ISSN (Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 12, December 2015

A Dynamic Approach to Detect & Prevent SQL Injection Attack to Overcome Website Vulnerability

Er. Kanika[1], Er. Prabhjot Kaur[2]

Research Scholar, Dept. of Computer Engineering, Universal Institute of Engineering & Technology, Lalru, India[1]

Principal, Universal Polytechnic College, Lalru, India[2]

ABSTRACT: SQL injection is a technique where the invader injects an input in the query in order to change the structure of the query intended by the programmer and acquirement the access of the database which results modification or deletion of the user's data.SQL injection attack is the most common attack in websites in these days. Some malicious codes get injected to the database by illicit users and get the access of the database due to lack of input validation. This paper presents the technique for detection and prevention of SQL injection attack. In this paper some predefined method of detection and the some new techniques of preventions are discussed.

KEYWORDS: Web application, SQLIA, Detection, Prevention, Vulnerabilities, Web architecture.

I. INTRODUCTION

Web browsers are software application that allows users to retrieve data and interact with content located on web pages inside a website. The web is an extremely programmable environment that allows mass customization through the immediate deployment of a large and miscellaneous range of applications, to millions of global users. Two important components of a recent website are flexible web browsers and web applications both available to all and as stated, websites depend on databases to deliver the essential information to visitors. If web applications are not secure, i.e., vulnerable too, at least one of the different forms of hacking techniques, then your entire database of sensitive information is at serious risks. Some hackers, for example, may maliciously inject code within vulnerable web applications to trick users and redirect them in the direction of phishing sites. This technique is called Cross-Site Scripting and may be used even though the web servers and database engine include no vulnerability themselves. Recent research shows that 75% of computer-generated attacks (cyber attacks) are done at web application level.

Database security is the system, processes, and procedures that protect a database from unintended activity. Unintended activity can be categorized as authenticated misuse, malicious attacks or inadvertent mistakes made by authorized individuals or processes [4].

II. RELATED WORK

Over years, many tools for detection and prevention of SQL Injection attacks have been developed.

AMNESIA [4] developed by Halfond and Orso is a detection and prevention tool for SQL injection attack. It used static analysis and runtime monitoring .The tool builds a model of the legitimate queries at each hotspot i.e. where SQL queries are issued to database engine and monitors the application at runtime to ensure that all generated queries match the statically-generated model.

In [5], a tool named CANDID is proposed for detecting SQL injection. The tool dynamically infers the programmer -

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 12, December 2015

intended query structure on any input, and detects attacks by comparing them against the intended query structure. In [6], SQLRand uses instruction set randomization to detect and abort queries with injected code and every SQL keyword is joined with a random integer to mislead the attacker.

The proposed technique in [7] prevents SQLIA in stored procedures by combining static application code analysis with runtime validation. In the static part, a stored procedure parser is designed and it instruments the necessary statements in order to compare the original SQL statement structure to that including user inputs for every SQL statement which depends on user inputs. The technique abstracts the intended SQL query behaviour in an application in the form of an SQL-graph and this graph is then validated against all the different user inputs at runtime to capture all malicious SQL queries, before they are sent for execution.

An efficient technique is presented in [2] for detecting and preventing SQL Injection attack using pattern matching algorithm. Pattern matching identifies or detects any anomaly packet from a sequential action, as the malicious code includes many anomaly packets or strings.

The technique proposed in [3] uses a new middle-ware-based prevention mechanism:

SQLIMW: - The SQLIMW avoids SQL-Injection attack from the programmer to the server. Hash function is used to replace encryption for better security. Furthermore, by combining the hash with XOR, it protects username, password and private key of SQLIMW. The proposal provides better security and efficiency. This kind of middleware is transparent to the programmer is not limited to sign-on authentication mechanism or single sign-on system, it can exist in any layer of Web application system exchanging information with the database. Almost every solution given to detect or prevent SQL Injection attacks is either for application layer or for database layer. But none of them provide security at both application and database layers. Also very little emphasis is laid on preventing SQL Injection attacks in stored procedures.

[7] Although the mechanism of SQLIA is the same for both stored procedure and application layer program, the same detection technique could not be applied to stored procedures, because of limited programmability of stored procedures and the technique's usability and deploy ability. Many existing techniques, such as filtering, information-flow analysis, penetration testing, and defensive coding, can detect and prevent a subset not all of the vulnerabilities that lead to SQLIAs.

