Explorative Study of SQL Injection Attacks and Mechanisms to Secure Web Application Database- A Review

Chandershekhar Sharma
Research Scholar, Computer Science and Engineering
Rajasthan Technical University
Kota, India

Dr. S. C. Jain
Professor, Computer Science and Engineering
Rajasthan Technical University
Kota, India

Dr. Arvind K Sharma
Department of CSI
Kota University
Kota, India

***Abstract*—The increasing innovations in web development technologies direct the augmentation of user friendly web applications. With activities like - online banking, shopping, booking, trading etc. these applications have become an integral part of everyone's daily routine. The profit driven online business industry has also acknowledged this growth because a thriving application provides the global platform to an organization. Database of web application is the most valuable asset which stores sensitive information of an individual and of an organization. SQLIA is the topmost threat as it targets the database on web application. It allows the attacker to gain control over the application ensuing financial fraud, leak of confidential data and even deleting the database. The exhaustive survey of SQL injection attacks presented in this paper is based on empirical analysis. This comprises the deployment of injection mechanism for each attack with respective types on various websites, dummy databases and web applications. The paramount security mechanism for web application database is also discussed to mitigate SQL injection attacks.**

Keywords—Injection Attacks; SQL vulnerabilities; Web Application Attacks

I. INTRODUCTION

Rapid advancement in web technologies has expedited the rate of adoption of database driven web application. The backend database servers of these web applications accumulate some general data along with critical & sensitive information about organizations and clients [40].As the database is accessible from anywhere over internet makes it prone to attacks. The most hazardous attacks against database driven web applications are –SQL injection attacks [48]. These attacks are very serious threat to any web application that receives inputs from user and incorporate it in to SQL queries to an underlying database[10][38].Although web application keeps the user's data secure for making any online exchange of information but presence of vulnerabilities makes this attack feasible. SQLIA are mostly caused by the insufficient validation of user input.

An attacker can submit a query (utilizing SQL command) directly to the database which can extract categorical information depending upon the severity of vulnerability. Database is the main asset of any web application to which attackers are keenly fascinated. SQL injection is very lucrative for attackers as there is a successful black market that deal all scarcely digitally purloined data like credit card information, bank accounts detail and social security numbers etc.[21].

With a little knowledge of SQL commands and ingenious conjecture work to crucial table name SQL injection attacks can be launched. These commands alter the desired output of queries to break in to database. Injection attacks were ranked 1st attack in 2013 by OWASP (Open Web Application Security projects) in TOP ten attacks and found that 80% of web applications are Vulnerably susceptible to SQL injection attacks [14] [33][47]. Before moving further, one must go through some fundamental definition for better understanding of the SQL injection attacks on web application and its underlying database [49].

Vulnerability: Vulnerabilities are the impuissance, loopholes, bugs or fault/imperfection in the subsisting system.

Attack: An attack is an illicit access i.e. a method to exploit vulnerabilities.

Threat: a series of events that utilizes the system in an unauthorized way compromising the principles of information security i.e. confidentiality, integrity and availability of the system.

Risk: Impact of the threat.

Albeit many researchers and practitioners have done the survey on SQL injection attacks against database but a detailed survey is done to elaborate the other aspects of attacks against database. In this paper an endeavour is done to provide the taxonomy of SQL Injection Attacks against database of a web application. This repository is the relegation scheme of attacks which includes- Research papers, white papers, technical reports and web sites. It can become a vital auxiliary in designing the security for web application and its underlying database.

The rest of paper is organized as follows: Section 2 describes the architecture of web application. Section 3 covers SQL injection attacks preview. Section 4 explains SQL injection classification and section 5 have explorative study of SQLIA. Findings of the study are described in Section 6.Section 7 briefs the security mechanisms. Finally section 8 concludes the paper.

II. Architecture of Web Application

For the better understanding of SQLIA one must have the cognizance of web application architecture. Web applications are a set of web pages and programs which reside on a web server [17]. The inputs provided by the user are sent to the server in the form of parameter string. These inputs are used to engender SQL query to retrieve information from the database. An authorized user can access it over the cyber world or over a public network and store the data in the database. A web application utilizes a web browser as an interface to extract the data from database server to accommodate the queries placed by the users [11][48]. Every web application is predicated on 3 –tier architecture consisting of three layers [48] .Each layer can run potentially on a different machine and each layer should be independent of other layers. The three layers are

Presentation Layer: Presentation layer contains presentation logic. It is the top most level of application and handles the interactions with users. Its main function is to receive input from the user and provide the result in a convenient way that user can facilely understand.

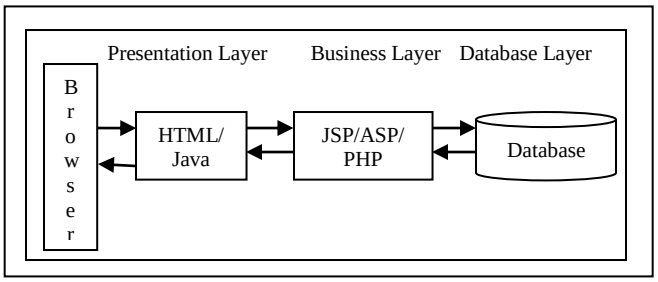

Fig. 1. Architecture of web application

Business Layer: This layer is present in between presentation layer and database layer. It is a logic layer which consists of a set of rules for processing the information between two layers. It contains application process commands which retrieves the data from database and sends to presentation layer for viewing the data. This tier can be programmed in any server scripting language like JSP, PHP, and ASP etc.

