Injection, Detection, Prevention of SQL Injection Attacks

Abhay K. Kolhe
Dept. Of Computer Engineering
MPSTME, NMIMS University
Mumbai, India
(Faculty)

Pratik Adhikari
Dept. Of Computer Engineering
MPSTME, NMIMS University
Mumbai, India
M.tech(student)

ABSTRACT
SQL injections have been always the top most priority for any website and web application. Every web application and website developed in php, asp.net, jsp which is connected to the database like MySQL, Microsoft SQL Server, and oracle are prone to SQL injection attacks. Most of the websites are created by using open source language such as php. The paper focuses the types of SQL injection attacks on the open source database in MySQL. The aim is to create a dummy website where users can login and register. The attacker can login these dummy website using different types of SQL injection, make changes in the database, detect these types of attacks using IP tracking methods with their injection types and to prevent them.

General Terms
Web security, SQL injection, detection and prevention techniques.

Keywords
MySQL, SQL injection, SQL injection vulnerability, web security, injection, detection, prevention of SQL injection

1. INTRODUCTION
Web security now a day is a major concern. Every web application, website is vulnerable to many attacks. The Majority of the web application and web site are made using PHP and MySQL as it is the open source for developing them. Open Web Application Security Project (OWSAP) has listed out the SQL injections as the top 10 vulnerability [1]. The goal of the paper is to find out the vulnerabilities related to SQL injection, provide the detection mechanism and prevention techniques for these attacks. The aim is to create a dummy website to perform the various kinds SQL injection attacks to get authentication, perform the variety of operations related to insert, update, delete, and drop. It also detects the IP address of the attacker for the particular type of attacks and provides the prevention method for the same.

1.1 SQL injection attacks
The SQL injection is the code injection technique, in which malicious SQL statements are inserted into an entry field for execution and used to attack the database and performs different types of the database interaction, operations and functions without sanitizing the inputs in the entry field [2].

The SQL injection generally fools the database as a regular query by the user and gets access to the system easily [3]. The paper is overall divided into three phase. Phases I is an injection phase where it will explain various types of SQL injection attacks, phase II deals with the detection of the phase I injection attacks with IP tracking mechanism and last phase III deals with the prevention mechanism.

2. PHASE I (Injection Phase)
This part gives the information about injection of the SQL queries for login, insert, update, drop and delete.

2.1 SQL Injection for Login
This part is divided into two sections first part is injection for the first row user and another is injection for the particular user such as admin.

2.1.1 Injection for 1st row user
An Attacker is able to get login into the website successfully by giving username: anything OR ‘x’=’x and password: anything OR ‘x’=’x as the following shown in the Fig 1. This is the type of SQL injections is for incorrectly filtered escape character. Later in the prevention phase this paper will discuss how it can be prevented. The table name as the “user” is made where the values of all registered users are stored.

Injection query can be explained as:

```
"SELECT * FROM user WHERE name = " + userName + " + password + " + ";"
```

If the “username” is replaced by SQL string, like anything’ or ‘x’=’x during authentication, the database responds to the code in the same manner as the first code and displays the records. This is because the evaluation of ‘x’=’x has always been true. So the attacker gets authenticated without any proper valid user name and password in the database. Here in this case single quote always allows the inside quote to get executed [3].

![Login Page](image)

Fig.1 SQL injection 1st row user

It can be seen in the Fig.2 that the attacker is able to login to the website without any username and password. In this case the row for the first user is always getting selected for the
login on the website. This form of the SQL injection is easy to check on any website. Most of the coders and developers forget to filter the escape character and this attack is always a vulnerability in PHP as well as asp.net for both MySQL and MS SQL server 2005, 2008. Consider the situation where the confidential medical reports can be leaked to the attacker where the attacker is not even registered in the database.

2.1.3 Injection for insert operation.

Here the attacker is injecting the insert query to the database as username, password, email, and the name for the particular user. Fig. 5 shows the injection for insert query.

