ISSN:0975-9646

A Survey on SQL Injection Attack, Detection And Prevention Techniques

Atul M. Shegokar[1], Arati K. Manjaramkar[2]

[1]M.Tech Student [2]Associate Professor

Department of Information Technology

Shri Guru Gobind Singhji Institute of Engineering and Technology Nanded, India

Abstract— **SQL Injection Attack causes a very serious security issue over web applications or websites. In this attack, Attacker is able to take benefit of poorly coded Web application software to put malicious or unwanted code into the organization's systems and network. The vulnerability exists within web application when a Web application does not provide proper validation or filtering for the input data entered by the user in the Input fields. In today's world there are large numbers of web application which are having many input fields where Hacker can get chance to attack as a SQL Injection (E.g. To dump the database contents to the attacker). So Attacker can access the confidential data of the organization. We are going to present a survey of SQL Injection attack, detection and prevention techniques in this paper .It Targets the back end data stores through web application inputs like forms, URLs etc**.

Keywords:**SQL Injection Attacks, Detection, Prevention, Injection Vulnerability, Blind SQL.**

I. INTRODUCTION

Now a day's most web applications are being hacked using SQL Injection attacks method. This comes under top ten security threat in web applications. SQL Injection Attack is most serious attack because it is categorized Under Open Web Application Security Project(OWASP) Top 10 Attack with high risk, Fix ability as medium having the impact on server as data loss or corruption of DB, lack of responsibility or denial of access to DB. It can sometimes lead to the complete host takeover [1]. SQL Injection Attack is most effective method for accessing the data from DB (Backend), with the help of this attack attacker can reach to the database and lift or steal sensitive information. The cause of SQL Injection is Malicious User input being treated as real DB Query. Commonly this attack is generated from web input so we can say it also an input validation attacks. With the help of this attack Hacker can get table details, database schema, and also Hacker can employ DML statement from the supplied input filed to web application to the database server which resulting a corrupt/ modified database.

SQL Injection has become a common issue with database-driven web application. The attack is easily detected, and exploited, and as such, any site or software package with even a minimal user base is likely to be subject to an attempted attack of this kind. Essentially, the attack is carried out by placing a Meta character into data input fields to then place SQL commands in the control plane, which does not exist there before. SQL injection attacks allow hacker to spoof identity of system, make the changes with**in** existing data, which causes repudiation issues such as changing balances or tampering to transaction, allow the complete disclosure of all data on the system, delete the data or make it otherwise unavailable, and become administrators of the database tier. SQL Injection is very common with various kinds of languages like PHP and ASP applications due to the prevalence of older functional interfaces. Due to the interfaces available in the JAVA and .NET applications are less vulnerable to have easily exploited SQL injections. Depending up the attacker's skill and imagination, such as low privilege connections to the database server and so on, the severity of SQL Injection attacks is limited.

This paper is organized in following section**s**. Section II provides the information about working of SQL Injection attack. Section III gives the detection and prevention mechanism for such attack. Section IV gives the Conclusion.

II.SQL INJECTION ATTACK

Fig.1 shows General Web Application Architecture which is divided into 3 tiers. 1] Web Tier, 2] Application Tier, 3] Database Tier. So out of the three tier the SQL Injection takes place at the first tier i.e. at Web Tier because Application tier and Database tier are typically stored well behind firewall so hacker often have access to Web tier. The Web tier is restricted to have access to Database tier. Actually the Application Tier handles the DB commands to prevent user from changing DB contents themselves. The Application tier acts as a Middle Man to DB to web users which don't have direct access to DB [8].

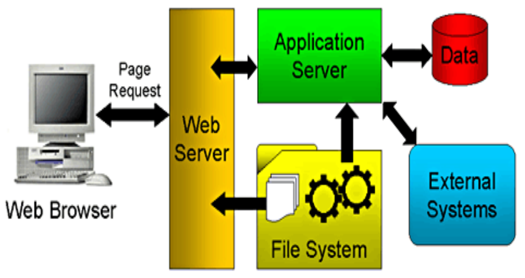


Fig. 1. Typical Architecture of Web Application

So by using web application the attacker is able to send invalidated inputs to the web server & then web server responds to database server, Database server gives the expected result as in the query passed from web server. The un validated input means the server is accepting modified query instead of input as string or by parameter tampering.

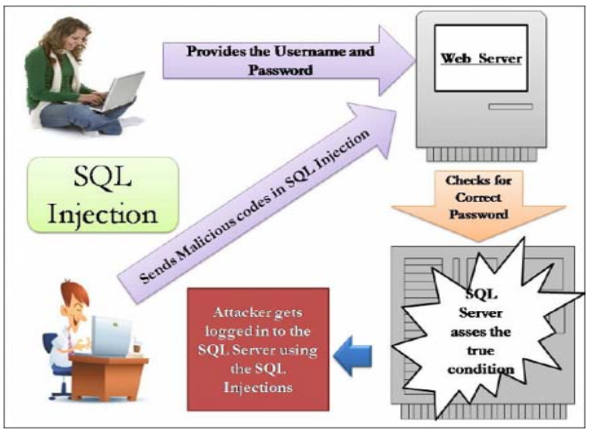

Provides the Username and Password
Web Server
SQL Injection
Sends Malicious codes in SQL Injection
Checks for Correct Password
SQL Server asses the true condition
Attacker gets logged in to the SQL Server using the SQL Injections

Fig. 2. How SQL Injection Work.

