Security Techniques in Mobile Agents: A Review

[1]Mahesh Kumar, [2]Jyotsna Parmar
[1]M.Tech Research Scholar, [2]Assistant Professor
[1,2]Department of Computer Engineering
[1,2]J.C. Bose University of Science and Technology YMCA, Faridabad, India

Abstract: This paper presents various security mechanisms and issues related to web application security. It describes the various security approaches and mechanism applied to detect and prevent SQLi (SQL Injection). Security is one of the major issues with web applications as they can be attacked by malicious users which increase the complications with data privacy and data protection. The paper analyses the security threats and the countermeasure of SQLi. The primary issue in the security of web application system is to protect agent from malicious attack launched by the intruder. It presents possible SQL injection threats to the Client-Server-Database (DB) paradigm and distinguishes between detection and prevention security mechanisms. The main intention of this work is to analyze the different detection and the prevention mechanisms for SQL injections. In our research, we will discuss about the SQL injection and how are these implemented followed with the solution given in literature and their comparative study.

***Index Terms* – SQL injection, SQLi, Detection, Prevention, Database, Security.**

1. Introduction

Due to the advent of affordable sensor-based hardware and exponential growth in wireless technologies, the areas of IOT and cloud are gaining a lot of popularity these days. Technologies like IOT foresees the interconnection of heterogeneous devices like smart phones, wearable sensors, smart appliances and other internet-enabled devices. For the successful implementation of these technologies, security still lies as one of the biggest challenge as the applications of these technologies are inordinate but it surely increases the vulnerability of user data. So, the concept of cyber security is a necessity for the wide acceptance of these technologies.

"Cyber security is the practice of protecting systems, networks, and programs from digital attacks. These cyber attacks are usually aimed at accessing, changing, or destroying sensitive information; extorting money from users; or interrupting normal business processes."[1]. Cyber security is a sub domain of network security which deals with the digital as well physical protection of data from unauthorized access within a network. Out of myriad techniques of exploitation ranging from simple Wifi password guessing to the very dangerous and creative cross site scripting, SQL injections come near the end of dangerous line and what makes these attacks dangerous is an inevitable solution of data management we know as databases.

Web applications mostly use 3-tier architecture for serving the required information to the user. This architecture consists of a *client*, *server* and *DB (database)*. The client sends a request to the server, the server then sends the query to the database to fetch data and provides the required result to the user, as shown in figure 1. In this architecture, the server makes use of structured query language (SQL) to communicate with the database. The SQL injection attacks make use of these SQL queries to perform some illegal actions like unauthorized access to a web page, stealing or modifying information, destroying a website etc. It typically sends some malicious query to the database in order to get some result, which only authenticated users can get. It is called SQL injection.

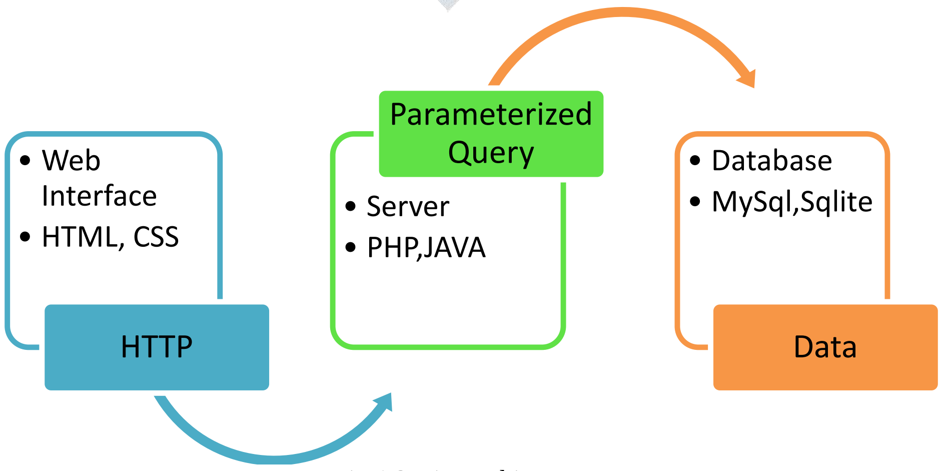

Fig 1 3-Tier architecture

SQL Injection refers to an injection attack where an attacker can send malicious SQL statements to a web application's database by manipulating the input data sent by a web browser to a server to get some information.SQL injections are basically used to get hold of the database to unauthorized data, login credentials, and privilege escalation and so on. This attack can cause serious harm to the web applications by accessing database of that website and since an SQL Injection vulnerability affects any website or web application that uses an SQL-based database, the vulnerability is one of the oldest, most prevalent and most dangerous second only to cross site scripting (XSS) having first case in 1998 [2],[3].Next section puts a light on the various types of SQL injection attacks in detail.

