and Monitoring for Neutralizing SQL Injection Attack (AMNESIA)

AMNESIA uses a model that is built statically containing of the valid queries and checks it with the dynamic queries generated upon the submission of the user's input in order to analyze the presence of SQLI in the application code.

SQL Check Approach: -

This approach is proposed by Zhendong Su and Gary Wassermann. They implemented their algorithm with SQLCHECK. Checks on given input queries were applied with defined ones by the developer and a secret key is applied for user input delamination in this approach. SQL check approach has no false positive or false negatives and overhead runtime rate is minimal that can be implemented in a variety of web application using different platform.

Stored Procedure Approach: -

The researchers like Ke Wei, M. Muthuprasanna and Suraj Kothari suggested this approach. Stored procedures are subroutines which help applications to interact with the database server for performing an operation on the database. The combination of static analysis and dynamic analysis is used to identify the SQL Injection attacks; static analysis is used for command verification which uses through subroutines parser and runtime validation by using SQLCHEAKER [8] for input validation.

Manual Approach:-

This approach is given by MeiJunjin. The manual approach is used in three ways; one is defensive programming which means that the developer writes code in such a way that the user cannot insert malicious character and keywords. In another way, a developer can also use safe API for preventing SQL Injection attacks like SQLDOM approach. The third way is the Code Review, which prevents the SQL Injection on the database but it is time-consuming.

III. PROPOSED SYSTEM ARCHITECTURE.

The effective approach to detect and curb the SQL attacks in the database using a modified data cleaning algorithm. The project work mainly includes tainted information from sanitization that is further sent to establish a new prototype and also to its database without any delay for runtime monitoring. The client request is redirected to reverse proxy server and if the incoming request is valid request then only it is requested to application server. The reverse proxy server checks for Dos attack, SQL/XSS attack and if the incoming request is genuine then the user is validated using one-time password. If the user validates all these parameters then only the user is redirected to application server.

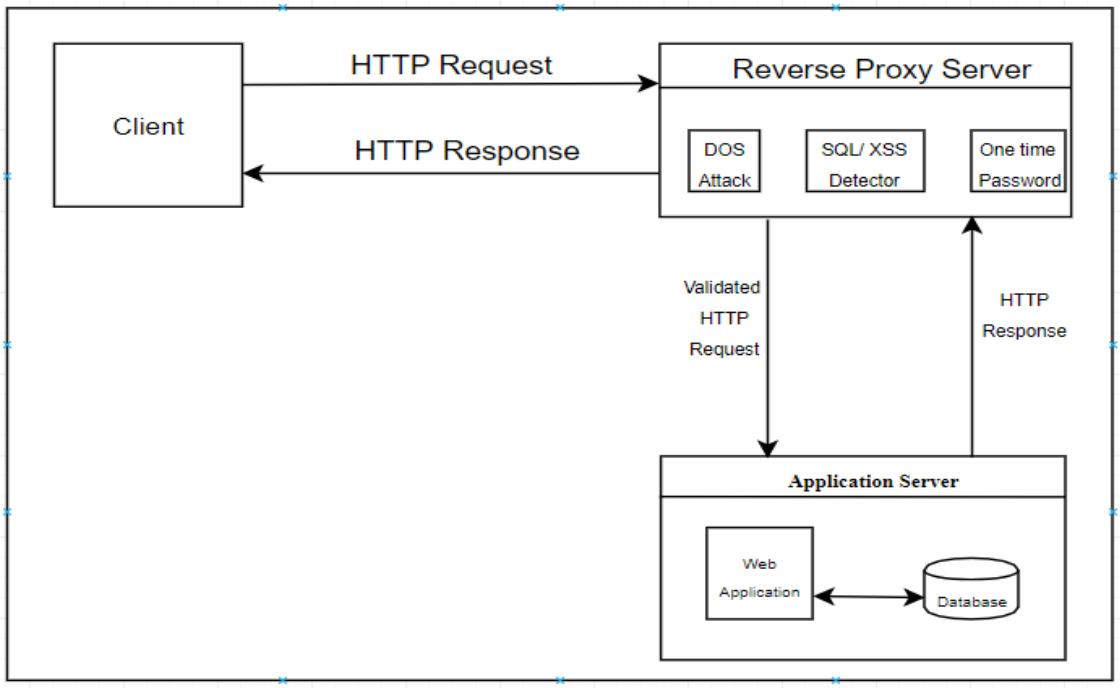

Fig - 1: - System Architecture.

