The Solutions of SQL Injection Vulnerability in Web Application Security

Syamasudha V, Abdul Raheem Syed, Gayatri E

ABSTRACT--- Web Applications are commonly using all the services made available online. The rapid development of the Internet of Things (IOT), all the organizations provides their services and controlled through an online, like online transaction of money, business transaction of buying and selling the products, healthcare services, military and GPS Systems. Web application development and maintenance is very difficult based on the security. Attacks are many forms to stealing the secure, personal information and privacy data. There is one major open source community Open Web Application Security Project (OWASP) providing information, development and validation of web application projects to make application to be secure. This research work, discussing few of the solutions, detection and prevention methods of Injection risk out of the top 10 OWASP risks. Due to the injection risk, impact on business that may lead to loss of information, unauthorized access of personal and secure information.

Keywords: Web application Security, Information Flow, Injection flaw, Secure Compositions coding

I. INTRODUCTION

Most of the people are interacting and utilizing many services online for transferring money to other accounts, updating the information in the banking and e-business services, online purchasing services and so on, the web applications are more popular. Web applications, online services are more popular, because of all services are to be accessed from anywhere and anytime. Based on the vulnerabilities on the web application development, the user accounts and other secure information of the organizations are not secure. The attackers are attacks on the web applications and steal the private information and also organizations secure information with unauthorized access. The attackers are selecting unsafe web applications to steal the information with unauthorized access. Now a day many networks are actively monitoring intrusions using firewalls and Intrusion Detection Systems (IDS). So the development of web applications required to maintain more security standards from the unauthorized user access [1].

The entire World Wide Web is accessing the web applications, the vulnerable development of applications, attacks specific unsecured and malicious ways to access the unauthorized data. The web application security is the major problem for developing the web application from the business regions and e-business. The entire world depends mainly on the source of web applications for processing their requirements and is examined as important design and development of the world information association. On computer world the growth of social, financial and economic world the major role is played by the internet. SQL injection, Cross-Site Scripting (XSS), Sniffing, Encoding the requests are causing a huge warning on the internet usage. Based on these important attacks causes blocks and trouble in the communication by providing services and the clients request inputs. Also this reduces the web application performance. To fulfill the security of distributed systems, all types of organizations are maintaining standards of developing the application. Even through there are different types of attacks and number of attacks increasing because of some of the applications are not maintained standards of the security concepts at the time of developing the web applications [2].

The OWASP top 10 security risks are releasing in the year 2017. The 2017 top 10 risks in the web application security are listed following:

1. SQL Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging and Monitoring

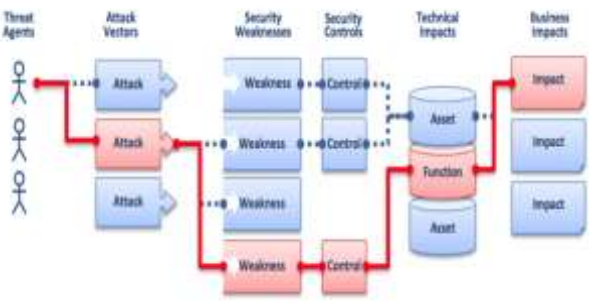

Fig 1. Top 10 OWASP Architecture Diagram

The web application two important and common risk are SQL Injecting and Cross Site Scripting (XSS). For SQL Injection, attackers modify the Query of SQL statement after

Revised Manuscript Received on 06 August, 2019.

Syamasudha V, Assistant Professor, CSE Department, Annamacharya Imnstitute of Science and Technology Tirupathi, A.P, India
(Email: syam.sudha7@gmail.com)
Abdul Raheem Syed, Research Scholar, School of Computing and IT Rev, University Bangalore, Karnataka, India.
(Email:rahamansd@gmail.com)
Gayatri E, Assistant Professor, CSE Department, SV Engineering College , Tirupathi, A.P, India
.(Email: gayatri.e13@gmail.com)

forwards to the database the retrieve the complete information based on the unsecured coding of the application. The attackers are inserting their altered query and then execute with their specified commands. In Cross-site scripting vulnerabilities web application first provides the user information to the browser to access the sensitive data without validation or marshaling that content [3].