Database Layer: This is a physical storage layer for data persistence. It manages all access to database and file system. Information is stored and retrieved from database .It is then passed back to the presentation layer for processing and eventually back to the user. The main function of this layer is to provide access to authorized user and restrict the maleficent user.

Working principle of architecture: The presentation layer receives the request from web browser, processes it and then passes the dynamic part to the business layer which processes server scripting languages. All requests for database access are passed to the database server. The result is then dispatched to web browser as web pages. This architecture is easy to maintain and all the components are reusable. For the faster and smooth functioning, all the layers are governed and managed by different groups of experts. Web designer looks after the presentation layer. Software engineer does the logic and database administrator manages the database servers.

III. SQL Injection Attack- Preview

The focus on implementing the functionality rather than security from internal and external environment causes the susceptibilities in web applications. These are described as most solemn threat for web application as it may allow attacker to gain access to the web application and its underlying database. The potential of attacker perforates the system to extract sensitive information. Exploitation of loopholes in the design breaches the fundamental principles of information security i.e. confidentiality, Integrity, authenticity and availability of information [39][50].Information stolen is loss of confidentiality. Loss of integrity takes place when data is modified in an unexpected manner. The Denial of service takes place when information is expunged for genuine user. Loss of authenticity means when information is accessed by unauthorized user. These attacks are application level attacks which are not obviated by firewalls [35][36][46].It is hard to detect SQL injection prior to its impact. In most of the cases the unauthorized activity is performed through a valid user credentials for accessing the critical section of database of web application. The database servers are convinced that injected code is syntactically as valid as a SQL code. The inputs provided by the user form the dynamic SQL query to access the backend database. If these inputs are not opportunely sanitized, they can cause the web application to generate unintended outputs. The basic underlying fact is that SQL injection attacks are very easy to execute without any professional training.

Consider the following SQL statement.

SELECT Emp_info from Employee where E_name=’abcd’ AND E_id = ‘12345’;

The above query will provide the information about a particular desired employee for supplied input values but a SQL expression with injection will deport differently because the logic of the query is transmuted by the attackers, as

SELECT Emp_info from Employee where E_name=’abcd’ AND E_id = ‘12345’ OR ‘1’=’1’;

Because of an injection statement (OR ‘1 =1’) the list of all the employees from the table in lieu of a desired output exposes the whole database. Such types of susceptibilities form the attacks [11]. Example expounded above is a very fundamental injection attack. The professional attacker uses very logical and resourceful keywords to extract the data from database servers.

IV. Organization of SQL Injection Attacks

The objective of SQL Injection Attack (SQLIA) is to penetrate the database system into running inimical code that can reveal confidential information. This is done by injecting the SQL queries and expressions as an input string to gain an unauthorized access. SQL injection is a threat that leads to a high level of compromise - conventionally the ability to run any database query. It is web-predicated application level attacks that connects to backend database and bypass the firewall. The advantage of insecure code and deplorable input validation is clinched by the attacker to execute unauthorized SQL commands. On the substratum of attacks against the

database management system, SQL injection attack can be classified as [7][10][16][43][45][46].

SQL Injection attacks against web application databases can be divided in four sections as follows:

1) *Code Injection*
2) *SQL Manipulation*
3) *Function Call Injection*
4) *Buffer Overflows*

Code injection is when an attacker inserts new database commands or SQL statements into user's statement. The manipulation involves modifying the SQL statement through set operations or altering the WHERE clause to return a different result. When attacker injects a customized function or already existing function into a SQL statement, that type of injection is known as function call injection. These are used to manipulate data in the database. Buffer overflows is a subset of function call injection. In several commercial applications vulnerabilities exist in database functions that may result in a buffer overflow.

a) Code Injection

In code injection, attacker attempt to add additional SQL statement or commands to the existing SQL statements.

Original query

SELECT Emp_salary FROM Employee WHERE E_name='kushagra' and E_id='abc123';

Output

A salary statement of desired employee is extracted from the database.

Injection query

SELECT Emp_salary FROM Employee WHERE E_name='kushagra' and E_id='abc123'; DELETE FROM Employee WHERE username = 'kushagra';

The above query uses the stored procedure command which is inserted between original query and injected query so that the both queries executes in a single run as a single query.

Output

Information about desired employee is deleted from the database due to additional command.