2. TYPE OF SQL INJECTION ATTACKS

Most of the common and dangerous methods used in SQLi are described below [4].

2.1 Tautology

This type of attack makes use of some statement that is always true in its logical form like 1=1 or 'a'= 'a' to bypass the authentication step.It makes the use of 'where' clause in the statements. A typical example can be a login page in which username and password are required as authentication step. If a hacker enters password=0000 and username such as 1' or '1 = 1'--, the query becomes

SELECT ∗ FROM user WHERE username = '1' or '1 = 1'--' AND password = '0000'.

Now, the string following '--' becomes a comment and '1 = 1' is always true, and in this way the authentication step is bypassed.

2.2 Comment out

This type of attack is used with tautology to bypass the required login for the access of a website or a webpage and is useful in piggybacked as well as blind attacks. It makes use of the comment out character "--" as the Statement following "--" is not executed as it is treated as acomment by the Database. The attacker will intentionally insert this character into the SQL statement, so that legitimate SQL query is not fully executed. For example: A login page requires Username and password to access a webpage or website. So, to bypass this requirement and to get the unauthorized access of the webpage, the attacker may input the Username as*"Mahesh" --"*
The server will execute the statement

*"SELECT * FROM users WHERE username =" Mahesh" --" and password = "0000"*

But this statement will not be normally executed as the values after '--' will be seen as a comment by the DB and the attacker will get the unauthorized access of the required webpage.

2.3 Union attack

In this attack, to perform an illegal action, two or more independent queries are sent by the attacker. As the name says, this attack makes use of UNION operator. For example,

*SELECT * FROM user WHERE username='mahesh'* ***UNION*** *SELECT * FROM member WHERE id='admin' --' AND password='1234';*

The query after the -- will be neglected and result will be the administration information of the DB.

2.4 Piggy backed

In this type of attack, the attacker inserts extra SQL queries into a valid one to perform some illegal action. It makes use of the semicolon operator '';''. For example, if an attacker uses ";" after each query, the DB sees the next line as a different query to execute.

*SELECT * FROM users WHERE id='admin' AND password='1234'; DROP TABLE user; --';*

And, this action ultimately deletes the user table.

2.5 Illegal/Incorrect queries

The attacker can sometimes deliberately insert illogically incorrect queries to produce an error. An error can be very useful sometimes as it can show internal schema, table names, and some critical information about the database. Even the simplest error can show the version name and database used. The attacker can use such type of information to conduct malicious activities in the future.

2.6 Blind injection

Blind injections are of broadly two types (Time based and Boolean based) and are mostly used for finding loopholes or vulnerabilities for the injection point in websites. Due to some preventive measures used for error handling, there can't be too much information about the injection point so in order to find and exploit that point, blind injections are used. They work like asking a question from database and the website returns result based on the options given with that question. For example in time based injections, a database can return result immediately if condition A is fulfilled or can add delay if condition B is met. Blind injections are usually automated. Next section puts light on different techniques available for the detection of possible SQLi attacks.

3. DETECTION TECHNIQUES

In this section, it describes the literature review of security mechanism for detection of security breaches against SQL injections.

3.1 AMNESIA [W. G. Halfond and A. Orso. 2005] [5] The goal of AMNESIA (Analysis and Monitoring for Neutralizing SQLIA) is to detect as well as prevent most of SQL injections. It successfully stops SQL injections by using dynamic combined with static approach. A static model is built for the analysis of allowed queries that an application can produce. In dynamic part of AMNESIA, every input query is checked with the static model of queries provided earlier. So any input query is either allowed to go to database or discarded based on the static model analysis of that query. Unlike many approaches, no input tokenization of the requests is done in this method. It addresses almost all types of SQL injections except stored procedures. Problem with this tool is that the accuracy depends on the model built because for each web application, there has to be a different model. It raises false positives and false negatives sometimes and AMNESIA is only useful for web applications built on JSP.

