Volume 5, Issue 3, March 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Survey of SQL Injection Attacks

Udit Agarwal
Asso. Prof., RBMI,
Bareilly, India

Monika Saxena
Asst. Prof., RBMI,
Bareilly, India

Kuldeep Singh Rana
Asso. Prof., RBMI,
Bareilly, India

Abstract- Due to the huge growth in the need for using Web applications worldwide, there have been huge efforts from programmers to develop and implement new Web applications to be used by companies. Since a number of these applications lack proper security considerations, malicious users will be able to gain unauthorized access to confidential information of organizations.
SQL injection attack (SQLIA) is the most common attack in websites in these days. Some malicious codes get injected to the database by unauthorized users and get the access of the database due to lack of input validation. Input validation is the most critical part of software security that is not properly covered in the design phase of software development life-cycle resulting in many security vulnerabilities. This paper presents the techniques for detection and prevention of SQL injection attack. There are no any known full proof defenses available against such type of attacks. In this paper some predefined method of detection and the some modern techniques of preventions are discussed.

Keywords: SQL–injection attacks, prevention, detection, Web Application

I. INTRODUCTION

The widespread adoption of the Web as an instant means of information broadcasting and various other transactions, including those having financial consequences, has essentially made it a key component of today's Internet infrastructure. These applications and their underlying databases often store confidential or even sensitive data. The potential downtime and damages that could easily amount to millions of dollars have also prohibited many mission critical applications from going online, which could greatly benefit the users. Hence, it is crucial to protect these applications from targeted attacks. An SQL injection attack targets interactive web applications that employ database services. These applications accept user inputs and use them to form SQL statements at runtime. During an SQL injection attack, an attacker might provide malicious SQL query segments as user input which could result in a different database request. By using SQL injection attacks, an attacker could thus obtain and/or modify confidential/sensitive information. An attacker could even use a SQL injection vulnerability as a rudimentary IP/Port scanner of the internal corporate network.
SQL-Injection Attack (SQLIA) constitutes an important class of attacks against web applications. By leveraging insufficient input validation, an attacker could obtain direct access to the database underlying an application. Any web application exposed on the Internet or even within a corporate intranet could therefore be vulnerable to SQLIAs. Although the vulnerabilities that lead to SQLIAs are well understood, they persist because of lack of effective techniques for detecting and preventing them. Defensive programming could, to an extent, protect against certain threats of SQLIAs. But it is impractical to undergo this entire process for protecting legacy systems. Several solutions have been proposed in literature to prevent SQLIAs in the application layer, which combine static analysis of application level programs and runtime validation of dynamically generated SQL-queries with inclusion of user inputs.

II. OVERVIEW OF SQL INJECTION ATTACK

While Typing SQL keywords and control signs an intruder is able to modify the structure of SQL query developed by a Web designer. SQL Injection is a kind of web application security exposure in which an attacker is able to expose a database SQL command, which is executed by a web application. SQL Injection attacks can take place when a web application utilizes user-supplied data without proper validation or encoding as part of a command or query. SQL injection attacks are nothing but injecting malicious queries by the hackers into the application projected queries to get the desired outputs from the database.SQL Injection allows an attacker to create, read, update, modify, or delete data stored in the back-end database. Thus, SQL injection exploits security vulnerabilities at the database layer.

III. SQL INJECTION ATTACK TYPES

There are various methods of attacks that depending on the goal of attacker are performed together or sequentially and its classification is given below:-
Tautologies: This type of attack injects SQL tokens to the conditional query statement to be evaluated always true. This type of attack used to avoid authentication control and access to data by exploiting vulnerable input field which use WHERE clause. "SELECT * FROM employee WHERE userid = '211' and password ='bbb' OR '1 '='1' as the tautology statement (1=1) has been added to the query statement so it is always true.

Illegal/Logically Incorrect Queries: When a query is not needed, an error message is returned from the database including useful debugging information. This error messages help attacker to discover vulnerable parameters in the application and consequently database of the application. In fact attacker injects junk input or SQL tokens in query to produce syntax error, type mismatches, or logical errors by reason.

Union Query*:* By this method, attackers join injected query to the safe and sound query by the word UNION and then can get data about other tables from the application. The output of this attack is that the database returns a dataset that is the union of the results of the original query with the results of the injected query.

Piggy-backed Queries: In the piggy-backed Query attacker tries to add on additional queries to the original query string. In this method the first query is original whereas the following queries are injected one. Here the intruders exploit database by the query delimiter, such as ";", to append extra query to the original query. With a successful attack database receives and execute a multiple different queries.

