International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

A Survey on SQL Injection Attack Countermeasures Techniques

Aniruddh R. Ladole[1], D. A. Phalke[2]

[1]Department of Computer Engineering, D Y Patil College of engineering, Akurdi, Pune,

[2]Prof. Department of Computer Engineering, D Y Patil College of engineering, Akurdi, Pune,

Abstract: ***As users of internet is increasing day by day. The demands for web services and mobile web application are also increased. The probability of a system being attacked is also increased. All the web applications maintain information at the backend database from which results are retrieved. As these services or web application, can be accessed from anywhere around the world which needs to be always available to all the clients, partners employees, and for different users located at different parts of world. SQL Injection Attack is nowadays one of the topmost threats for web application security as it is the easier than other attacks. Using SQL Injection attackers can steal confidential information. In this paper has reviewed most of the SQL injection attacks detection systems proposed by different authors. This paper can be useful to other researchers for their work who plans to work in security of the database from SQL Injection attacks.***

Keywords: SQLIA, Information security, web application vulnerability, cyber security.

1. Introduction

Due to the more digitalization of world, usage of computers, mobile phones, tablets, etc is increasing very fast. With the boosting of internet, use of the web applications has also increased. Most of the web applications produced over recent years have a Presentation Tier, CGI Tier and Database Tier. All the dynamic web applications needs to be always available to all the clients, employees, and partners all around the world. As the development of information technology is increasing effectively, these web applications can be accessed from anywhere. Most of the web applications have different vulnerabilities in them, due to which they are possible to hack by combination of attacks of different types. These days the topmost threat for web application security is SQL Injection Attacks, many web sites are being attacked/hacked by attacker [1]. Using combination of attacks one can steal important information such as internet banking passwords, ATM pins, Users credentials from web applications and even corrupt the database. With the development of information technology, digitalization of world, massive amount of sensitive personal information and private information has collected in databases continuously. This data can be considered as the most valuable assets of organizations. As we know that as the economic value of data increases, attempts to hack the data also increases. SQL is abbreviated form of the Structured Query Language, it is a computer programming language used for querying relational database. It is a non procedural language. When extraction of information from database using SQL is done, this is termed as querying the database. It is an easy language to learn for writing queries, but it gains considerable complexity because it is a very powerful language with simple commands.

Basically Vulnerabilities for web application are classified into four types [2].

Type A: During development of the web applications, lack of knowledge of security measures by the developers.

Type B: Delay for testing or analysis of the application till the runtime phase, because of testing current variables rather than the source code in the analysis.

Type C: Type specification is not handled properly. Usage of the string and number is not defined properly.

Type D: The input validation of the user is not well defined. Inputs are not checked correctly. No restriction on the input provided by user.

Broadly categorizing the Database intrusion attacks, there can be two types [12], depending on the use of the database. In the first type, malicious users with legal user accounts directly access the database, and make ill-use of the structured query language (SQL), to steal the data. In the second type, attackers indirectly access the database using the vulnerabilities present in the database-driven web applications. That is, attackers by altering the original SQL statements attack the database through the user input values. There are seven types of SQLIA. Different authors have published survey or taxonomy for SQL Injection attacks detection and prevention in [2] [3] [4] [5]. In this paper some recent approaches are discussed.

The rest of the paper is organized as: the next section describes the all the Types of SQLIAs. Section 3 describes all the approaches used to detect SQLI attacks. Section 4 gives a brief comparison of each approach methods and its future scope. Section 5 gives the conclusion of the paper.

2. Types of SQL Injection Attacks

Following are the different types of SQLIA

1. **Tautologies:**
SQL injection queries are injected into one or more conditional statements such that these statements are always evaluated to be true. If a malicious user or attacker enters input such as '1' or '1 = 1' the query within the CGI tier becomes [5].