In this paper, we describe, detecting and prevention mechanism of the Injection risk from the top ten vulnerability risk of OWASP.

II. LITERATURE SURVEY

In Web application Security SQL injecting many methods are proposed to detection, elimination also prevention of vulnerabilities. Here, we discussed with important solutions and associated working techniques are given the overview of each method. Few of the tools are used to detect and prevent the SQL Injecting. WebSSARI [5] is a tool released in 2004, this tool is used for analysis for detecting and finding the vulnerabilities in the PHP Applications. WebSSARI tool plays major role in finding three types of vulnerabilities from OWASP top 10. Those are SQL Injection, Normal Scripting Injection and HQL Injection. One more approach user cooperatively for PHP applications for finding the vulnerabilities in input validations explained in [6, 7]. This is used for finding the vulnerabilities in intra module, Cross-Site Scripting and also for the SQL Injection. This method is implemented for finding whole static PHP analyzed in the form of PHP-based Choices.

Fig 2. Overview of SQL Injection

The major implementation of web application security is like intrusions detection and prevention to protecting organization personal information and the information of an enterprise business system with remote access has more difficult. In these types of susceptible applications, the Injection of SQL query as input to an updated command for retrieving entire information like unauthorized data, personal information which is sensitive from the database. This type of input updated SQL command initiates the connection with unauthorized access of data base. The SQL Injection detection is very difficult without an attack on before happens. Many types of situations like the attackers are using same user credentials illegally, accessing the delegate code on the web applications can access and modify the command which is related to execute the query, illegal access grant the database. Based on these situations the major questions are rising like what are the causes and results, what is the latest, most popular SQL injections using are and what level of efficiently detecting and preventing the SQL Injections from the attacks [8].

III. SQL INJECTION VULNERABILITIES

The vulnerability of SQL Injection the hackers or attackers have the chance to process and inject the input through query statement execution for steal the entire information. For the understanding of SQL Injection vulnerability, first of all we must know how the server side programming are processing and controlling the SQL statements. With the help of example, suppose the real-time application produces SQL Statements with the help of input parameters, the statement generated as follows:

$sqlStatement = "select * from usersAccTable where user_name = 'abdul' and password = 'abdulsecretpw'";

The above SQL query is executed with the help of any specified method which passes the QSL query that method associated with the database where it converted in to Structural Query Language, processes and returned back associated output to the method or the application. In the SQL Query you may have seen and observe some updated and special symbols:

* (asterisk) symbol is the guidance to the Structural Query Language parser and the database for restore whole associated selections chosen from the data base.

= (equals) symbol is the guidance to the Structural Query Language parser to just restore the esteems which matches character string that associated from the table and database

'(Single quote) symbol is explains and guided to the Structural Query Language parser that the starts the search string to matches and close the search string. Presently take these are the samples in the web application employees are possible to modify and update the parameters of '$user' and '$password', of the home page to process the login page:

$sqlStatement = "select * from usersAccTable where user_name = '$user' and password = '$password'";

Attackers may passed and update all the special characters simply on SQL query, weather insertion parameter is not disinfected through request passed:

$sqlStatement = "select * from usersTable where user_name = 'adminUser'; -- ' and password = 'anyThing'";

From the above Structural Query Language statement the “adminUser'; --“is the input parameter passed by the hacker of having two new and special symbols.

; Semicolon is denoted for Structural Query Language parser which the query statement was finished.

-- Double hyphen is used to instruct the Structural Query Language parser by which reset of line, is not a comment and that should not be processed.

The process of executing the above SQL Injection query eliminates the password checking of the username -- admin and also restores the selected records form the database. From this SQL injection the hackers restore the complete information of the user account also enter as an administrator with successful login without knowing and validating the password in to the database.