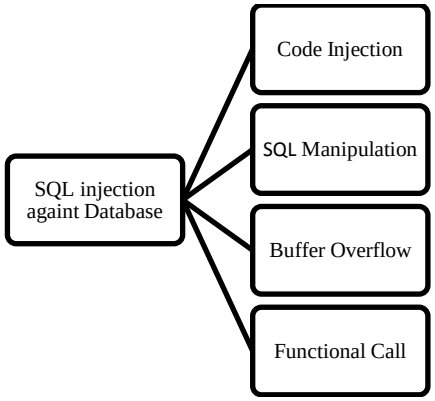

Fig. 2. Classification of SQLIA against database

b) SQL Manipulation

A very familiar kind of SQL Injection attack is SQL manipulation. The most significant examples of this type is by adding elements to the WHERE clause with set operators like UNION, INTERSECT etc. Or comparators like OR, <,> and many more. The simplest example is login authentication that a web application may check.

Original query

SELECT * FROM Employee WHERE username='kushagra' and Password='abc123';

Output

A desired employee details are returned by the database.

Manipulation query

SELECT * FROM Employee WHERE username='kushagra' OR '2' > '1'and Password='-----';

Based on operator precedence, the clause WHERE is true for every entry and (------) regarded as comment consequently ignored by server granting access to attacker.

Output

Information about all users is received without authorization.

It can also extract information about all the users using union query.

SELECT * FROM Employee WHERE username like '%kushagra'; UNION SELECT username FROM users WHERE username like '%';

c) Function call injection

Function call injection is the insertion of functions which executes with a SQL statement. Functions which are marked as "PRAGMA TRANSACTION" are executed as part of a SQL SELECT statement. Using INSERT, UPDATE, or DELETE in SQL statement attacker is able to modify data in the database. By using custom functions an attacker can send information to a remote computer or execute other attacks on the database server.

Example: A custom application has the function NEWUSER in the custom package MYAPP. The developer marked the function as "PRAGMA TRANSACTION", since it is marked "PRAGMA TRANSACTION", it can write to the database even in a SELECT statement

SELECT TRANSLATE ('' || *myapp.newuser ('admin', 'newpass')* || '', '12345ABCDE', 'abcde56789') FROM dual;

This type of injection can even exploit the simplest SQL statement at runtime.

d) Buffer Overflow

When the process of writing data to buffer overruns the boundary of buffer and overwrites adjacent memory then an abnormality is recorded known as Buffer overflow. These injection attacks are triggered either by variations in inputs or by amending the method of program operation. The standard database functions which can be exploited through a SQL

injection are tz_offset, to_timestamp_tz, and bfilename. A buffer overflow attack is executed using the function injection methods. The very effective buffer overflow attack is denial of service [44][52].

Example: The web process gets hanged until connection to the client is terminated.

V. EXPLORATIVE STUDY OF SQLIA

In this section the above briefed attacks are discussed in detailed with their code injecting mechanism, types and subtypes with appropriate SQL statements and queries.

A. Code Injection

Injection Mechanism: The attacker attempts to manipulate the SQL statement by injecting additional code, operator sub query in to original query.

The code injection attacks can be further divided in to four types.

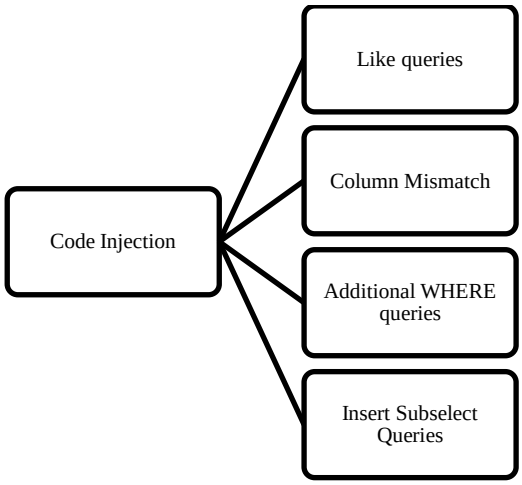

Fig. 3. Types of code Injection attacks

a) Like queries: Like query is used to compare a value to similar values using wildcard operators.Bascically two wildcards are used

i) % represents zero,one or multiple characters

ii) underscore (_) represents a single number or character.

The attacker manipulate the SQL statement using these wildcards. A Denial of Service attack can be launched with a few changes in a LIKE query by overloading the database [7][44][51].

Example:

SELECT Employement_No FROM Employee WHERE E_Name LIKE 'A%';

It provides the list of all employees starting with name A or having name an alphabet A

b) Column Mismatch: This particular attack occurs when there are errors like mismatch operand type or "Queries containing a UNION operator must have same number of expressions". To gather the information of this type of injection, consider a random SQL query.

Example:

SELECT product_name FROM all_products WHERE product_name like '&Chairs&'

It provides the name of product according to the query input but attacker changes the logic of the query it becomes

SELECT product_name FROM all_products WHERE product_name like '' UNION SELECT ALL FROM Objects WHERE '' = '';

Above query would give errors that indicate that there is mismatch in the number of columns and their data type in the union of Objects table and the columns that are specified using ALL. The error is caused by the injected string. Another error is because the number of columns is not matching. This information is enough to penetrate the database of any web application [40][43].

c) Additional where queries: The attacker can also extract the information from the message displayed by the database while using an additional WHERE clause with the input string.