3.2 CANDID [P. Bisht, P. Madhusudan, and V. N. Venkatakrishnan 2010][6] Candidate evaluation for Discovering Intent Dynamically (CANDID) is a tool that records the programmer-intended SQL query structure from the real client and compares it with the query structure generated with the hackers input. Tokenization and encoding of input as well as identification of source is done in CANDID. It uses static analysis to detect and prevent SQLi but does not provide any guarantee for the same. Main drawbacks of this method that for a big website, it is not feasible to develop all the legitimate inputs and a through developer learning is also required.

3.3 Parameters removal [Rajsree A. Katole et al.. 2018] [7]This method works on the basis of comparison between static and dynamic queries. Parameters of query are deleted at the run time by using a delete() function. This dynamically generated query is then compared with the predefined original static query. If there's a difference between static and dynamic queries, it means that there is a risk of SQL injection. This technique only performs the detection not the prevention of SQL. Another drawback of this approach is that with alternate encoding in big web applications, there is a chance of lots of false positives due to attacks such as alternate encoding.

3.4 Machine learning method using Bayesian algorithm[**Cheon E., Z. Huang and Y. Lee 2013][8]** This approach uses the input source code validation by capturing POST and GET request by the user. A website developer's defined training dataset is used that represents structure of malicious queries that can happen. A converter works in association with a classifier in this approach. Converter sends a numeric value of length and keywords present in POST or GET request from user to the classifier which in turns compares the probability of trained and the dynamic query. If both the probabilities are same, it mean that there is a probability of SQL injection and the request is dropped. Although this method addresses almost every type of attacks but it doesn't provide any guarantee of the detection and prevention and the effectiveness of the approach is highly dependent on the user defined trained dataset so it varies from person to person. No source identification and key management infrastructure are also some issues here.

3.5 Machine learning using trained data set [Joshi et al. 2014] [9] This approach uses the same machine learning approach as others but the main difference is the entropy as a comparison factor instead of the probability because it produces better results. In the calculation of entropy, data is categorized instead of constant value so it produces a great effect in proportion to the change in SQL query. Entropy in this method calculates the average information needed in the identification of the class of a training data set. Source code of application is analyzed to calculate the entropy of static query and then it is matched with the user input dynamic SQL query. If both have the same entropy, query is allowed to pass else blocked. Main disadvantage is like all other machine models, a highly skilled dataset is needed to define a model for comparison as poor datasets can produce false positives and false negatives. Though it provides the automatic detection yet prevention is semi automatic in this approach.

3.6 Negative tainting with syntax awareness [A. Alazab , A. Khresiat 2016] [10] Authors provide a hybrid approach for the SQL injection problem by mixing the negative tainting with the syntax awareness. In this approach, the database is the part where negative tainting is used to identify the malicious and un-trusted data. By the taint analysis, this method can find whether any data related to database is trusted or un-trusted. The web application layer make use of the syntax aware approach to find about the length and validation of the input query string so that the malicious queries can be restrained before going to the database for further damage. This approach do not even change the existing structure of the web application means it does not require any extra infrastructure. After all the advantages, overhead is an issue in this approach due to mixed approach at front and end level.

3.7 PHP Data Object (PDO) [M. Sendiang et al. 2016] [11] PDO or PHP data objects are used as a technique and have become quite famous in preventing the SQL attacks over the years. PDO is an extension that provides interface to the database connected to PHP and provides security to the system. The parameterized queries methods are used in PDO which gives the parameters of the query to the database only at the runtime so any malicious query will act as string during execution and will produce error. This method successfully tackles tautology and union attacks but is helpless against other type of attacks. Another drawback is that this is an extension for PHP only.

3.8 ASCII base string matching[Indrani Balasundaram and E. Ramaraj 2011] [12] Authors proposed a method extending the work of Su and Wasserman [13] in 2011 by proposing a technique that uses ASCII values of the data stored in the database. Each data has its own ASCII value and is stored in a column next to the data. The SQL input query parameters will be converted into ASCII and checked against saved ASCII values of data. If both are same, then data will be provided else query is discarded. This is efficient in securing critical data but makes the process slow and requires more storage. Data with critical value is stored with the ASCII value next to it. Every time there is need to get data from the critically flagged table, the user input value will be converted in ASCII and compared with the stored value in against the entry thus increasing overhead.