![Fig. 5 SQL Injection for Insert Query](image)

2.1.4 Injection for update

The attacker is injecting the update query to the database and updating the username for the particular email address. Fig. 6 shows the injection for the update query.

![Fig. 6 SQL Injection for Update Query](image)

2.1.5 Injection for delete

The attacker is injecting the delete query to the database and he is deleting the particular row in the database based on the username. Fig. 7 shows injection for the delete query.
Fig. 7 SQL Injection for Delete Query

Table 3 shows the delete results. It can be seen clearly that the username “naresh” with id “37” has been deleted successfully.

Table 3. Delete operation

<table>
<thead>
<tr>
<th>Id</th>
<th>Username</th>
<th>Password</th>
<th>Email</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>thanasree</td>
<td><a href="mailto:thanasree2@gmail.com">thanasree2@gmail.com</a></td>
<td>thanasree abcd</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>admin</td>
<td>9920999801</td>
<td><a href="mailto:admin@gmail.com">admin@gmail.com</a></td>
<td>admin test</td>
</tr>
<tr>
<td>60</td>
<td>bob</td>
<td>bob123sat</td>
<td><a href="mailto:bob@gmail.com">bob@gmail.com</a></td>
<td>Bob Hello</td>
</tr>
</tbody>
</table>

Delete operations done for user id 37 “naresh”

2.1.6 Injection for Drop

To inject the drop query extra table named “student” was made for the first name and last name so the main table “user” doesn’t get altered. Table 4 shows the table name “student”.

Table 4. Student Table

<table>
<thead>
<tr>
<th>First name</th>
<th>Last Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prakrit</td>
<td>Adhikari</td>
</tr>
<tr>
<td>Nikhil</td>
<td>Kamath</td>
</tr>
<tr>
<td>Jayesh</td>
<td>Kabra</td>
</tr>
</tbody>
</table>

The attacker is injecting the drop query and is dropping the table name “student”. Fig. 8 shows the injection for the drop query.

Fig. 8 SQL Injection for Drop Query

Fig. 9 shows the results after dropping the table.

Welcome, Hacker  Logout

Table ‘prakrit.student’ doesn’t exist

Fig. 9 SQL Injection for Drop Query

The message is made using PHP and HTML coding after dropping the table this message appears as results, same table can be checked in the GUI for the database.

3. DETECTION BASED ON THE IP ADDRESS

For the detection part, the array has been made where the types of the operations that can be performed are listed out and the injection query is match out to an array. It can track down the list of operations that can be performed. The IP address and particular date and time are recorded here in the table. Table 5 shows the Detection Based on the IP address with date and time recorded [4].

Table 5. Detection Based on the IP address.

<table>
<thead>
<tr>
<th>Id</th>
<th>IP Address</th>
<th>Injection Type</th>
<th>Date &amp; Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>46</td>
<td>127.0.0.1</td>
<td>%</td>
<td>2013-11-14 22:08:56</td>
</tr>
<tr>
<td>47</td>
<td>127.0.0.1</td>
<td>=</td>
<td>2013-11-14 22:20:36</td>
</tr>
<tr>
<td>48</td>
<td>127.0.0.1</td>
<td>LIKE</td>
<td>2013-11-14 22:21:02</td>
</tr>
<tr>
<td>49</td>
<td>127.0.0.1</td>
<td>UPDATE</td>
<td>2013-11-14 22:21:52</td>
</tr>
<tr>
<td>50</td>
<td>127.0.0.1</td>
<td>INSERT</td>
<td>2013-11-14 22:22:41</td>
</tr>
<tr>
<td>51</td>
<td>127.0.0.1</td>
<td>1</td>
<td>2013-11-14 22:25:48</td>
</tr>
<tr>
<td>52</td>
<td>127.0.0.1</td>
<td>DELETE</td>
<td>2013-11-14 22:26:38</td>
</tr>
<tr>
<td>53</td>
<td>127.0.0.1</td>
<td>1</td>
<td>2013-11-14 22:27:37</td>
</tr>
</tbody>
</table>

4. PREVENTION TECHNIQUE FOR SQL INJECTION

This section deals the prevention mechanism for SQL injection using Escape routine and MySQLi.