[1]Another technique provides a two phase security to the application, so that, if one phase is compromised, the second phase can still prevent the attack. Here an efficient scheme for detecting and preventing SQL Injection attacks has been introduced. This scheme provides security from both the frontend as well as the backend of the application so that, if security is compromised at one end, the second end can still prevent SQL injection attacks.

III. PROPOSED WORK

Here is an efficient and new algorithmic approach is developed for early detection and mechanism for the avoidance of the SQL Injections Attacks. We have implemented the mechanism with the use of Fuzzy Parameters and set of rules. The two rules stated below:-

RULE - 1

If PositiveAttempt => (0, 0, 1)
=> (FALSE, FALSE, TRUE)
NegativeAttempt => (111*)

=> (FALSE, FALSE, FALSE*)
PositiveAttempt => (0, 1, 1) =>
(FALSE, TRUE, TRUE)

OR

NegativeAttempt => (1, 1, 0)
=> (TRUE, TRUE, FALSE)

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 12, December 2015

Exit with Message ("Not Allowed")

RULE - 2

(IdentifyUser) =>

Navigate and Generate Previous User
Attempts x => (Fetch (AllUserAttempts))

If (ANY(x) = "SQL Injection")

Exit with Message "Blacklist User Profile"

In the above rules we categories the user in two group Authorized and unauthorized. Authorized users are those who are registered and access the database and attempt are tautology i.e. true. Unauthorized users are those who want to access the database and try to destroy them. To check the unauthorization all attempts are false. The users who try three false attempts are blacklisted and can't login again. All the attempts are saved in database. The total execution time also saved in the database. The encryption is used for the password protection so that any user can't able to see the password.

IV. ALGORITHM USED:

The steps of algorithm are as following

1. Initialize the URL (Ui)
2. Generate the Form Hierarchy FHj -> Ui
3. Create and Maintain an Injection Vulnerability Relation IVRt
4. Put the Sequence of Vulnerability Values with Fuzzy Parameter in FPj -> IVRt
5. Insert the SI (SQL Injection) in FHj
6. Measure Risk / Vulnerability Aspect with each SIS (SQL Injection String)
7. Create and maintain the separate and arbitrary Relation of Vulnerability Investigation
8. Detailed Analysis of the Attacks: -
    a. Classification of Attacks
    b. Analysis of Authenticated and Legitimate Traffic
9. Detailed Report and Plots Generation

The steps explained as following:-

Collection of the Training Data Set of Users and Cheat sheet for Analysis

1. The Training Data Set Consists of the URLs for investigation
2. Generation of the patterns and keys
3. Deep Analysis on each parameter.
4. Applying the proposed model on the Training data set
5. Fetch Results
6. Data Interpretation

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 12, December 2015

Firstly we collect all the entries of the user attempt stored in our database .User database consist of user name and password field that is linked with the URL of the website .It also consists of cheat sheet for analysis .After that set of keys is generated with respect of user attempt. Encryption is used to generate keys and patterns are generated according to the string match. Our next step is to analyze various parameters i.e. Fuzzy set Parameter and their generated pattern. After that our model applied on the data and results are gathered. Table below shows the database structure.

Table 1.1 Database Schemas and Structure

SQL Injection String	Penetration Level	Classification
admin' --	100	Attempt Admin Access
admin' #	90	Attempt Admin Access
admin'/*	80	Attempt Database Exploitation
' or 1=1--	90	Database Bypass
' or 1=1#	100	Access Database Tables
' or 1=1/*	70	Database Access
') or '1'='1--	80	Database Exploitation
') or ('1'='1--	50	Database Exploitation
1' ORDER BY 1--+	60	Access Database and Tables
1' ORDER BY 1--+	90	DB Exploitation
1' ORDER BY 2--+	60	DB Exploitation
1' ORDER BY 3--+	40	DB Exploitation
-1' UNION SELECT 1,2,3-	80	DB Exploitation
1' ORDER BY 4--+	90	DB Exploitation
1' GROUP BY username	80	Authentication Byepass
1' or '1' = '1	50	Authentication Byepass
1' OR '1' = '1	60	Access Username without Authentication

The table 1.1 shows the classification of the attack according to SQL Injection string.SQL INJECTION string has the penetration level. It specifies the risk included with SQL injection string.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 12, December 2015