Generally when a web application is communicating with the backend DB, it does so in the form of queries with the help of a DB driver as shown in Fig. 2. These drivers are dependent on the application platform being used and the type of Backend DB, such as IBM DB2, MYSQL or ORACLE.

The general login query would be look like this:

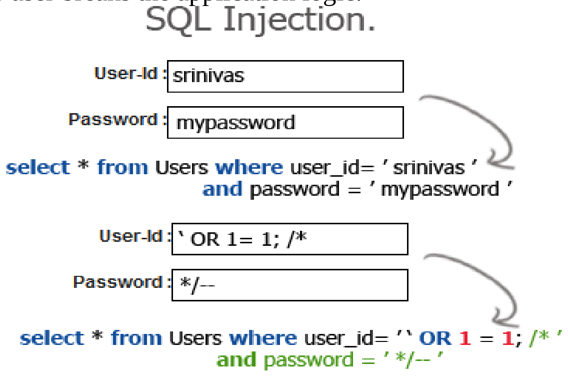
`SELECT Column1, Column2, Column3 FROM table_name WHERE username='$variable1' AND password= '$variable2';`

We will break down this query in 2 parts, Code section and the Data section. The Data section is the $variable1 and $variable2 and quotes are being used around the variable for defining the string boundary.

Let us try to see through the process in a unsophisticated way (Fig. 3). Say at the login form, the username entered is Admin and password isp@ssw0rd which is catch up by application and values of $variable1 and $variable2 are placed at their respective locations in the query, as is below

`SELECT Column1, column2, Column3 FROM table_name WHERE username='Admin' AND password='p@ssw0rd';`

Now the developer assumes that users of his application will always put a username and password combination to get a valid query for evaluation by DB backend. But if What is going to happen if the user is malicious and enters some invalidated input which will be having some special meaning in the SQL statement? For example putting a single quote. So, instead of putting Admin, he just puts Admin', so causing an error which will throw by the DB driver. Why? The reason is the unpaired quote entered by the user breaks the application logic.



```
select * from Users where user_id= ' srinivas '
                    and password = ' mypassword '

select * from Users where user_id= '` OR 1 = 1; /* '
                    and password = ' */-- '
```

Fig. 3. Example Scenario for SQL Injection

In the above Fig. 3 example scenario we are having two input fields User-Id and Password both fields are accepting string as input and we have entered as a normal user id as any name like srinvas and password as mypassword but the Attacker user is putting malicious input to get access to DB server table's data.

Normal Login Code:

```
String userid = request.getParameter("userid");
String password = request.getParameter("password");
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
connection =
DriverManager.getConnection("jdbc:odbc:projectDB");query =
"SELECT * FROM Users WHERE user_id ='" + userid + "' AND
password ='" + password +"'";
PreparedStatement ps = connection.prepareStatement(query);
ResultSet users = ps.executeQuery();
if(users.next()){
//some thing here
}
else{
}
```

The above code is normal login code in JAVA language where we entered user name and password in this code the user name and password entered in query is get from JSP page to servlet code by using request.getParameter() method then both of this values is used in query to select required user. As this is the normal scenario for login but attacker try for different UN validated input like as follows:

Injection Login Code:

```
$sql="SELECT * FROM users WHERE user="OR 1 = 1;//' and
password='....'";
```

Hacker is more intelligent than developer. So always try to hide the file extension (e.g.: *.jsp,*.php,*.asp)[7]

This is how SQL injection attack works there are various types of SQL attacks these various types of SQL attacks are generally not performed in separate way; Most of them are used together or separately, based on the specific objective of the hacker. Note that there are various types of each attack type. The various types of SQL Injection attacks are Illegal/Logically Incorrect Queries, Stored Procedures, Inference, Union Query, Piggy Backed Queries, and Tautologies [4]. The above attack comes under Tautologies because the objective of this attack is Bypassing authentication, identify inject able parameters,

Extracting data. There is one more type blind SQL Injection attack where attacker is trying his query again and again by modifying the parameter in the query because this case the error message is hidden by the developer from attacker.

III.SQL INJECTION ATTACK DETECTION & PREVENTION

If we want to protect a Web application from such Attack, there are two major concerns. First one, there is a great Necessity of a mechanism to detect and exactly identify SQL Injection attacks. Next ~~is~~ knowledge about SQL Injection Vulnerabilities is a must prerequisite for securing a Web Application. So far, many frameworks have been used and/or Suggested for detecting the SQL Injection attack Vulnerabilities in Web applications [3].