- **Banking Application for SQL injection simulation**

  Banking application is one of the critical applications which needs a lot of data and cross site scripting security. SQL injection happens at the data level or application level. Users of banking application can come from different regions to attack or steal user information or transfer payment from the user's account.

- **Web Application attack**

  Following are some list of web attacks;

a.	A hacker finds a vulnerability in your custom web application and sends an attack via port 80/443 to the web application.
b.	The unauthorized code is received from the web application server and again it is sent to the database server.
c.	The code execution is done by database server and data between credit card tables is

	returned
d.	Pages with data are dynamically generated by a Web application server
e.	The web server sends card details to a hacker using malicious code.
f.	The web server sends card details to a hacker using malicious code.
g.	The malicious code at the database is called a SQL Injection attack and code at the page is called a Scripting attack of Cross Site.

- **Reverse Proxy Server: -**

  This server is mainly responsible for maintaining security policies for the banking server.

- **Data Cleansing Algorithm**
  - **SQL Injection: -**
    SQL injection filters data attacks happening when the user types malicious data in text boxes. This data is checked for various patterns of SQL injection.
  - **Cross-site HTML and JavaScript: -**

    This type of scripting can also be avoided by checking script patterns in the data in text boxes

- **SQL Injection and prevention: -**
  a) SQL Pattern Matching.
  b) Keyword operator = in like search.
- **Cross Site Scripting Injection and prevention: -**
  a) HTML Tag Search and removal.
  b) Forbidden and allowed tag list**.**

IV. ALGORITHM USED: -

- **Data Cleansing Algorithm Details**

HTML Sanitizer:-

HTML Sanitizer removes damageable tags and its attributes from HTML code. It takes a string from HTML source code and lined all of them by tagging them as they do not belong to a list of safe tags respectively.

Tokenizer:-

Tokenizer usually splits the HTML text of user input into tokens. These tokens are nothing but a single atomic unit of supplied text. Following are some sample examples of tokens: tag start (), comment (), tag content ("text"), a tag closing (). Few tokens are created and every one token present in the list are matched with the tags and also with white list tags is its main functionality.

HTML Encoder:-

HTML encoder performs the character escaping. To encode the input entered by the user is the main objective of the HTML Encode Method. The Html Encode process is applied to the string that used to prevent some special characters that are method applies HTML encoding to a string to prevent a special character to be interpreted as an HTML tag.

Script Pattern:-

This contains all the tags and patterns that are used to match with the tokens which are formed by the tokenizer. It contains a list of all the forbidden tags, allowed tags, tag starting pattern, tag closing pattern, comment patterns, style pattern, URL pattern etc.

Pattern Matcher:-

The key purpose of this module is to fetch inputs from the token list and compare or match them with the scripted patterns respectively. An individual tag list is prepared that can store all the rejected tags on another hand forward all the accepted tags for encoding to HTML Encoder.

Forbidden tags	(script|object|embed|link|style|form|input)
allowedTags	(b|p|i|s|a|img|table|thead|tbody|tfoot|tr|th|td|dd|dl|dt|em|h1|h2|h3|h4|h5|h6|li|ul|ol|span|div|strike|strong|"+ "sub|sup|pre|del|code|blockquote|strike|kbd|br|hr|area|map|object|embed|param|link|form|small|big)
commentPattern	<!--.*
tagClosePattern	</(?i)(\\w+\\b)\\s*>$
tagStartPattern	<(?i)(\\w+\\b)\\s*(.*)/?>$
standAloneTags	(img|br|hr)
attributesPattern	(\\w*)\\s*=\\s*\"([^\"]*)\
stylePattern	([^\\s^:]+)\\s*:\\s*([^;]+);?")
urlStylePattern	(?i).*\\b\\s*url\\s*\\(["\"]([^)]*)["\"]\\)")

V. RESULT AND ANALYSIS

Following Table represents the Detection Results of the attacks and its accuracy.