4.1 mysql_real_escape_string()

Escapes special characters in the unescaped_string, taking into account the current character set of the connection so that it is safe to place it in a mysql_query()[5].

mysql_real_escape_string() calls MySQL’s library function mysql_real_escape_string, which adds backslashes to the following characters: \00, \n, \r, \', " and \xa. The query is filtered out before it is passed to the database. So attacker is unable to inject the query after using this mechanism[5]. It can be seen in Fig. 10 an on-click event was made, on clicking on the on-click event “mysql_real_escape_string” it changes its color to green and gets activated.

Fig. 10 mysql_real_escape_string()

It can be seen in the Table 6 that it is not possible to insert new query after applying mysql_real_escape_string so it get prevented.

Table 6. Using mysql_real_escape_string()

<table>
<thead>
<tr>
<th>Id</th>
<th>Username</th>
<th>Password</th>
<th>Email</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>thanasree</td>
<td><a href="mailto:thanasree2@gmail.com">thanasree2@gmail.com</a></td>
<td>thanasree abcd</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>admin</td>
<td>9920999801</td>
<td><a href="mailto:admin@gmail.com">admin@gmail.com</a></td>
<td>admin test</td>
</tr>
<tr>
<td>60</td>
<td>bob</td>
<td>bob123sat</td>
<td><a href="mailto:bob@gmail.com">bob@gmail.com</a></td>
<td>Bob Hello</td>
</tr>
</tbody>
</table>

Access Unauthorised

4.2 MySQLi

MySQLi optionally allows having multiple statements in one statement string[6].
Multiple statements or multi queries must be executed with mySQLi_multi_query(). The individual statements of the statement string are separated by semicolon. Then all the result sets returned by the executed statements must be fetched. The MySQL server allows having statements that do return result sets and statements that do not return result sets in one multiple statements [6]. An extra API call is used for multiple statements to reduce the likeliness of SQL injection attacks [6]. It can be seen in Fig.11.

Table 7. Using MYSQLI

<table>
<thead>
<tr>
<th>Id</th>
<th>Username</th>
<th>Password</th>
<th>Email</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>thomas.tree</td>
<td>than123</td>
<td><a href="mailto:thanasree@gmail.com">thanasree@gmail.com</a></td>
<td>thanasree abcd</td>
</tr>
<tr>
<td>36</td>
<td>admin</td>
<td>9920999801</td>
<td><a href="mailto:admin@gmail.com">admin@gmail.com</a></td>
<td>admin test</td>
</tr>
<tr>
<td>38</td>
<td>bob</td>
<td>bob123sat</td>
<td><a href="mailto:bob@gmail.com">bob@gmail.com</a></td>
<td>Bob Hello</td>
</tr>
</tbody>
</table>

Access Unauthorised

It can be seen in the Fig.12 by applying the preventing techniques, login error is displayed. So injecting both the queries for login is prevented and mySQL_error() is used, so the MySQL database on backend no longer issues warnings, which can give the valuable information to the attacker[7].

5. CONCLUSION AND FURTHER WORK
SQL injection has been always a topmost threat to any website or web application. Any web application or website that is developed PHP , asp, jsp with database Mysql, MS SQL server, Oracle is prone to SQL injection[8][9]. This paper provides the different types of SQL injection attacks (php and database Mysql) for e.g login without valid credentials and performing various operations to the database like insert, updates, delete, and drop. These attacks have been detected using IP tracking mechanism. MSQLi and mySQL_real_escape_string() technique have been used for prevention. Thus injection, detection and prevention of SQL injection attacks in this paper have been done.

6. REFERENCES