SQL Injection Vulnerability Types

The hackers can get the complete information from the database server by misusing the injecting Structural Query

Language vulnerabilities on many forms. The most simple and regular as usual strategy for retrieving information from the data base is mainly checking error conditions like true or false and timing. The below are the many conditions that are to be followed.

Error-Based SQL Injection

While using the error-based Structural Query Language vulnerability, the hackers can restore the data like name of the table and content from the noticeable error of the database.

Error-Based SQL Injection Example

https://samplewebsite.co.in/indexPage.php?id=1+and(select 1 FROM(select count(*),concat((select (select concat(database())) FROM database_schema.tables LIMIT 0,1),floor(rand(0)*2))x FROM database_schema.tables GROUP BY x)a)

An Error was Returned by the Request

Using 'group_by' key suing from the database_new of the duplicate section.

The similar technique like content and names of the table works for retrieving data form the database. In the production environment, disabling error messages that leads to eliminate the hackers from collecting and misusing similar data content.

Boolean-Based SQL Injection

In the failure of SQL query execution, there is no error specific messages are not displaying on the page, hard to the attackers can steal the data from the vulnerable application. Even though they are having many ways to stealing and retrieve the data from the server and manipulate. In the failure of SQL query execution, some situations portion and some section of web page vanishes or modified or the whole page site cannot be load. Based on these type of signals are enables the hackers to decide if parameters passed as inputs are vulnerable and also if it permits retrieve the information.

From the above conditions the hackers can try by injecting these conditions to Structural Query Language query:

https://samplewebsite.co.in/indexPage.php?id=1+AND+1=1

Whether the web application page loaded normally, that defines helpless against injecting SQL query vulnerable. Certainly, the hacker commonly attempts to tries to produce a fake or wrong output utilizing similar below:

https://samplewebsite.co.in/indexPage.php?id=1+AND+1=2

As long as false condition, even the outcome is not getting also or the web page was not loaded properly also, that specify that the web page is vulnerable due to SQL injection. The below is the sample that defines how the hackers extract information as above specifies:

https://samplewebsite.co.in/indexPage?id=1+AND+IF(version()+LIKE+'5%',true,false)

From the above query processing, the web page must executed and displayed as normally if the version of the database is 5.X. Still, the web page loads and behaves dissimilar than that of normal because of the different version of the database, specifying if it is vulnerable or not.

Time-Based SQL Injection

On many conditions, despite the fact of that a doubtful SQL query injection no other constraints specified impact on the outcome of the web page, it might even now be feasible to remove the data from basic database.

Attackers are decide this by directing the database to pause (waiting) an expressed specified time interval before answering. The web page is responded quickly, whether the there is no SQL query vulnerable. If the web page is not responding quickly, that means the SQL query vulnerable. That specifies that the attackers restore information, without any change of the web page. The syntax of SQL statement is same which applied on Boolean-Based SEL Injection Vulnerability.

But as per the predefined waiting time interval, the method 'true' is updated as that may be to process the query, like 'sleep(3)' which guide and instructs that reset for three seconds to the database:

https://samplewebsite.co.in/indexPage.php?id=1+AND+IF(version()+LIKE+'5%',sleep(3),false)

But whether web application loading takes more than normal page load it is good to expect that the 5.X version database.

SQL Injection Out-of-Band Vulnerability

In few situations, now a day, the specified way the hacker can restore the data from the database is with the help of out-of-bound method. Normally these attacks require forwarding the information from the server of database to the machine that is organized by the hacker. Hackers may utilize this type of techniques whether an SQL Injection vulnerability does not happen at a time once supplied the information is passed, but rather some time interval it will processed.

Out-of-Band Example

https://samplewebsite.co.in/indexPage.php?id=1+AND+(SELECT+LOAD_FILE(concat('\\\\',(SELECT @@version),' samplewebsite.co.in\\')))

https://www.samplewebsite.co.in/indexPage.php?query=declare @pass nvarchar(100);SELECT @pass=(SELECT TOP 1 password_hash FROM users);exec('xp_fileexist "\\' + @pass + '.samplewebsite.co.in\c$\boot.ini"')

From the above information passed to the server to process, destination builds Domain Name Service request to the attacker-owned domain, based on request outcome within the sub domain. This explains the specified hacker no need to visible the outcome of the SQL Injection, but suppose to wait till they get the request for the database server.