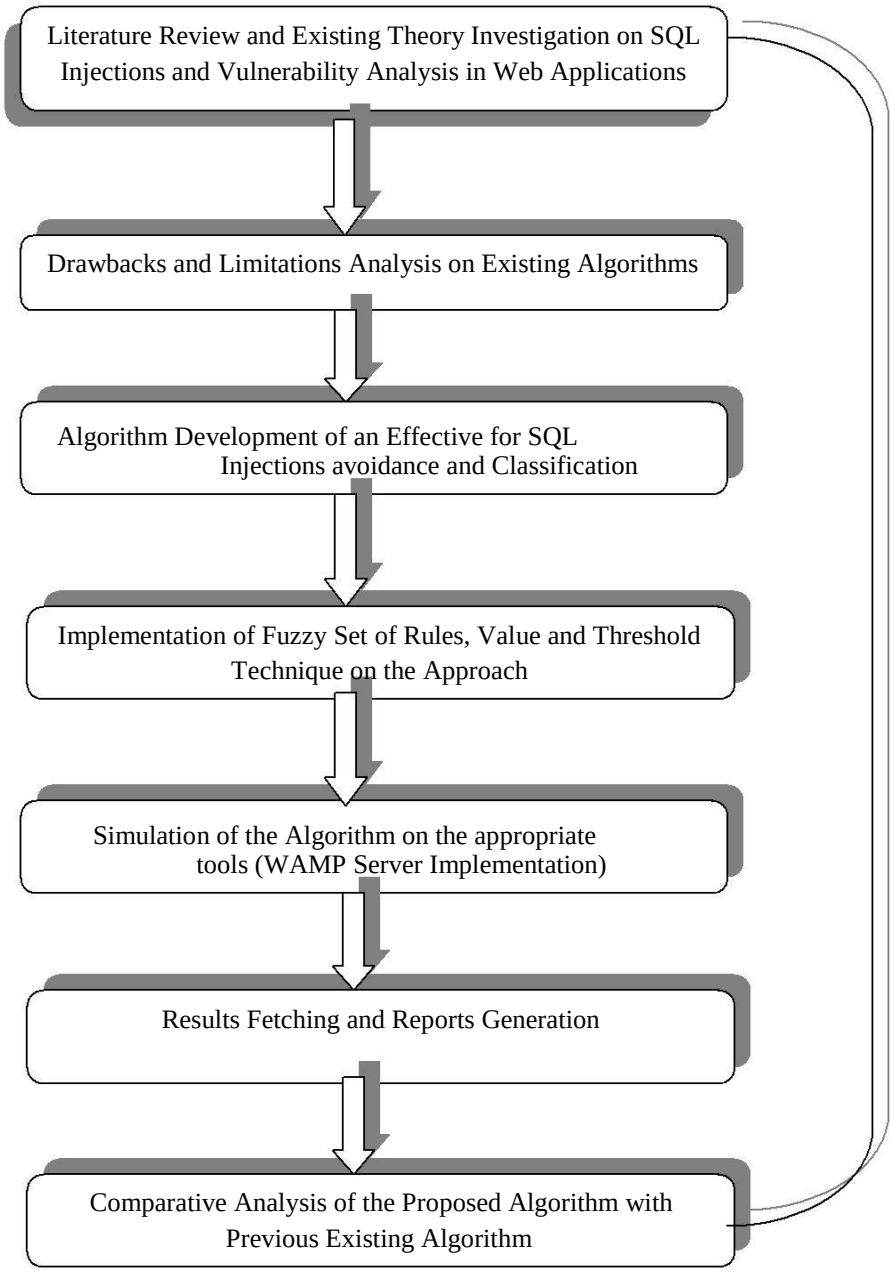


Fig. 1.1 Design Flow of Research Process

In Fig. 1.1 the first step is to review the related literature, and then in next step we find out the drawbacks&limitations in existing algorithm. To overcome the drawbacks&limitations we design the new algorithm in order to avoid SQL Injection Attack. We implement the fuzzy rules on the Wamp Server. Then next step is to generate the results. In the last step we have to compare the newly design algorithm with the existing algorithm.

