Detection and Prevention of SQL Injection Using Auto Comparator

Sonal Mittal[1] and Garima Garg[2]

[1]M.Tech Scholar, SGI, Samalkha, Kurukshetra University, India
[2]Assistant Professor, SGI, Samalkha, Kurukshetra University, India

Abstract-**In today's world, most of the web applications are associated with database technology with the database as back-end to store the data. As a result, there are possibilities of SQL injection attacks on such applications. SQL Injection Attacks (SQLIA) are mostly performed on the Internet. By doing the SQL Injection attack on the website, the attacker is able to take control of the database and can manipulate the data from the database server of that website. Hence, it becomes the big challenge to secure such website against these types of attacks via the Internet. SQLIA is one of the top ten attacks according to Open Web Application Security Project (OWASP) but still there are no proper solution to this problem till now. Numbers of solutions have been discovered to deal with this attack, but the major concern is which solution is more convenient and provides faster access to the application without compromising the security. There are some existing solutions that are good in security but they are not efficient to handle large user's requests. In this paper we have given a brief introduction about SQLIA and proposed a method by which we can detect and prevent SQL Injection in the login phase using a single code.**

Keywords—SQLIA, Database, Comparator, Web Application

I. Introduction

These days the organizations are becoming more and more concerned about the employment of their website because of the development of World Wide Web. The web has become the essential need of our society. The wide spread use of Internet has led some malicious users to work in negative direction by attacking websites of organizations. These types of users are known as website attackers.

The lack of proper knowledge of software and secure engineering leads to vulnerabilities in web security, like inappropriate programming, etc. Some of the security solutions may prove to be effective, but the changes in technology can lead to new risks and challenges.

A database-driven Web application mainly has three tiers namely presentation tier, Business logic tier, data link tier.

1. Presentation tier: This layer is the front end of the web application. This layer interacts with other layers based on the inputs provided by the user. This layer validates and verifies the input properly.
2. Business logic tier: This layer processes the user requests. It involves server side programming logic. This layer acts as the intermediate between the presentation tier and the data link tier. The objective of this layer is proper checking of input and input neutralization.
3. Data tier: This layer contains the database server. It store and retrieves the data to and from database. This layer is used for input rectification.

SQLIA is one of the top ten attacks to the web application security according to Open Web Application Security Project (OWASP). SQLIA are easily understandable and exploitable, therefore this type of attack is easily used by attackers.

Figure 1 shows the basic database-driven Web application architecture

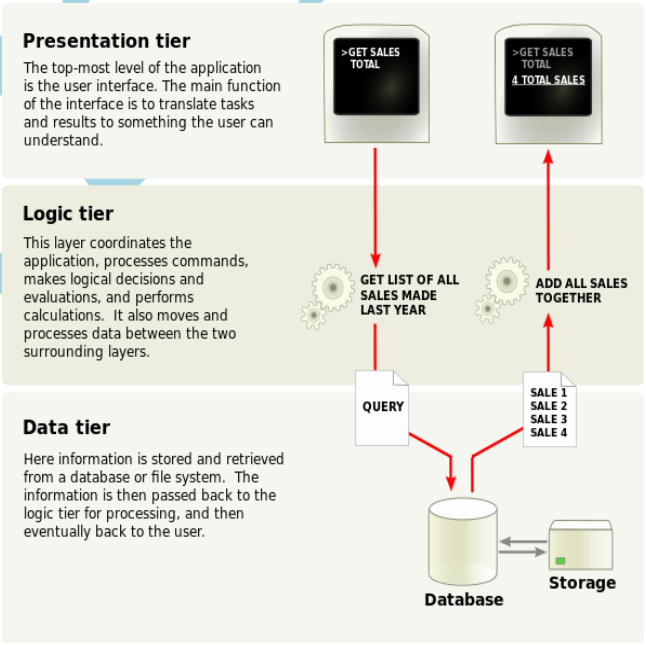

Figure 1

The traditional and most popular security systems like firewall, cryptography, encryption, Antivirus, intrusion detection systems, etc. having different security layers are not able to detect this type of attack. SQL injection attacks are becoming more and more common and they are easy to implement, so there exists a need to find an effective solution for this problem.

There are two types of approaches for SQLIA Detection:

Static Approach: This is also known as pre-generating approach. In this approach the programmers follows some guidelines for the detection of SQLIA during web application development. This approach also requires an effective validity checking mechanism for the input variable data.