Query 1:
SELECT _FROM user WHERE id = '1'or '1 = 1' AND password = 'abcd'.
Because it is always evaluated as true, the "authentication step" is bypassed. Tautologies are only possible if input validation is not done properly.

2. **Logically Incorrect Queries:**
This type of attack Use logical error messages rejected by the database of the application to find useful data such as type , version of database, and names of tables, no. of columns, rows, etc that can be used for injection at the backend database [5].
Query 2:
SELECT * FROM user WHERE id= '1111' AND password= 'abcd' AND CONVERT(char, no) ';
This attack is used to collect the structure and information of CGI layer of website.

3. **Union Query:**
In this type of attack Injected query is joined with a secured query using the keyword UNION to perform unions between two or more queries, to get information related to other tables from the application [5]. For Example:
Query 3:
SELECT FROM user WHERE id= '123' UNION SELECT FROM member WHERE id= 'admin'--AND password='123abc';
All subsequent string after "--" is recognized as comment, and two SQL queries are processed by the CGI and database layer and results are returned to the user. In the above example, the result of the query process shows administrators information of the DBMS, if proper security is not provided.

4. **Stored Procedure:**
In many database user can store built-in stored Procedures for future use to save time. The attacker can executes these built-in functions/procedures using malicious SQL Injection codes [5].
Query 4:
CREATE PROCEDURE GetPasswordInfo

@pass varchar2,

AS exec(SELECT @pass = password, FROM Customers WHERE Id = @pass); GO

This type of procedures can be activated by the attacker on certain conditions, and are known as Stored Procedure Attacks. It is very hard to detect and prevent these types of attacks as it is at the database level.

5. **Piggy-Backed Queries:**
As in SQL many queries can be processed if the operator „;" is added after end of each query. Additional queries are inserted into an original safe query which can be injected with attack [5].
Query 5:
SELECT LastName FROM user WHERE id= '23' or address ='pune' ; DROP TABLE user;--

The result of query 5 is to delete the user table.

6. **Inference:**
An attacker makes logical conclusions from the answer to a true/false question about the database. This type of attack is used to predict the number of columns in the table [5].
6.1. *Blind Injection:*
By using this type of queries, attacker collects Information from the replies of the page after querying the server true/false questions.
for ex: when we search for some product in a website, we see something like following
URL: www.abcd.com/product?id=39
Injected query for above query will be:
www.abcd.com/product?id=39 order by 50--
The above query is translated in to following query
SELECT * FROM TABLE where id=33 order by 50;
Result of the query execution is this query will return always false if the number of columns in a table is less than 50. Query will return true if the number of columns are greater than 50. This attack will infer that error message that infers the information about the table names, no. of columns.
6.2. *Timing Attacks:*
In this type of attack by observing the response time (behaviour) of the database an attacker collects useful information. By using conditions attacker can set the time delay for short duration in SQL query. If the given condition is true, the specified delay takes place, during this delay attacker gains the information about the response time of the database.

7. **Alternate Encodings:**
These types of attacks try to avoid being identified by secure defensive coding also with automated prevention mechanisms. In this way it can help the attackers to evade detection. It can be combined with other attack techniques [5].
For example:
Select * from Product where prodid="; exec(char(0x73687574646f776e)) -- and password="123@abc";
Result of the query is the shutdown command. Hexadecimal value of the SHUTDOWN is passed to char() function. This code will execute SHUTDOWN command and bypass the input validation.

By using the combination of these attacks any hacker or attacker can get useful information about the database. Such as number of tables, columns, rows, etc. This information about database can be used for different attacks such as Cross site Scripting attacks, Denial of Services attacks, IP Spoofing, etc. to attack the web sites and applications.

3. Different techniques for SQL injection attacks prevention and detection

By combining available countermeasures SQL injection attacks can be stop. Web application developer should do defensive coding practices and code verification. All vulnerabilities of web application can be detected. In the testing detected vulnerabilities can be removed. Runtime attack prevention methods can be used to prevent SQL injection attacks.

Inserting New Network Mechanisms

Combining different software diversification techniques in various components can make web application more secure [6]. Authors propose to reduce massive reuse of code on server side and client side. Monoculture in designing of web applications leads to a problem in which if attackers attack one site then they can attack any websites. By using multitier diversified software"s one can prevent SQL Injection attacks.

Protecting web sites by using secure delivery networks [7]. Secure delivery network consists of four types of server"s viz. DNS server, edge server close to end server, parent server closed to origin, origin server. By using network layer controls, adaptive rate controls, application layer control, client reputation rules, OWAPS core rules at different server"s web sites can be protected from SQL injection attacks and other attacks, without performance degradation.
The Unicode system is used for Protection of the web sites in [8]. Verification is done for the trusted and non trusted data by keeping track of information as the data is propagated. By inserting Request Intercept wrapper and Database Driver Intercept wrapper into the typical framework of web application through reengineering, SQL injection attacks, and other attacks can be prevented.

Two tire Defense

Two tier Defense [9] against SQL Injection by Naresh Duhan and Bharti Saneja. The two tires used, first defense tire is generated by applying static validation constraints on input fields, and called as client side input validation. This validation includes type of data, string length, checking of the invalid characters in the input fields. The second tire of defense is Identity Base Cryptography (IBC), in this tier login encryption field values are compared with values stored in database, and if it is not matched then control goes to attack analyzer module. Future work for this approach is to integrating this approach with other types of web application attacks.

Signature Based

pSigene (Probabilistic Signature generation) proposed by Gaspar Modelo-Howard et al [10], follows a four-step process for the automatic generation and update of intrusion signatures. In the first step attack samples are collected through crawling multiple public cyber security web portals. In the second step attack samples are used to generate feature set. Bi-clustering algorithm is used for grouping the features in the third step. A set of signatures are generated using regular expression for each cluster. They also rigorously benchmarked their solution with a large set of attack samples. Future work for this approach includes the implementation of the incremental update operation to reduce manual inspection.

Amith Pramod and et al. in [11] developed the system that can detect and prevent SQLI attacks for Union based, Tautology based and Illegal query based attacks by using a signature based approach to detect SQL injection attack. They used Kamal Dump Files, captured from the network analyzer (on the server side) with the help of mirrored ports. Deep Packet Inspection and pattern matching algorithm is used to detect attacks (on the client side).

Machine Learning

Detection of the stored procedure attacks is discussed in [12] by using machine learning approach. This approach useful to detect stored procedure attacks. SQL queries stored as tree format are collected from the log files. Queries trees are transferred to multidimensional array. By comparing these arrays, the normal and malicious queries are classified, with the help of the SVM classifier. Time for feature vector generation can be reduced in future; also this type of system can be used for Intrusion detection system to detect different types of attacks.

Anamika Joshi and Geetha V in [13] have proposed a classifier for detection of SQL Injection attacks, which uses combination of Naïve Bayes machine learning algorithm and Role Based Access Control mechanism for detection. They used the blank separation method and tokenizing method to extract terms with respect to each blank space in the query. Also the have proposed Role Based Access Control mechanism for detection. In this way they tried to overcome the limitations in Amnesia and SQLrand approaches for SQL Injection detection.

Tool Based

Abdilahi Liban and Shadi M.S. Hilles in [14] proposed a method to enhance MySQL injector Vulnerability. MySQL Injector was with limitations such as less number of attacks and attack patterns that the tool has covered. They covered four types of blind SQL injection attack true/false, true error, time-based and order by attacks. The existing blind SQL Injection tools have been enhanced by adding time-based attack using inference binary search algorithm. They used Perl programming language for the development of the tool, and it is only capable to penetrate the PHP based websites with MySQL Databases.

Bharti Nagpal et al in [15] have analyzed the use of Havij in penetration testing in IT industry and to compare various SQL Injection tools available in the market. It is an automated SQL Injection tool that helps the penetration testers to find and test SQL Injection vulnerabilities on a web

application, site pages. They have suggested that effort must be made to upgrade the tools, to a level where they could be used in industry.

Header Sanitization

Amirmohammad Sadeghian and et al. in [16] proposed model to patch SQL Injection vulnerability using Header Sanitization. Some of the existing tools for SQL injection defense needed too many changes in the existence web application and some other need high amount of time for debug and trace the bug to patch. The time factor is a big challenge for business which needs to be shut down for some time until the bug gets fixed. This system based on two methods. In the first method the array of HTTP header methods and fetch the contents of them. In next stage system cleans the content of variable from SQL injection and again it replaces the cleaned content back to the original header field. In this way it works by cleaning received vulnerable requests input and output variables inside HTTP header request methods.

Using New Algorithms

An algorithm proposed by Mahima Srivastava [17] for SQLIA prevention. In this approach author used the concept of ASCII values such that no of additional byte storage. The first phase is about what data should be stored in the database and how it should be stored. While the second phase illustrates how the data should be checked in the database and the third phase describes how data should be retrieved. Additional storage is not used for storing the values in the database. User can enter any data, any number of special characters without any restrictions.

Noor Ashitah Abu Othman et al in [18] proposed a prevention technique for SQLI attacks. Their system combines Query Tokenization and Adaptive Method which can be implemented on multiple platforms that are currently using asp.net programming language. SQL query inputs are first stored in database as ordered sequence of tokens. These tokens are then used to validate against the differences of incoming SQL query structure before sending it to database server for execution. This system has some disadvantages such as this approach is used to recognize and detects listed character only, can process only a single SQL query and it cannot handle code block. Also it is challenging to developers.

Anitha.V et al. [19] presented two algorithms namely Brute-Force String Matching Method and Longest-Common Subsequence algorithm to prevent the Union, Tautology, Error-based attacks and Piggy backing attacks. If there is malicious code in the query then, it blocks the query to get executed in the database by using these two algorithms.

Defensive Coding

Rules for developing the secured web application are given in [20]. These rules provide security constrains to developers. Best code Practices are: parameterised queries, escaping, data type validation, and white list filtering. It is a technique which comes in the category of defensive coding. It gives the ability of using dynamic statements without security problems. This tool is especially useful when developers need to use dynamic queries rather than parameterized queries for higher flexibility. Creating SQL statements in this way can lead to a many problems such as run time errors, SQL injection attacks, bad syntax, misspelled column and table names, data type mismatches, and code that is not easily maintainable. This consider very high user involvement, also developers should be trained. It can detect and prevent most types of SQLIAs but it fails to detect stored procedure attacks in SQL.

4. Comparisons and Discussions

In this paper gives the approaches implemented to protect web application, sites from SQL Injection Attacks. There are so many of approaches proposed by different authors. Most of the approaches deal with limited types of SQL Injection attacks. Table 1 for comparison and discussion gives a brief overview of the recent technologies for Protection of applications and web sites from SQL injection attacks.

Table 1: Comparison and Discussions.

Approach	Description	Future Scope.
Software Diversification Technique [6].	By using multitier diversified software SQL Injection attacks can be prevented.	Application code diversity. Using this approach on different machines.
Using Secure Delivery Networks [7].	Surveyed some common attacks on websites and techniques to mitigate those attacks.	To overcome technological challenges in complex WAF rules.
Protection of web applications via Unicode [8].	Verifying the trusted and non trusted data by keeping track of information at character level. Unicode system is used for tainting of data.	It can be used for dynamic information flow analysis and for defending applications against security threats.
Two tier Defense mechanism [9].	Defense tire, Identity Base Cryptography these two tier can be effectively used for detection.	To integrating this approach with other types of web application attacks.
Probabilistic Signature Generation [10]	A set of signatures are generated using regular expression for features cluster.	The implementation of the incremental update operation to reduce manual inspection.
A signature based approach to detect SQL injection attack[11].	Used Binary Search Algorithm. Also used Kamal Dump Files, deep packet inspection, With the help of mirrored ports.	It can be used for inspections of other types of attacks.
By using Internal query tree and SVM classifier [12]	It gets the internal query tree from the log files. Finds features, generates multidimensional arrays, Classifies queries by using SVM.	Time for generation of the multidimensional feature vector can be reduced, Intrusion Detection System can be developed.
Machine Learning - "Naïve Bayes" algorithm [13].	A classifier for detection of SQL Injection attacks, Which uses combination of Naïve Bayes machine learning algorithm and Role Based Access Control mechanism for detection.	It can be enhanced for detection of other types of SQL injection attacks also by extracting features appropriately.
MySQL injector Vulnerability[14]	Removed limitations such as less number of attacks and attack patterns that the tool has covered. Enhances SQL-injection vulnerability scanning tool.	To enhance the performance and the power of the tool, the process of extracting data such as tables.
Analyzing the use of Havij in penetration testing in IT industry[15].	It helps penetration testers to find and exploit SQL Injection vulnerabilities on a web site page.	To upgrade the tools, to a level where they could be used in industry.
Patch SQL Injection Vulnerability using Header Sanitization [16].	It detect and escape received vulnerable requests input and output by cleaning variables inside HTTP header request methods.	The same methode can be applied on POST, GET and REQUEST HTTP header requests.
Algorithm for Preventionn [17].	Using the concept of ASCII values data entry, authentication, retrival of data can be done.	Algorithm for other threats to application using ASCII values.
Combininning Query Tokenization and Adaptive Method [18].	Input-validation based on the Query Tokenization, Character List, Association List. Used a static analysis. Provide protection from attacks based on input vaidation.	It detects listed character only. This security module also can process only a single SQL query and it cannot handle code block.
Using String Matching and LCS Method [19].	Used Brute-Force String Matching Method and Longest-Common Subsequence algorithm to detect web attacks such as Union, Tautology, Banner-grabbing and Piggy backing attacks.	It can be applied to the other SQL Injection attacks. Can be applied to other technologies other than JSP, Microsoft SQL Server 2008R2.

It is required to make an "Intrusion detection system" which can detect and prevent attacks at application level and database level. Stored Procedure attacks in SQL database are very difficult to detect, and are detected by very few systems.

5. Conclusion

As daily usage of internet is increasing consistently, the vulnerability and the types of attacks are also increasing. Study and comparisons shows that SQL Injection is one of the dangerous attacks, there is lot of scope for improvement of the detection and prevention mechanisms. To protect web sites and web application from different types of attacks, programmers are needed to take care for defensive coding while developing the web application which maintains database at the backend. For detection and prevention of the SQL Injection attacks and other attacks these discussed approaches can be used as one or combinations.

References

[1] OWASP,http://www.owasp.org/index.php/MainPage.
[2] Diallo Abdoulaye Kindy and Al-Sakib Khan Pathan, "A survey on SQL injection: vulnerabilities, attacks, and prevention techniques", 2011 IEEE 15th International Symposium on Consumer Electronics.
[3] Lwin Khin Shar and Hee Beng Kuan Tan, "Defeating SQL Injection", 2013 Published by the IEEE Computer Society.
[4] Amirmohammad Sadeghian, Mazdak Zamani, Azizah Abd. Manaf, "A Taxonomy of SQL Injection Detection and Prevention Techniques" International Conference on Informatics and Creative Multimedia 2013 IEEE.
[5] William G.J. Halfond, Jeremy Viegas, and Alessandro Orso, "A Classification of SQL Injection Attacks and Countermeasures, IEEE 2006.
[6] Simon Allier, Olivier Barais, Benoit Baudry, Johann Bourcier, Erwan Daubert, Franck Fleurey, Martin Monperrus, Hui Song, Maxime Tricoire, "Multitier Diversification in Web-Based Software Applications", January/February 2015, IEEE Software.
[7] David Gillman, Yin Lin, Bruce Maggs, Ramesh K. Sitaraman, "Protecting Websites from Attack with Secure Delivery Networks", IEEE 2015.
[8] Boze Zekan, Mark Shtern, Vassilios Tzerpos, "Protecting Web Applications via Unicode Extension", SANER 2015, Montréal, Canada.