Stored Procedure: In this technique, attacker focuses on the stored procedures which are present in the database system. Stored procedures run directly by the database engine. Stored procedure is nothing but a code and it can be vulnerable as program code. For authorized/unauthorized user the stored procedure returns true/false.

Inference: By this type of attack, intruders change the behavior of a database or application. There are two well known attack techniques that are based on inference: blind injection and timing attacks.

Blind Injection: At times developers hide the error details which help attackers to compromise the database. In this situation attacker face to a generic page provided by developer, instead of an error message. So the SQL Injection Attack would be very difficult but not impossible. An attacker can still steal data by asking a series of True/False questions through SQL statements.

Timing Attacks: A timing attack lets attacker gather information from a database by observing timing delays in the database's responses. This technique by using if-then statement cause the SQL engine to execute a long running query or a time delay statement depending on the logic injected. This attack is similar to blind injection and attacker can then measure the time the page takes to load to determine if the injected statement is true. This technique uses an if-then statement for injecting queries. WAITFOR is a keyword along the branches, which causes the database to delay its response by a specified time.

IV. SQL INJECTION PROCESS

The attacker can gain access to web applications using several methods. Through the web application's input fields or hidden parameters the attackers adds SQL statement to access to resources is known as SQL Injection Attack (SQLIA). Due to the lack of input validation in web applications hackers can be successful. Injecting web applications means having illegal access to data stored in database

A. Normal Process in Web Application

In normal user input process in web application, user sends request by providing user inputs to the application server. The application server creates the SQL query statement. This SQL statement is submitted to the backend database. The result is fetched from the database and given back to the user.Fig.1 shows the normal User input process in web application.

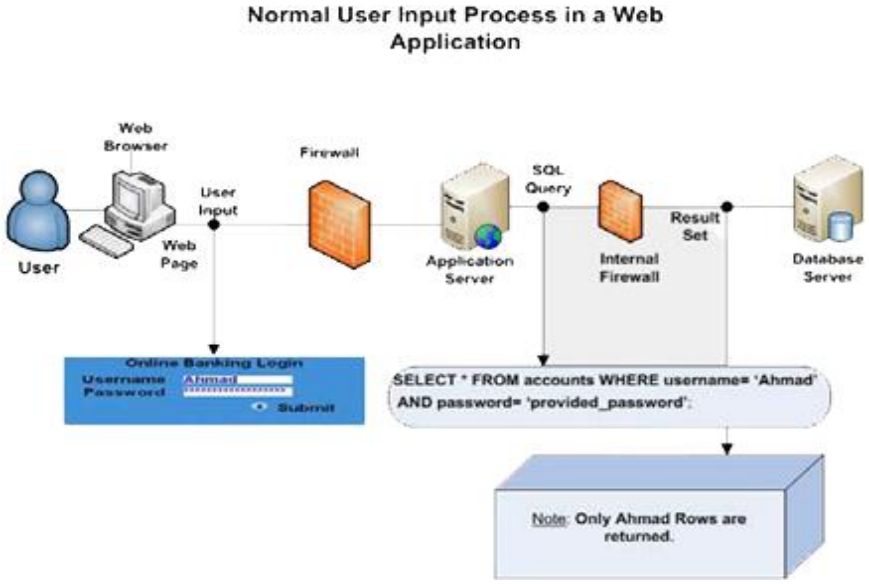

Fig.1 Normal User Input Process in Web Application

B. Malicious Input Process in web Application

In SQLIA, attacker enters malicious input in the input field for example in fig. a attacker enters username as Ahmad OR 1=1- and password as not needed. Because of this malicious input SQL query is altered which is always evaluated to be true. The result of such query will return all the rows of the table.Fig.2 shows malicious input process in web application.