Dynamic Approach: This is also known as post-generated approach. Dynamic approach is useful for the analysis of runtime or dynamic SQL queries, that are generated with user input data by the web application. In this post-generated approach, the detection techniques are executed before posting the query to the database server.

SQL Injection Attacks on the databases could be motivated by three objectives:

1. To hack the data from the database from which the data usually is not available.
2. To obtain the system configuration type data that would allow an attacker to build the profile.
3. To gain access to the organization's host computers through machine hosting the database.

II. SQL INJECTION

SQL injection is a technique where the attackers try to attack data driven applications. The attacker takes the advantage of poorly fitted escaped characters embedded in SQL statements into parsing the variable data from user input. The attacker injects arbitrary data in the form of a database query into a string that is eventually executed by the database through a web application, for eg. a login form.

SQL Injection Attacks are performed by submitting maliciously crafted input in the form of data or queries to database driven applications, such as interactive websites. These inputs are then used by applications to build dynamic SQL queries. Due to the lack of control on the data in SQL, these inputs are capable to alter the semantic structure of the query. There are numerous SQLIA techniques which are used by attackers. These techniques are based on the different statement structure combinations that are offered by SQL. Sometimes these techniques also take the advantage of DBMS implementations features like Microsoft's SQL Server. Their main aim is to extract the data from the database by allowing different SQLIA techniques. The resulting threats ranges from system fingerprinting to Denial-of-Service attacks and theft of confidential information.

SQL Injections Attacks thus affects the integrity, confidentiality, and availability of the data and structure in the databases and as a result it effects all the applications which are dependent on that database.

Let's see the example of basic SQL Injection:

Instead of submitting the inputs [user_login] and [user_password] in a website login form, the attacker enters [' OR 1=1 --] and [ ]. As a result, the following SQL query:

SELECT * FROM users WHERE login='user_login'AND pwd='user_password'

Becomes:

SELECT * FROM user WHERE login='' OR 1=1 –'AND pwd=' '

Here the attacker enters a single quote in its input login field followed by other characters. By doing this, the attacker closes the SQL login field in the Where clause, causing the SQL injection code right into the query. Since no login field can be blank, the attacker inserts the code OR 1=1, which will always evaluate to be true (also known as tautology). Next, the -- (double dash) operator denotes the starting of the comments, which tells the SQL parser to ignore the rest of the query including the password field.

As a result, the meaning of the altered SQL query will become equivalent to "select all users". Therefore, the application which is controlling the user authentication will authorize the attacker. In the worst case scenario, there is a possibility that the application returns an error message containing the data or the database details returned by the DBMS, i.e. the list of user credentials.

III. WAYS OF INJECTING CODE

The different ways used for injecting SQL statements in an applications are:-

1. **User Input:** The attackers inject the SQL Commands by providing properly crafted user input. Here, the attacker targets those web applications in which the user provides some information and then the request is processed.
2. **Order:** The attacker enters a malicious code in the form of string and enables the modified code to be executed immediately (direct attack) or by some other related activity (indirect attack). The attacker also manipulates the implicit functions by changing the values.
3. **Server Variables:** There are many server variables such as network headers, HTTP etc. which are used for knowing the usage of logging in and identifying the browser trends. The attacker can attack using these types of server variables when these variables are logged in the database.
4. **Database:** Here the attacker injects the attacks by manipulating SQL statements. This type of attack can be done through Basic Union Queries, Inference, and Piggy-Backed Queries and Tautology.
5. **Cookies:** Cookies are used to store information on client machine which is generated by web applications. If the Web application uses the content of the cookies to build the SQL queries, then an attacker can easily attack by embedding his code in the cookie.

IV. PROPOSED METHOD

There are some related methods for preventing SQL injection attacks which are used in the proposed method of auto comparator.

SQL Parse tree validation

A parse tree is a data structure for the parsed representation of a statement. We can parse a statement by using the grammar of the language of statements. We can find out if the two queries are equal by parsing the two statements and comparing their parse trees. When the attacker performs SQL injection into a database query, the parse tree of the original SQL query and the resulting SQL query do not match. Original SQL query means that when a programmer writes some SQL code to query the database, the programmer has got a formulation of the structure of the query. The hard-coded portion of the parse tree is the programmer supplied portion, and the empty leaf nodes in the parse tree are represented as user supplied portion. These

nodes represent the empty literals. The programmer intends the users to assign some values to these leaf nodes. A leaf node represents only one node in the resulting query. This node must be the value of a literal and it must hold the position where the holder was located.