[9] Naresh Duhan, Bharti Saneja, "A Two Tier Defence against SQL Injection. International Conference on Signal Propagation and Computer Technology (ICSPCT) IEEE, 2014.
[10] Gaspar Modelo-Howard, Christopher N. Gutierrez, Fahad A. Arshad, Saurabh Bagchi, Yuan Qi, "pSigene: Web crawling to Generalize SQL Injection Signatures" International Conference on Dependable Systems and Networks, IEEE, 2014.
[11] Amith Pramod, Agneev Ghosh, Amal Mohan, Mohit Shrivastava and Dr. Rajashree Shettar, "SQLI Detection System for a safer Web Application", 2015 IEEE International Advance Computing Conference (IACC).
[12] Dong Hoon Lee, Mi-Yeon Kim, "Data-mining based SQL injection attack detection using internal query trees",expert systems with applications 41 (2014) 54165430, 2014.
[13] Anamika Joshi, Geetha V, "SQL Injection Detection using Machine Learning", 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies.
[14] Abdilahi Liban, Shadi M.S. Hilles, "Enhancing Mysql Injector Vulnerability Checker Tool (Mysql Injector) Using Inference Binary Search Algorithm for Blind Timing-Based Attack". 2014 IEEE 5th Control and System Graduate Research Colloquium, Aug. 11 - 12, UiTM, Shah Alam, Malaysia.
[15] Bharti Nagpal, Naresh Chauhan, Nanhay Singh, Angel Panesar, "Tool Based Implementation of SQL Injection for Penetration Testing", International Conference on Computing, Communication and Automation (ICCCA2015).
[16] Amirmohammad Sadeghian, Mazdak Zamani, Azizah Abd. Manaf, "SQL Injection Vulnerability General Patch Using Header Sanitization" IEEE 2014 International Conference on Computer, Communication, and Control Technology.
[17] Mahima Srivastava, "Algorithm to Prevent Back End Database against SQL Injection Attacks" IEEE 2014.
[18] Noor Ashitah, Abu Othman, Fakariah Hani, Mohd Ali Mashyum, Binti Mohd Noh, "Secured Web Application Using Combination of Query Tokenization and Adaptive Method in Preventing SQL Injection Attacks", IEEE 2014 International Conference on Computer, Communication, and Control Technology.
[19] Anitha.V , Supha Lakshmi.A , Revathi.M, Selvi.K , "Detecting Various SQL Injection Vulnerabilities using String Matching and LCS Method", 2014 Sixth International Conference on Advanced Computing (ICoAC).
[20] Russell A. McClure and Ingolf H. Krger, "SQL DOM: Compile Time Checking of Dynamic SQL Statements". 2005 ACM.

Author Profile

Aniruddh R. Ladole received the B.E. degree in Computer Science and Engineering from Anuradha Engineering College, in 2013. He is pursuing Master"s from D. Y. Patil College of engineering, akurdi, pune. His area of interest is Cyber Security, Database Security, Intrusion detection system.

Mrs D. A. Phalke completed her Master"s in Computer Engineering from D Y Patil College of Engineering and pursuing PhD in Computer Science and Information Technology at Department of Technology, Savitribai Phule Pune University. She is having 14 years of experience of PG and UG teaching and 13 publications in International Journals and 16 in International Conference. Her area of interest is Data Mining, Database Security and Multimedia Data.