"SQL Injection attack in web application and its mitigation using prevention mechanism to secure Web Application Database"

Mital Parekh*[1], Prof. Chandresh Parekh*[2.]
[1.]Student in M. Tech, Cyber Security in Information Technology and Telecommunication, Raksha Shakti University, Ahmedabad, Gujarat, India.
2. Assistant Professor in Department of Information technology

ABSTRACT:

Now a day's cyber-attacks are increasing extremely on web application, mobile apps, networks which are more vulnerable for websites, web application. These attacks are used to gain unauthorized access into the system, data breach, credentials of the legitimate users and then penetrate the system. In vulnerability Assessment and Penetration testing the most vulnerable attack on websites, web application is SQL Injection. SQL Injection is a type of attack in which attacker trying to fetch database by manipulating SQL Queries. SQL Injection is common but the most popular attack in which the attacker intercepts the http request in which SQL query is passed with parameters, so attacker gain the access on SQL database in backend. On the basis of that attacker can find out the database names, table names, columns, and its row data too. This attack is the most vulnerable because attacker attacks on the database of the website and retrieves data from the tables. Authentication bypass, Data Breach, read source code from files on the database server, admin panel access can be done with SQL Injection. In this research paper a mechanism shall be developed which can be helpful to prevent the SQL Injection on the websites, web-apps. SQL injection is hard to prevent, although some solutions in this research paper can be fruitful for providing some better solution to prevent the SQL injection.

Keyword: SQL Injection, SQL Injection in web app, SQL Injection attack, SQL Injection vulnerability, Methods of SQL Injection attack

1. INTRODUCTION

SQL Injection is widely popular among web application vulnerability as mentioned in OWAPS (Open web application security project) platform. SQL Injection is widely done since many years yet there are some still problems in securing the database. SQL is a traditional database which is most widely used in all web applications. SQL is a structured query language in which database communicates with application through the SQL query. SQL databases like MSSQL, MySQL, SQLite, Microsoft Access. These databases are relational databases and which stores data in database structures. However, SQL Injection is most important as an attacker manipulates the query which database cannot understand, so it will throw an error. By manipulating the query an attacker can gain confidential, sensitive and secret data of the websites like credit card details, medical data, government critical data. So it becomes concern to secure the database from the injection.

SQL is structure query language is tightly bind with the database. In SQL Injection an attacker inserts malicious code into SQL query and that query passes with HTTP request and parameters are passed with login credentials id, username, password, so that an attacker finds vulnerable parameters and an attacker manipulates the query and uses various payloads and database exploit triggers.

2. BACKGROUND

Rapid advancement in web technologies has expedited the rate of adoption of database driven web application. The backend database servers of this web application accumulate some general data along with critical and sensitive information about the organization and clients. The most hazardous attack on web application is SQL Injection. SQL is a Structured Query Language specially designed for the database for creating, modifying, deleting database, tables. Injection is an attack in which the attacker manipulates the queries which is requested from client to the database server. SQL injection is mostly caused by the insufficient validation of user input. SQL Injection is a type of vulnerable threat penetrate the database and exploit the database vulnerability through various payloads.

An attacker can submit a query using 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 fascinated. With a little knowledge of SQL Command ad ingenious work to crucial table name SQL Injection attacks can be launched. These commands alter the desired output of queries to break into database.

Vulnerability Assessment and penetration testing is a part of cyber security and in which an attacker finds vulnerability and uses various payloads and exploit the vulnerabilities. Penetration testing is illegal activity in which an attacker gain access into the legitimate systems and gain unauthorized access, modifies the data, steal credentials. SQL Injection is a type of penetration testing in which attacker uses various payloads and insert

malicious code into the SQL query and database exploit triggers and sensitive data exposure.

3. LITERATURE REVIEW

SQL Injection is the most dangerous web threat on the web application and websites since many years. SQL injection vulnerabilities have been described as one of the most serious threats for Web applications [1]. OWAPS platform which generalize and monitors the most performing cyber threat on the web applications. SQL Injection is the most vulnerable and it causes the data breach, sensitive data exposure, remote code execution, privilege escalation, data modification, unauthorized gain of legitimate user. SQL is a relational database in which tables and data are tightly joint with each other.