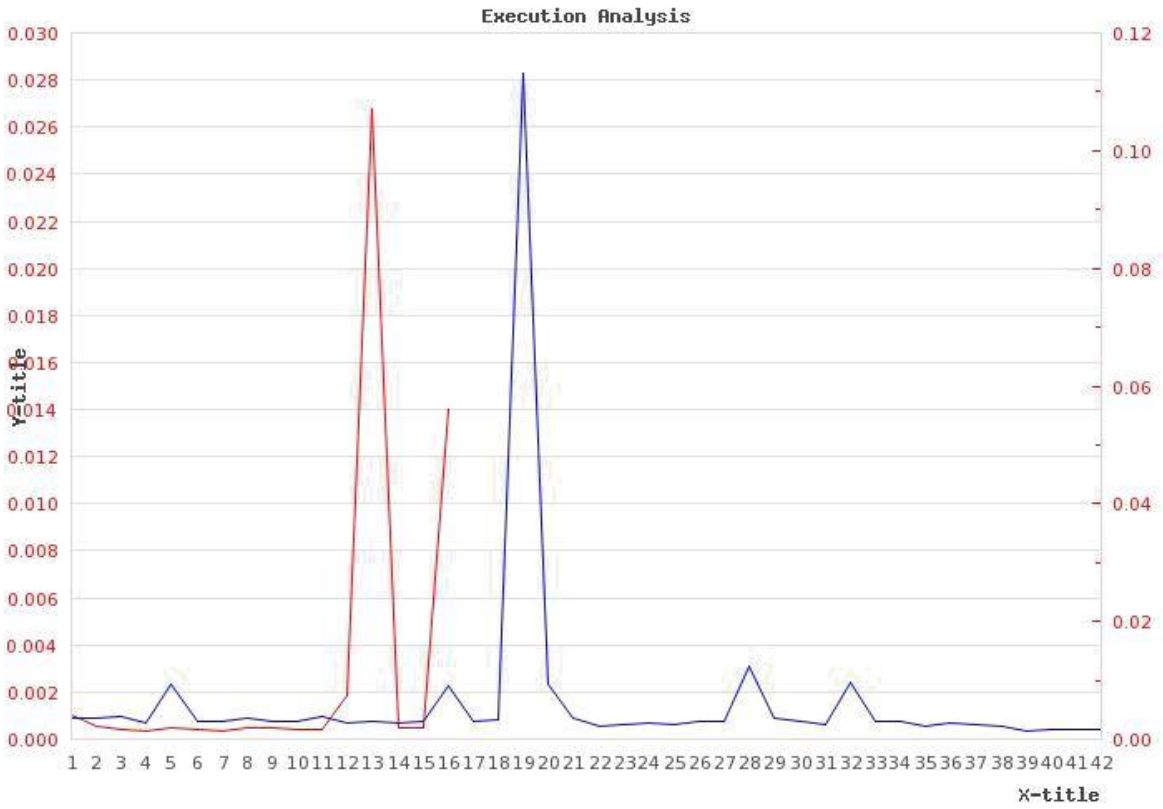
ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 12, December 2015

V. RESULT



The above Fig. shows that which user is authorized one and unauthorized one and also shows that execution time of our approach and classical approach. The **blue line** shows the new approach and **red line** shows the classical approach. New approach is better than classical approach. In our approach we can find the type of attack which user is unauthorized we can implement two level of security at front end and back end which was not in previous approach. We are implementing dynamic approach. In this approach changes can be done at anytime.

VI. CONCLUSION AND FUTURE SCOPE

SQL injection is one of the attack using specialized code injection techniques, used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker. In this research work, we have developed and simulated a new algorithmic approach for the avoidance and detection of the current techniques of SQL injection as well as solution methodology. To perform this operation, this work first identified the various types of SQLIAs and Cheat sheet known to date. Many of the techniques has problems handling attacks that take advantage of poorly- coded stored procedures and SQL queries cannot handle attacks.