Researchers have provided a wide range of techniques for helping developers and helping for the shortcomings problem in the application. The first thing to understand when the web application starts to interact with DB for accessing some data from it. The example when application talks with DB are:

Authentication of form:-After performing authentication using web form so chances are the user credentials are verified against a DB which contain records for all user name & password.

Search Engines:-The string or text provided by the user for search a particular topic could be used for forming a SQL statement which then be used for extracting all related records from Database.

E-Commerce Sites: - Various products and their related information such as name, price, id, and quantity are stored in the database.

So for detecting SQL Injection attack the tester has to keep a list of all input fields of which value can be used in the SQL query. The very first test to be carried is adding a single quote (') or a semicolon (;) to the field or parameter under test, it is used as the string terminator and if it not filter by the application then it is lead to incorrect query. The second is used to end the for termination of SQL statement & if it is not filtered out correctly it will also generate error.

The following are some techniques which are used for detection of SQL Injection they are as follows:

A) Static Code Checker:

In Case of Static Code Checker for detecting the SQL injection attack your source code is being scan**ned** i.e. it performs static code analysis on source code also called as white box testing means when the code is at rest this type of code checking is carried out after scanning your source code depending upon the pattern present in your source code the attack detection takes place [2].

B) Black Box Testing:-

In this case your source code is being scanned by the standard Web Vulnerability Scanner (WVS) Tools like Acunetix, Burp Suite. These techniques uses the web crawlers for identifying the all points in the *web* application that can be used for carry out SQL Injection attacks [2, 9].

C) *Taint Based Approach:-* This approach detects the validation related errors with help of information flow analysis. [6]

Now moving towards the Prevention the SQL injection flaw is introduced into the web application when:

1. Developer creates a dynamic DB query that includes the user provided inputs so to avoid or prevent this attack the developer needs to either stop the writing the dynamic database queries and/or
2. Prevent user supplied input that includes malicious SQL Query which will affect the logic of normal executing query.

So the above technique can be used practically and any types of programming language with any DB.

Preventions are:

A) Use Parameterized Query:

The use of parameterized Query includes use of prepared statement. They are simple easy to understand than dynamic query. The parameterized Query forces the developer firstly defines complete SQL code and then pass in each parameter the query later. This coding style allows the developer to differentiate between the code and data regardless of what user is going to provide the input. This statement ensures that hacker is not able to change the content of query. So there are some language specific recommendations which developer must follow. [1]

B) Use of Stored Procedure:

Stored procedure is having same effect as that of prepared statements when making use of it. They needed the software developer to define structure the SQL code firstly, and then pass in the parameters. The difference between prepared statements and stored procedures is that the SQL code for a stored procedure is defined and stored in the DB itself, and then called from the web application. Both of these techniques have the same effectiveness in preventing SQL injection.

C) Using DB Accounts With Limited Permission:-

Give only necessary permissions to every account. Generally a web application uses an account to access the DB and denied user transaction at the application level. But, if a user is making use of sql injection, then all application level security will be going to bypass and the user will able to access to DB with the full access of account the web application uses to connect to the DB[5].

IV. Conclusion

As we know Vulnerabilities in today's web applications can perform malicious activity to gain unrestricted access to personal or private and confidential information. SQL injection attack is ranking at the top of the list of OWASP directed at any database-driven application written for the Web. This paper surveys the approach for SQL Injection attack & its various Detection and prevention techniques and enlists some of its types.

References

[1] OWASP Top 10 Web Application Vulnerabilities, https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[2] Diallo Abdoulaye Kindy and Al-Sakib Khan Pathan, *"A SURVEY ON SQL INJECTION: VULNERABILITIES, ATTACKS, AND PREVENTION TECHNIQUES"* 2011 IEEE 15th International Symposium on Consumer Electronics 978-1-61284-842-6/11

[3] K. Kemalis, and T. Tzouramanis (2008). *SQL-IDS: A Specification-based Approach for SQLinjection Detection.* SAC'08. Fortaleza, Ceará, Brazil, ACM: pp. 2153 2158.

[4] Puspendra Kumar R.K. Pateriya *A Survey on SQL Injection Attacks, Detection and Prevention Techniques* ICCCNT'12 26th _28th July 2012, Coimbatore, India

[5] http://www.cleverlogic.net/articles/sql-injection-depth-attacks-and-prevention-methods

[6] Devata R. Anekar Prof. A. N. Bhute *SQL Injection Detection and Prevention Mechanism using Positive Tainting and Syntax Aware Evaluation* International Journal of Advances in Computing and Information Researches ISSN:2277-4068, Volume 1– No.3,August 2012

[7] http://www.9lessons.info/2008/12/sql-injection.html

[8] http://securitycompass.com/computer-based-training/#!/get-free-owasp-course

[9] Zoran Djuric,"*A Black-Box Testing Tool For Detection SQL Injection Vulnerabilities*" ISBN: 978-1-4673-5256-7/13