SQL injection can be triggered by various types such as Attacker intent as login bypass, identify injectable data, extracting the data from the database, union query, stored procedure in which database logic is already defined at the database side, however an attacker manipulates the SQL query and causes the damage of database. SQL Injection can also be triggered when illegal or logically incorrect will be fired at that time an attacker lets an attacker gather important information about the type and structure of the back-end database of a Web application.

3.1 SQL injection through user input in this case, attackers inject SQL commands by providing suitably crafted user input. A Web application can read user input in several ways based on the environment in which the application is deployed. In most SQLIAs that target Web applications, user input typically comes from form submissions that are sent to the Web application via HTTP `GET` or `POST` requests. [3]. Improper input sanitization invites the SQL Injection and an attacker can bypass the login of the website.

3.2 SQL injection through cookies, cookies are client side variables which are used to store information related to user query in browser. Cookies are client side, and they store the information of the browser. Cookies are files which store web application information of user activity on browser. When a client returns to a Web application, cookies can be used to restore the client's state information. Since the client has control over the storage of the cookie, a malicious client could tamper with the cookie's contents. If a Web application uses the cookie's contents to build SQL queries, an attacker could easily submit an attack by embedding it in the cookies. [4]

3.3 SQL injection through server variables in this method server variables are a collection of variables that contain HTTP, network headers, and environmental variables. Web applications use these server variables in a variety of ways, such as logging usage statistics and identifying browsing trends. If these variables are logged to a database without sanitization, this could create an SQL injection vulnerability [5]. Because attackers can forge the values that are placed in HTTP and network headers, they can exploit this vulnerability by placing an SQLIA directly into the headers. When the query to log the server variable is issued to the database, the attack in the forged header is then triggered.

SQL Injection through Tautology attack can be performed using various techniques in which the most common techniques are tautologies in this method an attacker intension is to bypass the authentication, identifying the injectable parameters and extracting the data. The general goal of a tautology-based attack is to inject code in one or more conditional statements so that they always evaluate to true. The most common usages are to bypass authentication pages and extract data. In this type of injection, an attacker exploits an injectable field that is used in a query's `WHERE` conditional. [2]

Examples: In this example attack, an attacker submits " ' or 1=1 - " for the *login* input field (the input submitted for the other fields is irrelevant). The resulting query is:

```
SELECT accounts FROM users
WHERE login='' or 1=1 -- AND
pass='' AND pin=''
```

As shown in above example when SQL is fired at that time in database it will check the where condition and here 1=1 is universal truth and will bypass the login authentication and – is comment section which omits the remaining SQL query.

3.4 SQL Injection attack Union method in this technique, an attacker an exploits a vulnerable parameter to change the data set returned for a given query. With this technique, an attacker can trick the application into returning data from a table different from the one that was intended by the developer. Attackers do this by injecting a statement of the form: `UNION SELECT <rest of injected query>`. [2]. Union query is used to find the number of columns, vulnerable columns from the database.

http://kaizerpk.com/content.php?Id=3 union all select 1,2,3,4,5,6--+.

As here by applying above query an attacker get knowledge about the number of columns in the database and an attacker modifies the query and tries to find vulnerable columns in the database. When query triggers, SQL query will check the database interpreter will search the number columns in the database. And if the website is vulnerable to the SQL Injection, then website will display number of columns on the web page of website. So an attacker easily can identify the number of columns in the database.

3.5 SQL injection can through Piggy Backed query in which an attacker does not manipulate the query and attackers are not trying to modify the original intended query; instead, they are trying to include new and distinct queries that "piggy-back" on the original query. As a result, the database receives multiple SQL queries. The first is the intended query

which is executed as normal; the subsequent ones are the injected queries, which are executed in addition to the first. This type of attack can be extremely harmful. If successful, attackers can insert virtually any type of SQL command, including stored procedures, into the additional queries and have them executed along with the original query. Vulnerability to this type of attack is often dependent on having a database configuration that allows multiple statements to be contained in a single string. [2].