Future work should focus on evaluating the existing techniques precisely with metaheuristic techniques and effectiveness in practice. Empirical evaluations can be performed which allow comparing the performance of the different techniques when they are subjected to real-world attacks and legitimate inputs. In the future work, the combination of metaheuristics can be implemented.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 12, December 2015

REFERENCES

[1] Abhishek Kumar Baranwal, "Approaches to detect SQL injection and XSS in web applications", EECE 571B, Term Survey Paper, 2012.
[2] Bojan Jovicic, Dejan Simić, "Common Web Application Attack Types and Security Using ASP.NET", ComSIS Vol. 3, No. 2, December 2006.
[3] Closing the Back Door on Network Application Vulnerabilities Web Server Protection and Your Security Strategy, A Sophos Whitepaper April 2012
[4] Debasish Das, Utpal Sharma & D.K. Bhattacharyya, "An Approach to Detection of SQL Injection Attack Based on Dynamic Query Matching", ©2010 International Journal of Computer Applications (0975 - 8887) Volume 1 – pp 25, 2010.
[5] Diallo Abdoulaye Kindy and Al-Sakib Khan Pathan ,"A Detailed Survey on Various Aspects of SQL Injection in Web Applications: Vulnerabilities, Innovative Attacks, and Remedies,2012
[6] Isabel Nájera, "Literature review on Software Security", 004C RWG, Eric Walker April, 2011.
[7] Jan Jantzen DTU, Automation, Bldg 326, 2800. Kongens Lyngby, DENMARK. Tech. report 98E, revised 22, March 2006.
[8] Nicole Duff and Huiming Yu, "Tutorial on SQL Injection Attacks", Department of Computer Science, North Carolina, A&T State University.
[9] Mayank Namdev,Fehreen Hasan, Gaurav Shrivastav , "A Novel Approach for SQL Injection Prevention Using Hashing & Encryption", International Journal of Computer Science and Information Technologies, Vol. 3 (5), 2012, ISSN: 0975-9646, pp 4981 – 4987, 2012.
[10] Mayank Namdev , Fehreen Hasan, Gaurav Shrivastav , " Review of SQL Injection Attack and Proposed Method for Detection and Prevention of SQLIA" , International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 7, July 2012, ISSN: 2277 128X, 2012.
[11] M. Roslinmary, S. Sivasakthi, A. Shenbaga Bharatha Priya , "An Effective Approach for Detecting and Preventing Sqlinjection Attacks", International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014.
[12] Munqath H. Alattar S.P. Medhane, "Efficient Solution for SQL Injection Attack Detection and Prevention", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume Issue-1, March 2013.
[13] Rohilla, Pradeep Kumar Mittal, "Database Security by Preventing SQL Shelly Injection Attacks in Stored Procedures", IJARCSSE, pp 915, 2013.
[14] Shabnam R. Makanadar1, Vaibhav V. Solankurkar , "An Approach to Detect and Prevent SQL Injection Attacks using Web Service,"International Journal of Science and Research(IJSR) ,ISSN: 2319-7069,Volume 2,Issue 4,pp 242,April 2013.
[15] Shanu Verma, Urvashi , "SQL Injection Attack on Data and Prevention through Hashing", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (1), 2014, ISSN: 0975-9646, pp-51-54, 2014.
[16] Shaukat Ali,Azhar Rauf,Huma Javed, SQLIPA , "An Authentication Mechanism Against SQL Injection", European Journal of Scientific Research, ISSN 1450-216X, Vol.38 No.4 (2009), pp 604-611, 2009.
[17] Shubham Srivastava, Rajeev Ranjan Kumar Tripathi2 , "Attacks Due to SQL Injection & Their Prevention Method For Web-Application", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (2), ISSN: 0975-9646, pp 3615-3618, 2012.
[18] Rahul Johari,Pankaj Sharma , " A Survey On Web Application Vulnerabilities Exploitation and Security Engine for SQL Injection", 2012 International Conference on Communication Systems and Network Technologies, 978-0-7695-4692-6/12,IEEE, pp 243, 2012.
[19] V.Nitya, R.Regan, J.Vijayaraghavan, "A Survey on SQL Injection attacks, their Detection and Prevention Techniques", IJECS, Vol. 2, ISSUE 2013,pp 886-905,2013.
[20] William G.J. Halfond and Alessandro Orso , "AMNESIA: Analysis and Monitoring for Neutralizing SQL Injection Attack's", ACM 1581139934/05/0011 ,2005.
[21] William G.J. Halfond, Jeremy Viegas, and Alessandro Orso, "A Classification of SQL Injection Attacks and Countermeasures", IEEE, 2006.
[22] https://www.owasp.org/index.php/ Top_10_2010-Main
[23] "Web Application Vulnerabilities and Avoiding Application Exposure", F5Networks.