IV. IMPACTS OF SQL INJECTION VULNERABILITY

There are many points and propositions the hacker can do when utilizing an SQL injection on website vulnerability. Normally, it relies upon benefits of the client web application client associate server of the database. As per the utilizing the vulnerability of SQL injection the hacker can be

as follows:

Add, delete, edit or read content in the database

Source code information read and write from files on the database

These all are types and expertise of the attacker, but the utilization of this vulnerability that may leads to hack the entire database and the also the server of web application. From this we will understand and trained many ideas on how examine the injection vulnerability on the website by seeing the cheat sheet of SQL Injection. The best method to prevent the harm is to limit attack as much as possible. This is more responsible to contains many databases for many reasons.

Preventing SQL Injection Vulnerabilities

Scripting language is mainly not possible to control or not if the SQL query statement processed of Server. Everything they can forward a query string to the server of the database and also delay for return the response.

Most likely, there is a way to just disinfect client input parameter and guarantee a SQL injection is not suitable and in secure. Tragically, this is not the same situation. There may be a unlimited count of approaches to purify client contribution, from all around applying PHP's addslashes() to whole, right way to applying the sterilization to "clean" factors at the season of collecting factors at the situation of combining the SQL query itself, for example, enclose the above $_GET['id'] in PHP's mysql_escape_string() method. Be that as it may, registering cleansing at the inquiry itself is an extremely poor writing the queries, now standards of writing queries also hard for keep up also monitor. This is place framework of database have engaged the utilization of arranged queries.

SQL Injection Prevention using Prepared Statements

As per the prepared statements, consider the approach of printf executes also how it designs strings. Actually, you gather your string with placeholder as information to be embedded. Also apply the information in same arrangement from the placeholder. The prepared statement query work as fundamentally the same as idea, where rather than directly gathering your query string and processing the same, then you can save the prepared statement, passed input data values, also that collect and clears after processing. Fantastic! Presently there should never be another SQL injection again. From the, attacks of injection are continuously happening also greatest predominant attack techniques?

Insecure SQL Queries are a Problem

Basically, this may be comes poor for web application coders lethargy also absence of instruction and understanding. Unsecure statements of SQL are basic in nature to write, also safe SQL statements are somewhat difficult to build. From the above thing the example, the malicious attacker or the programmer can infuse or passes anything the individual in question in the same itself.

Few sample SQL Prepared Statement with explanation

Whatever it may be any case, with SQL prepared statements, many steps are follows. No main database system works like printf statement. In the MySQL, directly needs minimum two commands those are PREPARE and another EXECUTE. PHP, by means of the PDO library, additionally needs a comparative stacking methodology, as follow the below example:

```
$sQLStatement = $dbh->prepare("SELECT * FROM userAccounts WHERE USERNAME = ? AND PASSWORD = ?");
$stmt->execute(array($username, $password));
```

At first look, this isn't intrinsically risky and, on an average, expands each SQL query by just an additional line or two. Be that as it may, as this needs additional alert and effort in the interest of effectively drained and burdened developers, in many cases they may get a small lazy and cut corners, selecting rather to simple procedural mysql_query() instead of the further developed article arranged PDO prepare().

Next of this numerous programmers simply stay with what they know to take care of business and they by and large get familiar with the least demanding and best direct approach to execute SQL queries as opposed to indicating certified interest for enhancing what they know. Be that as it may, it is also problem due to no knowledge on SQL queries and poor in coding standards.