Code Conversion Approach

1. Convert the User input to ASCII, binary,octal, hexadecimal etc. like codes.
2. Search the availability of the converted input in the data table and returns valid User_id and Password.

The proposed method consists of the Auto comparator using parse tree validation and code conversion method. We first accept the user input and compare it with the SQL query using parse tree validation. If there is a mismatch, then there is a possibility of an attack. If there is a possibility of an attack, then the user input is encoded into some other code and decode it afterwards.

The basic idea of this algorithm is as follows:

For checking the vulnerability and displaying the safe data:

- begin
- accept the user input
- compare the input with generalized SQL Query
- if
  - length of the parse tree mismatch
  - Possibility of an SQL injection attack
  - Encode the user input
  - Assign the encoded user input to any variable
  - set flag as 1
- else
  - the user input is safe
  - assign the user input to the any variable
  - set flag as 0
- display the value to be stored in database “variable”
- end

For decoding the coded input and displaying the data:

- begin
- if flag is 0
  - display the safe variable
- else
  - decode the encoded user input variable
  - display the decoded variable
- End

From the value of flag we can make out whether the user input is converted or not.

V. CONCLUSION

SQL injection attacks technique is a common technique that is used by the hackers to hack the important databases and extract the confidential data from them. The presence of these attacks have led to the development of some detection and prevention methods so as to disable such types of attacks so that the confidential information in the databases remains safe and are not hacked and exploited by the attackers. In this paper we have proposed a method to differentiate between the valid queries and invalid queries. The query is compared with the input and if they are not same, this leads to possible SQL injection attack. In this case the input is encoded and then decoded. This method can lead to detection and prevention of attack in single code only, but it has a drawback: The encoding and decoding of the user input by code conversion can lead to more processing time and more memory.

Since the available methods to detect and prevent SQL injection attacks are not sufficient to stop SQL injection attacks, therefore, in the present scenario many different methods are used to ensure higher security level of the databases and web applications.

REFERENCES

[1] Etienne Janot, PavolZavarsky, Preventing SQL Injections in Online Applications: Study, Recommendations and Java Solution Prototype Based on the SQL DOM, Concordia University College of Alberta, Department of Information Systems Security, 7128 Ada Boulevard, Edmonton, AB, T5B 4E4, Canada

[2] Ashish John, SQL Injection Prevention by Adaptive Algorithm, IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 1, Ver. III (Jan – Feb. 2015), PP 19-24

[3] Khaled Elshazly, Yasser Fouad, Mohamed Saleh, Adel Sewisy, A Survey of SQL Injection Attack Detection and Prevention, Journal of Computer and Communications, 2014, 2, 1-9

[4] DrR.P.Mahapatra and MrsSubi Khan, A Survey Of Sql Injection Countermeasures, International Journal of Computer Science & Engineering Survey (IJCSES) Vol.3, No.3, June 2012

[5] Priyanka, Vijay Kumar Bohat, Detection of SQL Injection Attack and Various Prevention Strategies, International Journal of Engineering and Advanced Technology (IJEAT)

[6] Diallo AbdoulayeKindy and Al-Sakib Khan Pathan, A Detailed Survey on Various Aspects of SQL Injection in Web Applications: Vulnerabilities, Innovative Attacks, and Remedies, International Journal

[7] AtefehTajpour , Suhaimi Ibrahim, Mohammad Sharifi, Web Application Security by SQL Injection DetectionTools, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 2, No 3, March 2012

[8] Ogheneovo, E. E. and Asagba, P. O., A Parse Tree Model for Analyzing And Detecting SQL Injection Vulnerabilities, Department of Computer Science, University of Port Harcourt, Nigeria.

[9] Gregory T. Buehrer, Bruce W. Weide, and Paolo A. G. Sivilotti, Using Parse Tree Validation to Prevent SQL Injection Attacks, Computer Science and Engineering The Ohio State University Columbus, OH 43210

[10] Navjot Verma, Amardeep Kaur, A Detailed Study on Prevention of SQLI attacks for Web Security,

International Journal of Computer Applications Technology and Research Volume 4– Issue 4, 308 - 311, 2015, ISSN:- 2319–8656

[11] ParveenSadotra, Hashing Technique - SQL Injection Attack Detection & Prevention, International Journal of Innovative Research in Computer and Communication Engineering, Vol. 3, Issue 5, May 2015

[12] Manveen Kaur, SQL Injection Attacks - Its Prevention using Flag Sequencing Approach, Computer Engineering and Intelligent Systems www.iiste.org ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online) Vol.6, No.2, 2015