3.6 SQL Injection through Stored Procedures. Stored procedures are used to store SQL queries within the stored procedure. When user requests for any query, the query will be passed through HTTP request using GET or POST method at the database side, so database interpreter executes the queries stored in the stored procedures and result will be displayed on the web application. The stored procedures do not use the standard Structured Query Language; it uses its own scripting languages which does not have same vulnerability as SQL but different vulnerability related to the Scripting language still exist. [2]

Example: CREATE PROCEDURE user

@username varchar2 @pass varchar2

@customerid int

AS

BEGIN

EXEC('Select customer_info from customer_table where username='

''+@txtusername '' ' and pass = ' ''+@txtpass

'' '

As shown in above Stored procedure when it will execute any user can insert the malicious data in the username and password, which can lead to service disruption. It is advised to not store critical and sensitive data into stored procedures.

3.7 SQL Injection through Alternate Encoding

In this method an attacker modifies the injected content in the way that the database exploit triggered. avoid detection by defensive coding practices and also many automated prevention techniques. This attack type is used in conjunction with other attacks. In other words, alternate encodings do not provide any unique way to attack an application; they are simply an enabling technique that allows attackers to evade detection and prevention techniques and exploit vulnerabilities that might not otherwise be exploitable. These evasion techniques are often necessary because a common defensive coding practice is to scan for certain known "bad characters," such as single quotes and comment operators.

As in alternate coding method, the query is passed with the encoding of char so database cannot understand the syntax and database displays the vulnerability. And an attacker injects the payloads in the query and database exploit triggers. As below:

http://kaizerpk.com/content.php?Id=-3/*!50000uNIon*/+/*!50000aLL*/+/*!50000sELect*/1,2,3,/*!50000gROup_cONcat(column_name)*/,5,6/*!50000

When above query triggers, in database interpret the statement as in database the asci values of the characters by doing encoding of the char are allowed. So database is not able to detect the SQL injection attack and query will display the table names.

3.8 SQL Injection through DDOS attack

DDOS is distributed denial of service attack. In this type of attack, an attacker sends multiple requests to the target server and the whole network get jammed. So end user cannot access the resources and service is not available. However, SQL injection with DDOS attack, in this attack an attacker creates the complex query and send multiple requests. In order to pursue with this type of attack there are the basic steps to be followed which has to be done by finding the vulnerability, preparing for the vulnerability and after that the complex code is used for the attack. The greater the number of columns and rows in the database it will be easier for the SQL DDOS attack. Hence the sample code is used to make SQL DDOS attack on the website.

```
select user from (select
decode(encode(convert(compress(post)
using
latin1),concat(post,post,post,post)),
sha1(concat(post,post,post,post))) as
user from table_1)a;
```

Fig. 1: Sample code for the SQL Injection DDoS Attack [8]

As shown in above fig.1 when above query will trigger, in database complex query passes which cause encoding and decoding of the query and it will try to fetch the database name and also find the vulnerability of SQL injection.

3.9 SQL Injection Detection and Prevention Mechanism

There are various prevention and detection mechanisms are developed which are used to prevent the SQL Injection attack.

3.9.1 Black Box Testing.

Huang and colleagues [6] propose WAVES, a black-box technique for testing Web applications for SQL injection vulnerabilities. The technique uses a Web crawler to identify all points in a Web application that can be used to inject SQLIAs. It then builds attacks that target such points based on a specified list of patterns and attack techniques. WAVES

then monitors the application's response to the attacks and uses machine learning techniques to improve its attack methodology. This technique improves over most penetration-testing techniques by using machine learning approaches to guide its testing. However, like all black-box and penetration testing techniques, it cannot provide guarantees of completeness.

3.9.2 Static code Analysis

JDBC-Checker is a technique for statically checking the type correctness of dynamically-generated SQL queries [7,4]. This technique was not developed with the intent of detecting and preventing general SQLIAs, but can nevertheless be used to prevent attacks that take advantage of type mismatches in a dynamically-generated query string. JDBC-Checker is able to detect one of the root causes of SQLIA vulnerabilities in code— improper type checking of input. However, this technique would not catch more general forms of SQLIAs because most of these attacks consist of syntactically and type correct queries.

3.9.3. Input sanitization

SQL injection is the most impactful web threat in web application since many years. However, it requires some fruitful prevention mechanism, so by proper input sanitization it is possible to prevent SQL injection. Proper input validation at the client side must prevent special symbols, keywords like UNION, ORDER BY which must not be passed in the http query of SQL. To prevent sql injection blocks the uppercase letters, special symbols in the query. Below fig shows sql injection code when improper input is provided in form.

Fig2. SQL Injection susceptible by improper input.

4. PROBLEM STATEMENTS

SQL injection is widely and most harmful attack which causes data breach, sensitive data exposure, privilege escalation, remote code execution which modifies data file stored on the database server. Although as mentioned in this research paper there are various detection and prevention mechanism mentioned, still due to lake of proper input validation, stored procedure data storage SQL injection is still most serious web threat for web application. Although there are various manual SQL Injection techniques are available which causes SQL Injection.

5. PROPOSED WORK

SQL Injection can be performed using different methodologies. Impact of an SQL injection in web in web application security is very harmful as an attacker can gain access unauthorized critical data of the website. In this paper in future work how an SQL injection can be performed using different techniques shall be mentioned in which some scenario parameters (id, username) which are vulnerable to an SQL Attacks are passed in query using GET method exploits the SQL injection and data can be easily fetched, while in other scenario parameters are not passed in POST parameter methods, web application firewall bypass techniques. Also shall provide prevention mechanism to secure the critical data of the Web application in which how developer can do secure coding and block some special symbols passed during SQL query to prevent the SQL injection.

6. CONCLUSION

SQL Injection is the most serious threat on web application. In this review paper, there are various techniques are mentioned which are used to inject malicious code into SQL query and causes the SQL Injection. Also SQL injection detection and prevention techniques are mentioned to detect the SQL Injection and prevention mechanism to be used for preventing the SQL injection attack. SQL injection is easy to detect but hard to prevent. In this research paper, useful techniques are mentioned for detecting and preventing SQL injection as by providing proper input sanitization, using secure coding. This is the review paper which addresses various SQL injection techniques which addresses various types of SQL Injection methods.

7. REFERENCES

[1] T. O. Foundation. Top Ten Most Critical Web Application Vulnerabilities, 2005. http: //www.owasp.org/documentation/topten.html.

[2] William G.J. Halfond, Jeremy Viegas, and Alessandro Orso, College of Computing, Georgia Institute of Technology A Classification of SQL Injection Attacks and Countermeasures.

[3] N. W. Group. RFC 2616 – Hypertext Transfer Protocol – HTTP/1.1. Request for comments, The Internet Society, 1999.

[4] M. Dornseif. Common Failures in Internet Applications, May 2005. http://md.hudora.de/presentations/ 2005-common-failures/ dornseif-common-failures-2005-05-25.pdf.

[5] T. M. D. Network. Request.servervariables collection. Technical report, Microsoft Corporation, 2005. http://msdn.microsoft.com/library/default. asp?url=/library/en-us/iissdk/html/ 9768ecfe-8280-4407-b9c0-844f75508752.asp.

[6] Y. Huang, S. Huang, T. Lin, and C. Tsai. Web Application Security Assessment by Fault Injection and Behavior Monitoring. In *Proceedings of the 11th International World Wide Web Conference (WWW 03)*, May 2003.

[7] W. R. Cook and S. Rai. Safe Query Objects: Statically Typed Objects as Remotely Executable Queries. In *Proceedings of the 27th International Conference on Software Engineering (ICSE 2005)*, 2005.

[8] Z. Javanicus, "http://www.securityidiots.com/web-pentest/sqlinjection/ddos-website-with-sqli-siddos.html," 2016.