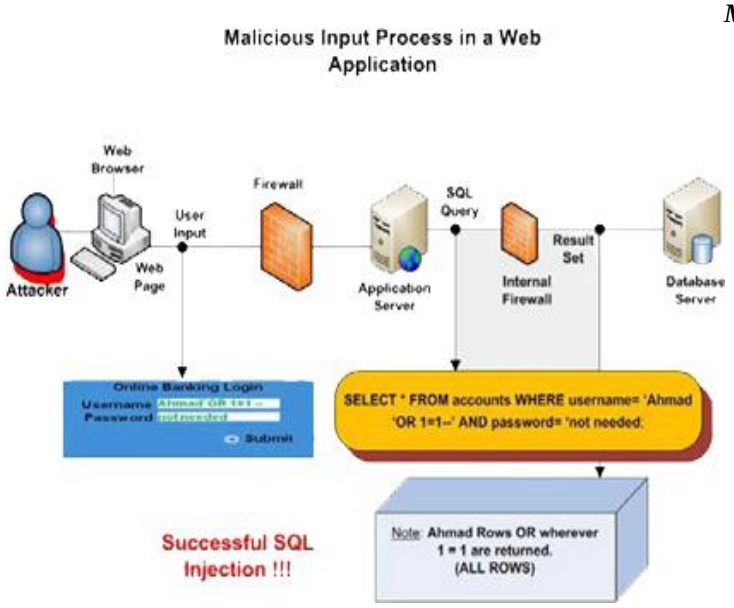

Fig.2 Malicious Input Process in Web Application

V. SQL INJECTION PREVENTION MECHANISM

There are number of techniques available in literature to address SQLI attacks. Here we review all the techniques briefly with their advantages and disadvantages:

A. Defensive Coding Practices

The defensive coding is for the developer who is responsible for developing the web application. As the coding practice is very much prone to human error, developers always give the extra effort to code safely. The root cause of SQLI is the insufficient input validation and sometimes developers forgot to add checks or did not perform adequate input validation. So there are various guide lines proposed to fix this problem

1) Input type checking

SQLI attacks can be performed by injecting commands into either a string or numeric parameter. A simple check of such inputs can prevent many attacks.

2) Encoding of inputs

Injection into a string parameter is often accomplished through the use of meta-characters that trick the SQL parser into interpreting user input as SQL tokens. So the solution is to use the functions that encode a string in such a way that all meta characters are specially encoded and interpreted by the database as normal characters.

3) Positive pattern matching

Input validation should be able to identify all good inputs as opposed to all bad inputs. Because the negative validation is not always possible due to the new type of attack signature. So better solution is to implement the positive validation.

4) Identification of all input points

Developers must check all input points to their application. There are many possible sources of input to an application. If used to construct a query, these input sources can be a way for an attacker to introduce an SQLIA. Simply put, all input sources must be checked.

B. Black Box Testing

The web vulnerability scanner are used for the black box testing, the vulnerability scanner are used for finding the loop holes in the existing application. The vulnerability scanner mainly visits the web application's input point and simulates the attack against and if the attack is possible or made success then it summarizes it in the form of a report.

C. White Box Testing

The static code analyzers are used for the white box testing, the static code analyzers basically analysis the byte code of the web application with the intension of finding the vulnerability.

D. Run Time Monitoring

For the run time monitoring IDS (Intrusion detection System) can be used, the IDS system is based on a machine learning techniques that is trained using a set of typical application queries. The technique builds models of the typical queries and then monitors the application at run time to identify queries that do not match the model.

VI. CONCLUSIONS

In this paper various types of SQL injection mechanism, detection type and prevention techniques are discussed. There is no one complete infallible solution to database security and have some issues hard to eliminate. Any organization that attempts to secure a database system, must consider the security of the overall environment including the communication channel, user access methods, the database, and any application which is used to access the database.

REFERENCES

[1] Priyanka, Vijay Kumar Bohat," Detection of SQL Injection Attack and Various Prevention Strategies**",** International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-2, Issue-4, April 2013

[2] Sonam Panda, 1 Ramani, "Protection of Web Application against Sql Injection Attacks", International Journal of Modern Engineering Research (IJMER) Vol.3, Issue.1, Jan-Feb. 2013 pp-166-168 ISSN: 2249-6645

[3] Mihir Gandhi, JwalantBaria, "SQL INJECTION Attacks in Web Application "International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-6, January 2013

[4] Mayank Namdev, Fehreen Hasan, Gaurav Shrivastav "Review of SQL Injection Attack and Proposed Method for Detection and Prevention of SQLIA" Volume 2, Issue 7, July 2012.

[5] Shubham Srivastava1, Rajeev Ranjan Kumar Tripathi, "Attacks Due to SQL Injection & their Prevention Method for Web-Application ", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (2) , 2012,3615-3618

[6] S. Ali, SK. Shahzad and H. Javed, "SQLIPA: An Authentication Mechanism against SQL Injection", *European Journal of Scientific Research ISSN 1450-216X Vol.38 No.4 (2009),* pp 604-611.

[7] Inyong Lee , Soonki Jeong Sangsoo Yeoc, Jongsub Moond, "A novel method for SQL injection attack detection based on removing SQL query attribute", *Journal Of mathematical and computer modeling, Elsevier* 2011.

[8] 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,* pp.1–39, 2010.

[9] Kemalis, K. and T. Tzouramanis, "SQL-IDS: A Specification-based Approach for SQL injection Detection", *SAC'08. Fortaleza, Ceará, Brazil, ACM* , pp.2153 2158, 2008.