Example:

SELECT FName, LName from Employees WHERE City= 'Delhi' AND Country='India';

This query will provide first name and last of all the employees working in a particular city & country. The modified query with injected additional WHERE clause will be

SELECT FName, LName from Employees WHERE City= 'Nosuchcity' UNION ALL SELECT SomeField from SomeTable WHERE 1 = 1 AND Country = 'USA';

Due to this insertion clause the detailed error message displayed by the database reveals the name of table and column name in error message- like Invalid column Name 'Country' because 'Table1' does not have a column called 'Country'. It exposes the database with table name

d) Insert subselect Queries:

The insertion of a sub query into a query can also help the attacker to access all the records of the database

Example:

SELECT E_Name FROM Employee WHERE Employee Employement_No = (SELECT work_in.Employement_no FROM work_in WHERE D_No= 23)

The result of above query will provide the result to find the name of all employees working in department no 23.If the query is executed then the result is displayed otherwise there will be an error message like "subselect returned too many rows". Attacker can go through all the record using NOT operator. This attack is possible where users are allowed to edit user information [7][52].

B. SQL Manipulation

Injection Mechanism: The attacker attempts to manipulate the SQL statement by injecting the code in one or more conditional statements.

These SQL manipulation attacks have four types as shown in the figure given below.

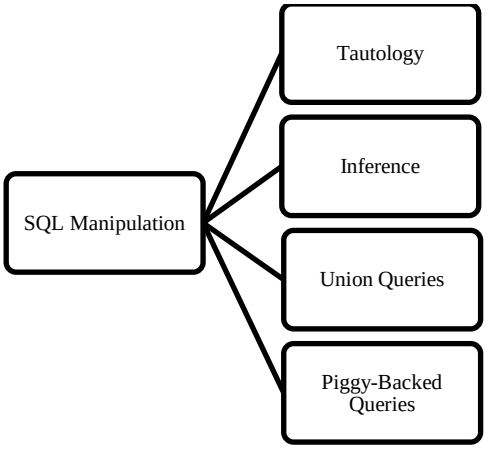

Fig. 4. Types of SQL manipulation

a) Tautology: In this type of SQLIA, an attacker exploits an injectable field that is used in a query. The query always returns result upon evaluation of a WHERE conditional parameter. The aim of this attack is to inject malicious codes into one or more conditional statements which are always evaluated to be true.All the rows in the database table targeted by the injected query returns the conditional WHERE into a tautology.

Example: It allows an attacker to log on to application without supplying a valid user name. The attacker submits

'Abcd' OR '1'='1'/'A'='A' in login field and '-----'in password or pin field.

The resulting query is:

SELECT Emp_id FROM Department WHERE

Login='Abcd' OR 'A'='A' AND pass='-------';

As the condition (OR A=A) is always true and ---- is used for comments, it converts the entire WHERE clause into a tautology [38][40][45].

b) Inference: when the attacker tries extracting information from not enough secured backend database via error messages then inference injection is executed. A different method must be used by the attacker to get the response from the database since database error messages are not available without executing a query or statement .The error messages displayed by database may become useful tools for attacker to plan an attack. In this situation, the attacker injects commands and then observes the change in function/response of application. By carefully observing the behaviour of application vulnerable parameters can be extrapolated by attacker with added information about the database.

Example: SELECT Employee from Bankers where E_number=' $%^&*!@#'and E_id ='AZ+=79%':

The input provided in query is incorrect & results in an error message. The displayed error message should be like

"Microsoft OLEDB provider for SQL Server (0x80040E07) error converting nvarchar value 'E_number' to column of data type int".

The Information about version and schema of backend database and is reveled to attacker to plan further Inference attacks [10][40][43].

c) Basic union queries: This type of attack is also called statement injection attack. The attacker tricks the database server to return data that which is not intended by the authentic user. The vulnerable parameters are exploited with the help of keyword UNION, which is used to join original query and an injected query. The attacker controls the injected query completely, to retrieve information from database. The output of this attack causes the database return values which is union of two queries

Example: SELECT Basic_info from Employee where user E_name='xyz' and E_id='------'; UNION SELECT

Salary _info from Employee where Emp_id= '1234';

The first part of the query gives null values but second part of the query returns the information of employee having id 1234 [10] [16][40][51].

d) piggy- backed queries: In this category the attacker's aim is not to modify the query. The addition of distinct queries with the valid query is desirable. When database receives multiple queries, shows extremely harmful results like deletion or removal of information with a harming intention.

Example: SELECT Basic_info from Employee where user E_name='xyz' and E_id='1234'; DROP table Employee;

Here two queries are separated by delimiter (;). So both the queries get executed. After execution of first query the database proceeds for the injected second query. When second query executes, it will drop table 'Employee', from database which results in potential damage to important information. In the similar manner there are various other types of queries like inserting new employees to table etc. [10][40][45].

C. Function Call Injections

Injection Mechanism: The attacker attempts to manipulate the SQL statement by inserting database functions or custom functions into a vulnerable SQL statement [43][53]. This type of injection attack has two types role function and system stored procedure

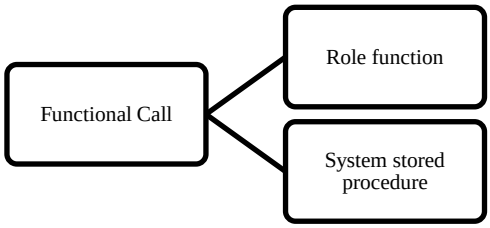

Fig. 5. Types of function call injection

the knowledge of this type of injection, Consider the Webportal of a company which provides press release on regular intervals.

Example:

www.company.com/PressRelase.php?pRealeaseID

The corresponding SQL statement used by the application would look as under, if pRelease is 7

Select title, description, releasedate, body from pressRelease WHERE pRelease ID=7

All the information requested corresponding to the 7th press release is returned by the database server. This information is in a HTML page (understood by the browser) and provided to the user. Certain changes in parameters in the address of a HTML page by attacker can change the role of a query. Like

Select title, description, releasedate, body from pRelease WHERE pRelease ID=7 AND 2 > 1; or

Select title, description, releasedate, body from pressRelease WHERE pRelease ID=6;

If the application still returns some document i.e. it is susceptible to SQL injection attack and attacker can plan the attack accordingly [7][47].

a) Stored system procedures: In this type the in-built stored procedures are attacked by attacker using malicious sql injection codes. Every database uses the stored procedures. The knowledge of running backend server allows the attacker to penetrate the system using the stored procedures. This attack is like Piggy Backed queries attack in which stored procedure command is inserted inbetween original query. The both queries execute as one.

Example: SELECT Emp_ Salary from Employee where username='abcd'; SHUTDOWN; and password='12345';

In above query SHUTDOWN is a stored procedure which causes shutdown the database. The admittance to system store procedures depends on the access privileges to user by the application [10][40][51].

D. Buffer Overflow

Injection Mechanism: The attacker attempts to manipulate the SQL statement by using database functions which are susceptible to buffer overflows. In several databases some database functions are vulnerable to buffer overflows that can be exploited through a SQL injection attack. A very efficacious denial of service attack hangs the process until the connection is terminated with the above said attack [44][52][53].

During the preparation of this repository some other methods are encountered. These methods do not fit under any attack schema but can be used to inject malevolent code in to user's code for executing SQL injection attacks. The methods are explained as under [17][35][36][46][48][52].

a) Sophisticated Matches: One of the prevalent signatures utilized by such mechanisms is some remotely variant on the famous OR' 1'='1'attacks. Sophisticated matches technique uses alternative expression of OR' 1'='1'. For example: OR 'Unusual' = 'Unusual', OR 'Simple' ='Sim'+'ple', OR 2 > 1 and OR 'Simple' BETWEEN 'R' AND 'T' all have the same effect as OR' 1'='1'.

b) Hex Encoding: Hex encoding technique uses hexadecimal encoding to represent a string. For example, the string 'SELECT' can be represented by the hexadecimal number 0x73656c656374, which most likely will not be detected by a signature protection mechanism.

c) Char Encoding: Char encoding technique uses build-in CHAR function to represent a character. For example, the string 'SELECT' can be represented by the CHAR function as char (73)+char (65) +"LECT", which make it very arduous for detection system to build a signature that match it.

d) In-line Comment: In-line comment technique obscures input strings by inserting in-line comments between SQL keywords. For instance, /**/UNION/**/SELECT/**/name designation can elude detection from signatures that expects white space between SQL keywords.

e) Dropping White Space: Dropping white space technique obscures input strings by dropping white space between SQL5 keyword and string or number literals. For example, OR 'Simple'='Simple' works precisely the same way as OR 'Simple' = ' Simple', but has no spaces in it, make it capable of eschewing any spaces predicated signature.

f) Break Words in the Middle: With MySQL, the in-line comments would not work as supersession for a space. The in-line comments can be utilized in MySQL to break words in the middle, for Instance: UN/**/ION/**/ SE/**/LECT/**/ is evaluated as UNION SELECT.

VI. FINDINGS OF STUDY

The study of the SQL injection attacks (SQLIA) against database centric web application concludes that these attacks possess the great threat. Unfiltered user inputs invite these attacks. Bypass the authentication process sanctions the attacker to postulate all privileges associated with genuine user. The Retrieval of personal and sensitive information is highly desirable by the attackers. Mostly SQL injection attacks are done to steal the sensitive information. By knowing the version and type of the database utilized by the user makes it facile for the attacker to craft a query. Sometimes the goal of attacker is to integrate the information in a database to mark one's identity; if it is done for enjoyment then no harm is caused but an attacker with destructive intention can delete the whole database. By remote command execution attacker can even

shutdown the database. Based on above discussion some findings are listed as under

A. Most common reasons are behind SQLIA

a) Mismatch Data type

b) Accounts with more access

c) Insufficient input validation

d) Detailed Error messages

e) No sanitization of data sent to the server through URL.

B. Privileges gained on successful SQLIA

a) Access to Database schema.

b) Disclosure threats of the asset.

c) Gain access to host and internal network.

d) Exploitation of susceptibilities of the web application.

e) Privilege escalation.

f) Impose deception and usurpation threats.

C. Methods for eschewing SQLIA

a) Avoid building dynamic SQL expression from user input.

b) Length of input string must be constrained.

c) Avoid using query delimiter, SQL keyword, character data string delimiter and single line comment in user input.

d) Use different Database account for different calibers of privileges.

e) Error messages must be customized to hide the details of injectable parameters.

f) Use parameterized queries for Database access.

g) Use stored procedures to avoid direct access of Database.

h) Evade building SQL statements from cookie and HTTP variables.

VII. Security Mechanisms

When good programming habits and eschewing methods are not sufficient for the avoidance of SQL injection attacks then some manual and automated security mechanisms are applied for the protection of web application database. These are very inimical attacks which target the most valuable assets of web application. The consequences of SQL injection attacks range from modification in data to denial of data. Researchers have proposed several techniques to contravene SQL-injection attacks which include - Code review, Defensive programming, Software hardening techniques, and Hardware extensions feature in modern processors, Attack detection and containment mechanisms and many more. These approaches are not sufficient to counter the problem of SQL injection attacks. Some security mechanisms are suggested to faceoff SQL injection attacks.

A. Detection and Prevention tool

When the techniques like defensive coding & operating system hardening [13] are not enough to stop SQLIA, some tools are required for detection and prevention of web application and its underlying database. Prevention designates to evade unauthorized user (attacker) from accessing any component of system or data. Prevention tools [26][30][34] are runtime analysis for checking susceptibilities by placing a validation checker between web server and database server. It additionally averts the attacks that capitalize on type mismatch, sanitization of inputs and input sources withal [20][22]. Detection determines whether someone has attempted to break into your system, if yes, then up to what extent of damage may have been done. Detection tools are implemented in two approaches [18][31][37] – Static technique and Dynamic technique. Static technique [33] is applied directly without running the code, it includes approaches [6][27][28] like – Pattern Matching, Lexical Analysis and Parsing (includes type qualifier, dataflow analysis, taint analysis and model checking).Whereas dynamic techniques [33] include approaches like- Fault injection, Fuzzy testing, Dynamic taint & Sanitization of inputs [5]. Most of the approaches [31] are not implemented yet as a tool. So the scope of developing prevention tool, detection tool or the combination of both prevention and detection tool is on the cards for the researchers.

B. Instruction Set Randomization

A technique to counter SQL- injection attack, is instruction set randomization (ISR) [1][4][12][46]. The fundamental idea behind this approach is that attacker doesn't know the language spoken by the runtime environment on which an application runs, so a SQL-injection attack will ultimately fail because the foreign code, however injected, is written in a different language. An ISR to SQL injection is straightforward approach. It randomizes both the underlying runtime environment the SQL parser and the SQL program (the template that the Web application uses). A simple approach for randomizing the SQL grammar consists of appending a random numeric tag (the randomization key) to each statement and operator in SQL .This can be an efficacious way of ceasing injection attacks, but it typically requires extensive modifications to the runtime environment.

C. Intrusion Detection System and Proxy Server

Most of the SQL injection attacks are application level attacks. A model can be built to filter the SQL queries which work as security layer between database server and web server to filter the queries at run time[9][12][19][24]. Similarly a Proxy filter can be developed to intercept the HTTP request and enforce input validation so that malevolent SQL expression would be averted to send to database server [15][23][29].Many models have been proposed by researches based on different approaches like machine learning, intended structure of SQL verbalizations, data flow analysis, identification of critical points and many more but are not sufficient to avert SQL injection attacks [29][31]. So an incipient hybrid model can be proposed which may be the combination of several approaches to counter injection attacks.

D. Threat Model

Threat modeling [2][3][8][13] is a procedure for optimizing Security by identifying objective, susceptibilities and then defining countermeasure to avert the effects of threat to the

system [19].The threat modeling process customarily involves identifying information sources to be bulwarked, ingression points or access points to the system's assets, analyzing the threats, evaluating the associated risks and developing mitigation strategies[25].There are three different approaches of threat modeling – Asset centric (Find risks associated and rank the risks), Attacker centric (Find level of harm, Evaluating damage potential and risk rating) and Software centric (Decomposes the application to identify the threats and then mitigate the threats).Many more threat models [41][42] or a hybrid model (amalgamation of two or more models) can be proposed to filter the susceptibilities and malevolent SQL verbal expressions from different input sources and mitigate the attacks.

VIII. CONCLUSION

In this paper the detailed analysis is presented on various types of SQL injection attacks with related vulnerabilities and injection mechanisms. SQL Injection is a common technique that attackers employ on web based data centric applications. These attacks modify the SQL queries in a manner to alter the behaviour of application. This paper also provides the taxonomy of mechanisms for avoidance, prevention and detection from these attacks. In future work, a Threat model will be proposed as a security mechanism for securing database of web applications.

REFERENCES

[1] E.G. Barrantes,"Randomized Instruction Set Emulation to Disrupt Binary Code Injection Attacks," Proc. 10th ACM International Conference on Computer and Communication Security, pp. 281–289, 2003.

[2] F.Swiderski and W.Snyder,"Threat modeling" (Microsoft press, 2004)

[3] Threat risk modeling.,"A Guide to Building Secure Web Applications and Web Services," (2.0 Black Hat edition, p. 38-51,July 2005)

[4] A. Sovarel, D. Evans and N. Paul,"Where's the FEEB? :The Effectiveness of Instruction Set Randomization," Proc. Usenix Security Symposium, Usenix Association,pp. 145–160, 2005.

[5] W.G.Halfond and A.Orso.Amnesia,"Analysis and monitoring for neutralizing sql injection attacks," Proc.20th ACM Int. Conf. on Automated Software Engineering., Long Beach, California,USA, p.174,2005.

[6] A McClure and Ingolf H.Kruger,"SQL DOM: Compile TimeChecking of Dynamic SQL Statements," Proc. International ACM Conference of Software Engineering, pp. 88-96, May 2005.

[7] Sagar Joshi,"SQL injection attack and defense: Web Application and SQL injection," 2005.

[8] P.Torr,"Demystifying the threat modeling process"IEEE transaction on Security & Privacy, 3(5), pp. 66- 70, 2005.

[9] F. Valeur, D. Mutz and G. Vigna,"A Learning-Based Approach to the Detection of SQL Attacks" Detection of Intrus Malware and Vulnerability Assessment Proceedings, Volume: 3548, pp.123-140,2005.

[10] W.G.Halfond,J. Viegas and A. Orso,"A Classification of SQL-Injection Attacks and Countermeasures," Proc. IEEE Int. Symposium on Secure Software Engineering, Washington,DC,USA,March 2006.

[11] Z.Su and G. Wassermann,"The Essence of Command Injection Attacks in Web Applications," Proc. 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages,Charleston,South Carolina, USA, pp. 372-382,2006.

[12] Y.Weiss and E.G. Barrantes,"Known/Chosen Key Attacks against Software Instruction Set Randomization," Proc. Annual Computer Security Applications Conf. (ACSAC),pp. 349–360,2006.

[13] E. A.Oladimeji and S.Supakkul, L. Chung,"Security threat modeling and analysis: a goal-oriented approach,"Proc. of the 10th IASTED International Conference on Software Engineering and Applications (SEA), pp.13-15,2006.

[14] Jonse Fonseca,"Testing and comparing web vulnerability scanning tool for SQL and XSS attacks," Proc. 13th IEEE symposium on Pacific Rim Dependable Computing,pp 365-372,2007.

[15] Xiang Fu and Xin Lu,"A Static Analysis Framework For Detecting SQL Injection Vulnerabilities," Proc. 31st IEEE Annual International Computer Software and Application Conference, pp. 87-96,24-27 July 2007.

[16] San-Tsai Sun, Ting Han Wei, Stephen Liu, and Sheung Lau,"Classification of SQL Injection Attacks,"Electrical and Computer Engineering, University of British Columbia, November 2007.

[17] D.Stttard and M.Pinto,"The Web Application Hacker's Handbook:Discovering andExploiting Security Flaws,"(Wiley publication, 2007)

[18] M.Cova,D.Balzarotti,V.Felmetsger and G.Vigna, "Swaddler: An Approach for the Anomaly-based Detection of State Violations in Web Applications", the Recent Advances in Intrusion Detection (RAID), Gold Coast, Australia, pp.63 – 86, 2007.

[19] K.Kemalis and T.Tzouramanis,"SQL-IDS:A Specification based Approach for SQL Injection Detection," Symposium on Applied Computing. ACM, pp.2153-2158, 2008.

[20] Monticelli, F,"SQL Prevent," University of British Columbia (UBC) Vancouver, Canada.2008.

[21] Symantec,"Symantec Report on Underground Economy," pp.9-12,Symantec, 2008.

[22] Jin-Cherng Lin, Jan-Min Chen and Cheng-Hsiung Liu,"An Automatic Mechanism For Sanitizing Malicious Injection," Proc. 9th IEEE International Conference For Young Computer Scientists,pp.1470-1475,18-21Nov 2008.

[23] Anyi liu and yi yuan,"SQLProb: A Proxy based Architecture towards preventing SQL injection attacks,"ACM, pp.2054-2061, March 2009.

[24] Anglos D. Keromytis,"Randomized Instruction sets and run time Environment,"IEEE Transaction on Security & Privacy, pp.18-25, Jan/Feb 2009.

[25] F.Swiderski and W.Snyder,"Threat modeling" O'Reilly Media, Inc., 2009.

[26] P.Bisht, P.Madhusudan and V.N. Venkatakrishan," 'CANDID: Dynamic Candidate Evaluations for Automatic Prevention of SQL Injection Attacks," ACM Transaction on Information System Security, pp.1-39,2010.

[27] Ivano Alessandro Elia, Jose Fonseca and Macro Vieira,"Comparing SQL Injection Detection Tools using Attacks Injection: An Experimental Study,"Proc. 21st International Symposium on Software Reliability Engineering,pp.289-298,1-4 Nov 2010.

[28] Jeom-Goo Kim,"Injection Attack Detection using the Removal of SQL Query Attribute Values,"Proc. International Conference in Information Science and Application(ICISA),pp.1-7,26-29 April 2011.

[29] Chai Wenguuang, Tan Chunhui and Duan Yuting,"Research of Intelligent Intrusion Detection System Based On Web Data Mining Technology," Proc. IEEE 4th International Conf. on Business Intelligence and Financial Engineering,pp.14-17,17-18 Oct 2011.