Table: - Detection Results

Detection Results				
Sr.No	**Attacks**	**No of Test Cases**	**% Caught**	**Accuracy**
1	SQLi	100	93	93
2	XSS	100	85	85

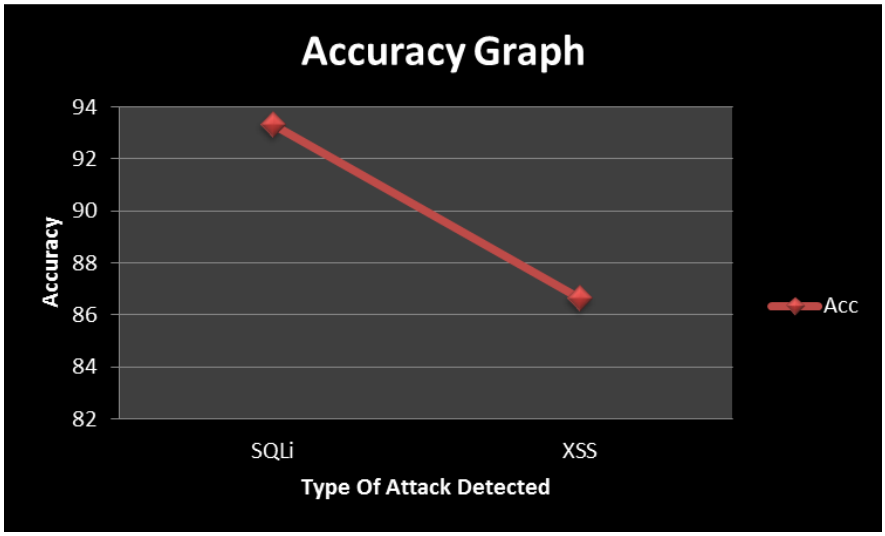

Figure: - Detection Graphs

CONCLUSION

One of the most dangerous vulnerabilities in the web application is SQL injection. Until now many different techniques are proposed by researchers to defeat it. However attackers always found a new method to bypass these solutions. The attacker can leverage the syntax and capabilities of SQL itself, as well as the power and flexibility of supporting database

functionality and operating system functionality available to the database. The experimental results confirm that the proposed approach is effective in detecting and preventing almost all types of SQLIA with low false positive rate and high accuracy.

REFERENCES

[1] Takeshi Matsuda, Daiki Koizumi, Michio Sonoda, Shigeichi Hirasa_, "On predictive errors of SQL injection attack detection by the feature of the single character" Systems, Man, and Cybernetics (SMC), 20 II IEEE International Conference on 9-12 Oct 2011, On Page 1722-1727.

[2] Angelo Ciampa, Corrado Aaron Visaggio, Massimiliano Di Penta: "A heuristic-based approach for detecting SQL-injection vulnerabilities in Web applications".

[3] G Buehrer, B.W. Weide, P.A.G Sivilotti, Using Parse Tree Validation to Prevent SQL Injection Attacks, in 5th International Workshop on Software Engineering and Middleware, Lisbon, Portugal, 2005, pp. 106-113.

[4] Z. Su and G Wassermann "The essence of command injection attacks in web applications". In ACM Symposium on Principles of Programming Languages (POPL'2006), January 2006.

[5] RA. McClure, and H. Kruger, "SQL DOM: compile-time checking of dynamic SQL statements," Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on, pp. 88- 96, 15-21 May 2005.

[6] Ke Wei, M. Muthuprasanna, Suraj Kothari, "Preventing SQL Injection Attacks in Stored Procedures" Proceedings of the 2006 Australian Software Engineering Conference (ASWEC 06 IEEE).

[7] P. Bisht, P. Madhusudan, and V. N. Venkatakrishnan. CANDID: Dynamic Candidate Evaluations for Automatic Prevention of SQL Injection Attacks. ACM Trans. Inf. Syst. Secure., 13(2): 1- 39, 2010.

[8] Mei Junjin, "An Approach for SQL Injection Vulnerability Detection," Proc. of TTNG '09, pp.1411-1414, 27-29 April 2009.

[9] William GJ. Halfond, Alessandro Orso." WASP: Protecting Web Applications Using Positive Tainting and Syntax-Aware ENGINEERING, VOL. 34, NO. I, JANUARY/FEBRUARY 2008.

[10] Shikhar Jain & Alwyn R. Pais," Model-Based Approach to Prevent SQL Injection Attacks on .NET Applications" International Journal of Computer Science & Informatics, Volume-I, Issue-H, 2011.

[11] T. Pietraszek and C. V. Berghe. Defending against Injection Attacks through Context-Sensitive String evaluation. Recent Advances in Intrusion Detection, Volume: 3858, Pages: 124-145, 2006.

[12] Bibliography: Bernard Menezes, Indian Institute of Tech, Mumbai, “Network Security and Cryptography”, Cengage Learning Publications.

[13] An Article on Web Application Security 101 by Applicure technologies “dotDefender Web Application Security" published in the year 2011