V. RESEARCH METHODOLOGY & RESULTS

The Attack of SQL injection contains injection or insertion or updating part or full SQL query in the form of data input either passed through client or browser to the web application. The SQL injection attack may get the personal and sensitive data, update the existing data using insert or delete or update commands, executing like the similar admin access to the database. The basic common type of web applications builds SQL Statements containing SQL syntax with the combination of user input supplied by the programmer [9]. For example the syntax of SQL Query is

SQL Query: select user_name, salary from employee where id=$id

From SQL statement the $id is the input passed by the programmer based on the user supplied input. So the final input is like static value depends on the user passed input data, and finally makes the SQL statement dynamically. As per the above query written by the programmer, the attacker can passes defined input data mixing with this query to execute the SQL Query should be like originally. The below query explains how the user supplied input data updating the SQL Logic on where clause. From this the where condition check the tautology is true or false. If where condition is false then SQL query will not execute but if the where condition is true then query will execute like where username =”abc” or 1=1. The change of where clause is the data supplied “10 or 1=1”to the logic “or 1=1” are below

SQL Query: select name, salary from employee where id=10 or 1=1.

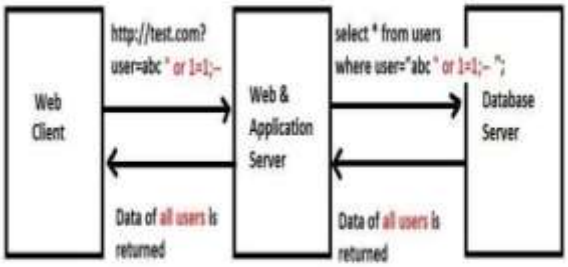

Fig 3. Flow of SQL Injecting execution

The SQL Injection is mainly three types Inband, Out-of-band and Inferential of Blind. In Inband the data are injected with the extraction SQL Code on the same level. Inband is one of the straightforward attack, from this received data is displayed in the web page itself. In Out-of-bound type the data is received based on a different channel like the resultant output is generated and sent to mail id or separate log file. In inferential type there is no actual data transfer, the tester only tests the behavior of database based on reconstructing the query. The actual SQL injection attacker can update the SQL query syntactically as per their required query to get the information. There any error message displayed by the web application based on the wrong or syntactical mistake query. This is the easiest way to reconstruct the original query to the attacker with injecting correctly.

There are mainly five techniques are used individually by combining or with another also in SQL Injection. The Union operator can be used for combining two select queries in to single result set. Boolean conditions are used for verifying few true or false conditions. Similarly, the Out-of-band method is used to get the information based on another channel based on establishing the connection through HTTP to forward the output to the web server. Finally, Time Delay technique is used in SQL queries in a conditional statement for getting results with a specified delay from the database. This technique is useful to the attackers till the result or error messages are not available from the application. The Injection defects are common because of legacy code. Injection defects are very commonly identified mainly in SQL or NoSQL, XPath and LDAP, ORM queries, SMTP Headers, XML parser and expression languages. The Injection vulnerabilities are very simple to find out based if you are reviewing the code. Similarly, few of the scenarios of the attackers are following below

String sqlQuery = "SELECT * FROM accountDataBase WHERE customerID='" + request.getParameter("custId") + " ' ";

Query HDLQuery = "session.createQuery("FROM accountDataBase WHERE customerID=' " + request.getParameter("custId") + " ' ");

The above both the query having vulnerable code, from browser the attacker send modified parameter value of the "id" like 'or '1'=1. https://somebank.com/app/accountDatabase?id=' or ' 1'='1

The final from the change of both queries, the account table all the information is returned from the database. Also, the attackers are dangerous sometimes they delete or update the table as their wish.

VI. OBSERVATIONS AND PROPOSED SOLUTIONS

Few of the common examples of Java and other UNSAFE code vulnerabilities are allowing attackers to processing the information of code in to the query for executing the database.