[30] Indrani Balasundram and E .Ramaraj,"An Authentication scheme for Preventing SQL Injection Attack Using Hybrid Encryption," (PSQLI-HBE),53(3), pp.359-36, 2011.

[31] Atefeh Tajpour, Suhaimi Ibrahim and Mohammad Sharifi,"Web Application Security by SQL Injection Detection Tools," International Journal of Computer Science, 9(2), pp.332-338,2012.

[32] Avanish Kumar Singh and Sangita Roy,"A network based Vulnerability scanner for Detecting SQLI Attacks in Web Applications," Proc. 1st Int. Conf. on Recent Advances in Information Technology (RAIT-2012), Dhanbad, India, pp.585-590,15-16 March 2012

[33] Sruthy Mamadhan, Manesh T and Varghese Paul," SQLStor: Blockage of Stored Procedure SQL Injection Attack Using Dynamic Query Structure Validation," Proc. IEEE 12th International Conf.on Intelligent Systems Design and Applications(ISDA),pp.240-245,27-29 Nov 2012.

[34] Debabrata Kar and Suvasini Panigrahi,"Prevention of SQL Injection Attack Using Query Transformation and Hashing," Proc. IEEE 3rd International Conf. on Advance Computing, pp.1317-1323,22-23 Feb 2013.

[35] Chad Dougherty,"Practical Identification of SQL Injection Vulnerabilities," Produced for US-CERT© 2012

[36] Cenzic vulnerability report 2013.

[37] Jaskanwal Minhas and Raman Kumar,"Blocking of SQL Injection Attacks by Comparing Static and Dynamic Queries," International Journal Computer Network and Information Security,vol.2,pp.1-9,2013.

[38] Chandershekhar Sharma and S.C. Jain,"SQL Injection Attacks on Web Application," International Journal of Advanced Research in Computer Science and Software Engineering,4(3),pp.1268-1272,2014.

[39] F. S. Labs,"Threat report" Last Accessed: 27-3-2014.

[40] Chandershekhar Sharma and S.C. Jain,"Analysis and Classification of SQL Injection Vulnerabilities and Attacks on Web Applications," Proc. IEEE Int. Conf. on Advances in Engineering & Technology research (ICAETR-2014),Dr. virendra Swarup group of institutions, Unnao, India,pp.1-6, August 2014.

[41] Adam Shostack,"Threat Modeling: designing for security" Wiley publication,2014.

[42] Satapathy Soumya Ranjan,"Threat Modeling in Web Applications," Thesis-NIT, Rourkela, 2014.

[43] C.anely,"Advanced SQL injection in SQL server applications"White paper.

[44] Kevin J.Houle,"Trends in Denial of Service Attack Technology," Whitepaper.

[45] S.Mcdonald,"SQL Injection: Modes of Attack,Defense, and Why It Matters,"-White paper.

[46] www.salientsecurity.com

[47] www.securitydocs.com/librarys

[48] www.owasp.org/index.php/Top_10_2013.

[49] www.bcs.org/upload/pdf/infosec-mgt-principles.pdf

[50] www.GovernmentSecurity.org.

[51] www.spidynamics.com/paper/SQLIWhitePaper

[52] M.Howard and D.LeBlanc,"Writing Secure Code,"Microsoft Press, Redmond, Washington.

[53] S.Kost,"An introduction to SQL injection Attacks for Oracle Developers.pptx

AUTHOR'S PROFILE

Chandershekhar Sharma received B.Tech and M.Tech in Computer Science and Engineering from Kurukshetra University, Kurukshetra. He is Research Scholar in Computer Science and Engineering Department at Rajasthan technical university (Kota).His research interests include Web application security, Information security, High performance Computing and Threat modeling

Dr S.C.Jain has done his PG in Computer Science and Technology from IIT Roorkee and PhD in VLSI design from IIT, Delhi. He has served Defense Research and Development organization, Bangalore, India and presently working as Professor, Computer Science and Engineering, Rajasthan Technical University, Kota, India. His Research interests are VLSI design, Real Time Embedded System, Reversible Computing and High Performance Computing.

Dr. Arvind K Sharma holds PhD degree in Computer Science. He has more than 13 years of work experience in academics field. He has published more than 27 Papers in various National, International Journals and Conferences. He has authored & co-authored almost 5 books. He has received Best Paper Award in International Conference, 2012 held in Thailand. He has visited Thailand & Dubai for attending International Conferences. He has participated as Speaker & Keynote Speaker in many National and International Conferences. He is a Sr. Member of numerous academic and professional bodies i.e. IEEE, WASET, IEDRC, IAENG Hong Kong, IACSIT Singapore, UACEE UK, ACM, New York. He is a Member of Technical Advisory Committee of many International Conferences in India and abroad. He is also Editorial Board Member & Reviewer of several National and International Journals. Besides it, he is serving as a recognized Research Supervisor to guide Masters and PhD scholars in many Universities of Rajasthan, India. His area of interest includes Web Usage Mining, Web Intelligence Applications, Web Data Mining, Big Data Analytics and Machine Learning Tools.