3.9 RegEx and escape strings [Benfano Soewitoa et al. 2018] [14] Method proposed by authors is thatthere is a filter on the http requests in the form of Regular Expressions and those Urls which qualify for the criteria, will be executed else there will be an error. Another idea was to use MYSQL escape string method to modify the incoming query to make it illegal if there is any undesirable character present. Escape method makes any user entry as a string by adding an escape character. So any entry with predefined characters in list will be automatically converted to string and the database will result in error. This method is useful for the newest frameworks like Ruby on Rails[15] and Django[16] where requests are handled in the form of Regular expressions but it'll be ineffective for forums based web applications that cannot use URL to send data. One more disadvantage is that the escape string method is for MYSQL only.

3.10 Behavior and response[Zeli Xiao et al. 2017] [17] Model can detect SQL attacks by analyzing previous history of the user in many phases starting fromextracting the predefined URL and its SQL query to map between query and URL. Critical or sensitive data sources are defined and are monitored constantly. Execution results of each SQL query is recorded for the future comparison that includes user information (like IP address), execution time and error status. The execution history is thus used as a base for comparison if there are changes beyond threshold or not. If great changes are experienced, user is flagged and requests are blocked. Disadvantages of this method go from false positive to the overall overhead and extra requirement of memory and processing power.

3.11 Expectation criteria [Linghuan Xiao et al. 2016] [18] Authors suggested a theoretical approach to detect the SQLi on the basis of intention orientation and expectation criteria by defining a function that analyzes the query on the basis of classification of statements defined already. Intention defined by them is the percentage of harmful code present in the input and more the percentage, more the intention of the user is to access the unauthorized data. Expectation is defined for a discrete random variable V as the probability-weighted average of all possible values that this variable can have with different probabilities. This approach works on the basis of predicting the intention taking consideration of different factors like attack string probability, special character's probability and expectation. Effectiveness of this approach is based on the attack data set described and the probability of a special character being malicious or not. When under looked, these things are found to be very dangerous.

4. Comparison of SQLi detection and prevention techniques

Comparison of different techniques is given below.

S.No	Author(s)	Year	Technique Used	Disadvantages
1.	W. G. Halfond, A. Orso	2005	Static and dynamic analysis	Effectiveness depends on model created for comparison Provides false positives Only for web applications built on JSP
2.	P. Bisht et al.	2010	Static analysis	Developer learning needed Big websites need bigger models thus more complex models Broadness and accuracy of model defines the effectiveness
3.	Indrani Balasundaram, E. Ramaraj	2011	String matching	Storage and processing power constraint Overhead due to conversion and comparison Not feasible for non-critical and high amount

				of data
4.	Cheon E et al.	2013	Machine learning	No source identification Key management infrastructure Accuracy depends on trained data set
5.	Joshi et al.	2014	Machine learning	Semi automatic prevention No guarantee of successful resolution of SQLi False positives
6.	A. Alazab , A. Khresiat	2016	Negative taint analysis	Does not address modern type of SQLi Overhead due to comparison at both web and database interface
7.	M. Sendiang et al.	2016	PDO	Does not address attacks other than tautology and union Only for PHP
8.	Linghuan Xiao et al.	2016	Expectation criteria	Overhead due to probability calculation and comparison of different vectors No guard for modern SQLi
9.	Zeli Xiao et al.	2017	Behavior and response	Too complex to implement for websites with increased users as it is not feasible to track every user Memory and processing constraints
10.	Benfano Soewitoa et al.	2018	Regular expression and escape strings	Only works for data submitted through URL False positives due to even little deviation Escape string function works only for MYSQL
11.	Rajsree A. Katole et al..	2018	Static and dynamic comparison	Only detection no prevention Does not address alternate encoding False positives

5. CONCLUSION

According to OWASP, injections are the most prevalent and most dangerous attack methods used for the hacking of web applications [19]. We have discussed many proposals and methods presented by various brilliant minds over the years. In our survey, we have presented the history, types of SQLi, their effects on web applications and many techniques to stop these over a period of almost 13 years. Many of the techniques we have discussed are unable to deal with stored procedures or modern types of injection which are a combination of different type of cyber attacks mixed with SQLi example SQLi+CSS, SQLi+DDos [20].

This paper gives an overview about the security techniques of web applications against attack from SQL injections. Some of those techniques are still at the theoretical level and are not yet widely used in practice. None of the existing techniques provides an optimal solution for all scenarios. However, a combination of various techniques may yield powerful solutions.

6. REFERENCES

[1] https://www.cisco.com/c/en/us/products/security/what-is-cybersecurity.html

[2] https://www.vice.com/en_us/article/aekzez/the-history-of-sql-injection-the-hack-that-will-never-go-away

[3] Justin Clarke, “SQL injection and defense”, Elviser Publication. Second Edition, 2012

[4] Sun, S.T, Wei T.H., Liu, S. & Lau, S., (2007). “Classification of sql injection attacks. University of British Columbia, pp.1–6.”

[5] W. G. Halfond and A. Orso. AMNESIA: Analysis and Monitoring for NEutralizing SQL-Injection Attacks. In Proc. of the IEEE and ACM Intern. Conf. on Automated Software Engineering (ASE 2005), pages 174–183, Nov. 2005.

[6] P. Bisht, P. Madhusudan, and V. N. Venkatakrishnan, “CANDID: Dynamic Candidate Evaluations for Automatic Prevention of SQL Injection Attacks”, ACM Transaction on information System Security, pages.1–39, 2010.

[7] Rajashree A. Katole, Swati S. Sherekar, Vilas M. Thakare, “ Detection of SQL Injection Attacks by Removing the Parameter Values of SQL Query, Proceedings of the Second International Conference on Inventive Systems and Control (ICISC 2018) IEEE Xplore Compliant, ISBN:978-1-5386-0806-7.

[8] Cheon, E., Z. Huang, and Y. Lee 2013. “Preventing SQL Injection Attack Based on Machine Learning”. International Journal of Advancements in Computing Technology. 5(9): 967-974

[9] Joshi, A., and G. V. 2014. SQL Injection Detection Using Machine Learning. IEEE International Conference on Control, Instrumentation, Communication and Computational Technologies. 1111-1115

[10] A. Alazab , A. Khresiat , “ New Strategy for Mitigating of SQL Injection Attack”, International Journal of Computer Applications (IJCA), Volume 154, paper No.11, November 2016

[11] M. Sendiang, A. Polii, J. Mappadang, “Minimization of SQL Injection in Scheduling Application Development”, International Conference on Knowledge Creation and Intelligent Computing (KCIC), IEEE, Indonesia, November 2016

[12] Indrani Balasundarama , E. Ramaraj, “An Efficient Technique for Detection and Prevention of SQL Injection Attack using ASCII Based String Matching”, in the International Conference on Communication Technology and System Design 2011

[13] Z.Su and G. Wassermann, “The Essence of Command Injection Attacks in Web Application,” in the 33rd Annual Symposium on Principles of Programming languages, 2006, pages 372-382.

[14] Benfano Soewitoa , Fergyanto E. Gunawanb, Hirzic , Frumentiusa, “Prevention Structured Query Language Injection Using Regular Expression and Escape String”: 3rd International Conference on Computer Science and Computational Intelligence 2018 : Elsvier Publications, Procedia Computer Science 00 (2018) 000–000 678–687).

[15] https://guides.rubyonrails.org/

[16] https://docs.djangoproject.com/en/2.2/intro/overview/

[17] Zeli Xiao, Zhiguo Zhou, Wenwei Yang and Chunyan Deng , “An Approach for SQL Injection Detection Based on Behavior and Response Analysis” at 2017 9th IEEE International Conference on Communication Software and Networks, 978-1-5090-3822-0/17/2017 IEEE

[18] Linghuan Xiao, Shinichi Matsumoto, Tomohisa Ishikawa, Kouichi Sakurai, “SQL Injection Attack Detection Method using Expectation Criterion”, 2016 Fourth International Symposium on Computing and Networking- 2016 IEEE 2379-1896/16

[19] https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf

[20] Zainab S. Alwan et al., International Journal of Computer Science and Mobile Computing, Vol.6 Issue.8, August- 2017, pg. 5-17.