Using the un-validated parameter code is included to the executed query for process and retrieve data which they want from the table of the database to the attacker.

```
String sqlQuery = "SELECT userACBalance FROM account_table WHERE customerName = "
  + request.getParameter("userName");
 try {
      Statement stmt = connection.createStatement( connectionCodeStuff);
 ResultSet resultSet = stmt.executeQuery(sqlQuery);
 }
```

Safe Java Prepared Statement

Below sample code for the example uses a PreparedStatement, the parameterized SQL statement implementation, for executing the similar query.

```
String customerName = request.getParameter("userName ");
// for detecting the attacks to perform the input validation
String sqlQuery = "SELECT userACBalance FROM account_table WHERE customerName = ? ";
PreparedStatement preparedStmt = connection.prepareStatement(sqlQuery);
preparedStmt.setString( 1, customerName);
ResultSet tableReseult = preparedStmt.executeQuery( );
```

Prepared Statement of Hibernate Query Language

Unsafe HQL Statement:

```
Query hQLQueryUnsafe = session.createQuery("from InventoryTable where prodID='"+parameterSuppliedByUser+"'");
```

Safe HQL Statement of above query:

```
Query hQLQuerySafe = session.createQuery("from InventoryTable where prodID=: prodid");
hQLQuerySafe.setParameter("prodid", parameterSuppliedByUser);
```

Fig 4. SQL Injection implementation in Bank information

Safe Java Stored Procedure Example:

```
String userName = request.getParameter("custName"); //Must and should authorized
 try {
        CallableStatement callableStmt = connectionObjRef.prepareCall("{call storedproce_getAccountBalance(?)}");
        callableStmt.setString(1, userName);
        ResultSet finalResults = callableStmt.executeQuery();
 // processing the resultset
  } catch (SQLException sqlExcelption) {
        // … Handling errors also maintain logs
  }
```

VII. CONCLUSION AND FUTURE WORK

This paper described about different studies and review suggests to both programmers and researcher scholars of the web application security Injection vulnerabilities from the consistent reference to OWASP top 10 Security risks. How the attackers are identifying vulnerable code and also appending supported code for user supplied input to the executed query for retrieving, modifying and deleting the sensitive and personal information of the table from the database. This research work is to understand, detect and few of the prevention methods of Injection risk in web applications. In future work, we will discuss few more SQL Injection, Blind SQL Injection and SQL Injection Bypassing though Web Application Firewall (WAF). Also, the web application security design and development life cycle using different techniques and tools for avoiding effectively.

VIII. REFERENCES

1. Ge, X., Paige, R.F., Polack, F.A., Chivers, H. and Brooke, P.J., Agile Development of Secure Web Applications, International Conference on Web Engineering, Vol.11,Issue-14, Pg. 305-312
2. Monika Sachdeva, Roorkee, monika, "Performance Analysis of Web Service under DDoS Attacks", IEEE International Advance Computing Conference (IACC) 2009.
3. "https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf "
4. Vandana Dwivedi, Himanshu Yadav, Anurag Jain "Web Application Vulnerabilities: A Survey" International Journal of Computer Applications, Volume 108 – No. 1, Pg. 0975 – 8887, 2014
5. Y.-W. Huang, F. Yu, C. Hang, C.-H. Tsai, D. Lee, and S.-Y. Kuo. "Securing Web Application Code by Static Analysis and Runtime Protection". International World Wide Web Conference (WWW"04), pages 40–52, 2004.
6. N. Jovanovic, C. Kruegel, and E. Kirda. Pixy, "A Static Analysis Tool for Detecting Web Application Vulnerabilities". IEEE Symposium on Security and Privacy, 2006.
7. N. Jovanovic, C. Kruegel, and E. Kirda. "Precise Alias Analysis for Static Detection of Web Application Vulnerabilities". ACM SIGPLAN Workshop on Programming Languages and Analysis for Security (PLAS"06), June 2006.
8. Lawal Muhammad Aminu, A.B.M. Sultan, A.O. Shakiru, "Systematic literature review on SQL injection attack" in International Journal of Soft Computing Vol.11, Iss.1,Pg. 26-35 · January 2016
9. "https://www.owasp.org/index.php/Testing_for_SQL_Injection_(OTG-INPVAL-005)"
10. "https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet"