Project 1

Topic – SQL Injection

Introduction: -

Every web application has its own vulnerability. In SQL injection we exploit this vulnerability. SQL injection is a code injection technique in which we gain access to the database. The database should not be accessible to the user, but due to this technique the user gets access to the database. This is the most famous and commonly known attack which is used for tampering the data that already exists, one can also destroy the data by gaining access and mimic any particular user for their own personal benefits. One can also obtain admin rights of the entire database, thereby making it vulnerable.

Vulnerability: -

SQL injection can be easily applied on those servers which do not validate their users properly. There are many different types of vulnerabilities involved in which the application would accept any type of data from the source which is not trusted. The input can be in such a way that it intentionally matches the SQL query that is required by the web application to authenticate its users thereby gaining access to the user data. Consider the example, where a sample query is= "SELECT * FROM users WHERE username = " " +username + "" AND password =" " +password, the query being constructed using concatenation of the input string which is supposedly entered by the user, this would authorise the user only when the password contains a single quoted character. Here in this example the vulnerability is the SQL query that can be generated by changing the input and the exploit is the code that is used to send the SQL commands to the application.

Attack Vector: -

Attack vector is a path or the means by which anyone can gain access to the web server or to the system. The most common type of attack vectors in SQL Injection are form data which is sent using HTTP GET and HTTP POST methods. There is also an attack vector called HTTP USER-AGENT and HTTP cookie data.

Attack Surface: -

Attack surface in case of SQL Injection is the database itself. Here in SQL injection the queries are fired and they help in gaining the unauthorized access to the user data. They make use of the database to gain access thereby making the database, attack surface.

Attack occurrence: -

The SQL Injection technique is used to get access to the database by getting around the login process. The following example will explain the working of an SQL Injection attack. Here we make use of an HTML webpage form where the user needs to enter his username and password to gain access to his data. If the username and password are correct then the user gets a message of “Login Successful!”. HTTP POST or HTTP GET methods can be used for the request. However, in this example we make use of HTTP POST.

```
<form action=”/cgi-bin/login” method=post>
Username: <input type=text name=username>
Password: <input type=password name=password>
<input type=submit value=Login>
```

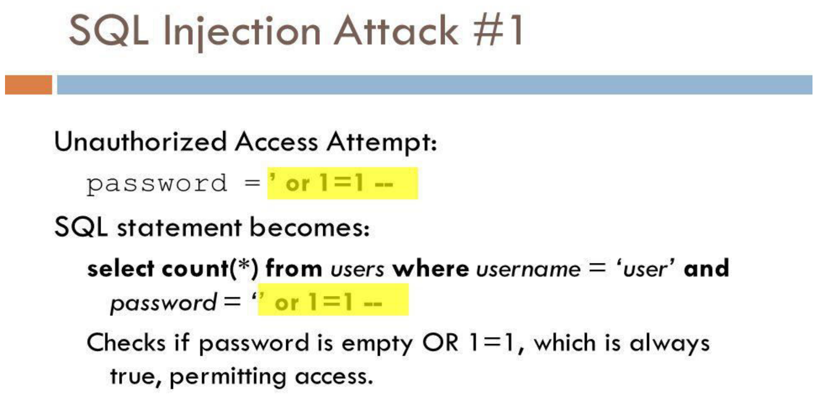
When the user enters the required username and password in the form specified above and presses the login button, the web browser submits the entered username and password in the form of string to the web server. The web server contains all the required user data which is necessary for login in. The string is contained in the body of the request as follows:

```
Username =enteredName&password=enteredPassword
```

The application does not validate the login process will give direct entry to the user without validating and making use of extra security steps by firing a SQL query.

```
SELECT * FROM Users WHERE (username = ‘enteredUser’ and password = ‘enteredPassword’);
```

This SQL statement will locate the profile of the user that contains the entered username and password. The application is vulnerable to an SQL injection attack if it does not use any input validation method.



SQL Injection Attack #1

Unauthorized Access Attempt:

password = ’ or 1=1 --

SQL statement becomes:

select count(*) from *users* **where** *username* = ‘*user*’ **and** *password* = ‘’ or 1=1 --

Checks if password is empty OR 1=1, which is always true, permitting access.

Here in the above example, when the user enters username and password as ‘or 1=1 --, then the SQL query becomes SELECT * FROM userpass WHERE user = ‘$user’ AND pass =’’ or 1=1 –

As the logic statement of 1=1 is always true, thus the attacker gets access to the database no matter what.

The basic idea is to give such an input which will change the logic of SQL statements to effectively remove the password check. In case of empty string of password, the SQL query will consider the password field to be commented and will ignore the password check thereby giving access to the attacker of the database.

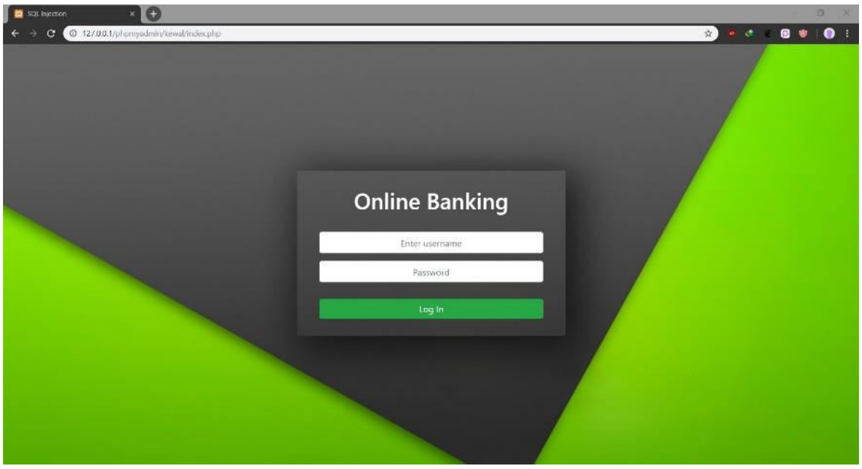
<u>How the attack works: -</u>

Basically, in every SQL attack there is a query which acts like a pathway for the attacker to the database. The SQL query becomes the command which gets the attacker entry to the vulnerable data. For this type of attack, one doesn’t require any special tools as such, the browser is enough for the attack. There are two situations for the attack to happen where the user can just enter the username and password and get access to the database and second where the user has to fill a php form which will be then passed to the server using HTTP POST method.

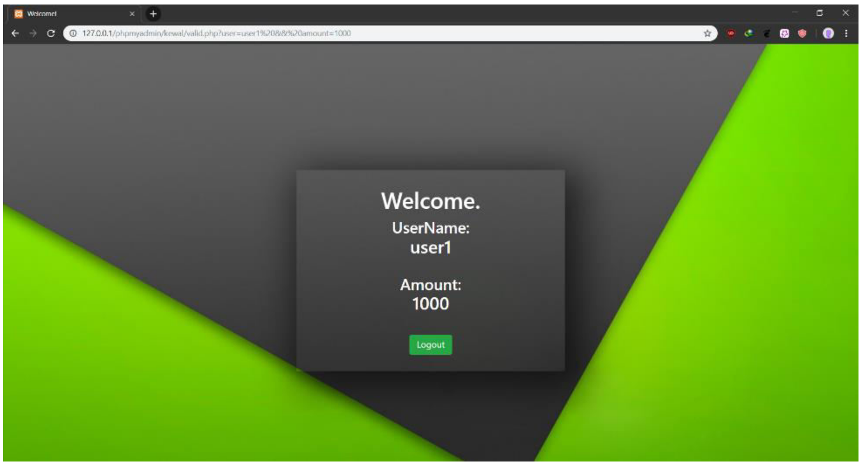
<u>Code Explanation: -</u>

A simple PHP code is used for the login page. The login page accepts the inputs from the user as username and password. The method used for sending the data is POST method. Xampp server is used for the database handling. login.php is the code file that is used to validate the inputs with the database. The query which is created has the form: SELECT * FROM userpass WHERE user=$user' AND pass='$pass'". Here the database is named userpass and the user and pass are its columns respectively. When the input given for the user is 1' or '1' = '1, it gives first row of the database as the output thereby authorizing the user. If the input is of the form 1' or '1' = '1' and user<>'user1 then the row which has name user1 i.e. the user user1 will not be selected and the next immediate user will be logged in. This is the basic working of SQL Injection. After logging in, one can also change the amount that is in the account. This can be done by entering custom amount in the URL of that webpage.

The following screenshots show the working of the project.



>Basic login form



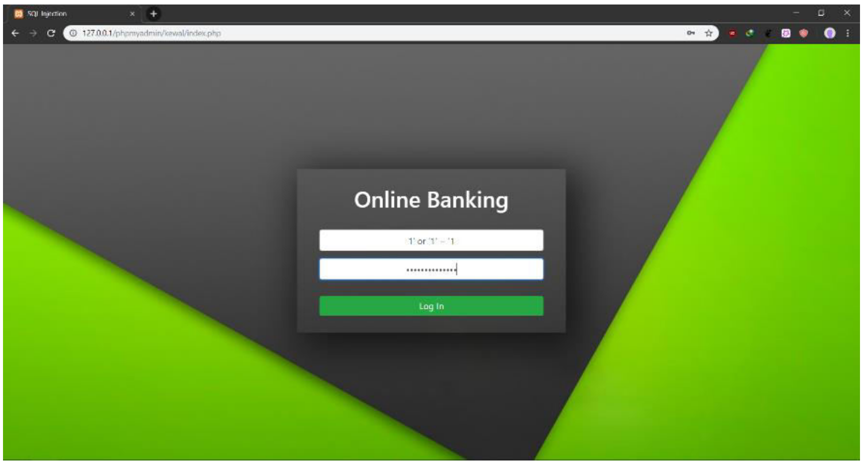
Welcome!
127.0.0.1/phpmyadmin/kewal/valid.php?user=user1%20&&%20amount=1000
Logout

Welcome.

UserName:
user1

Amount:
1000

>Entry to the database with valid username = user1 and password = 123


SQL Injection

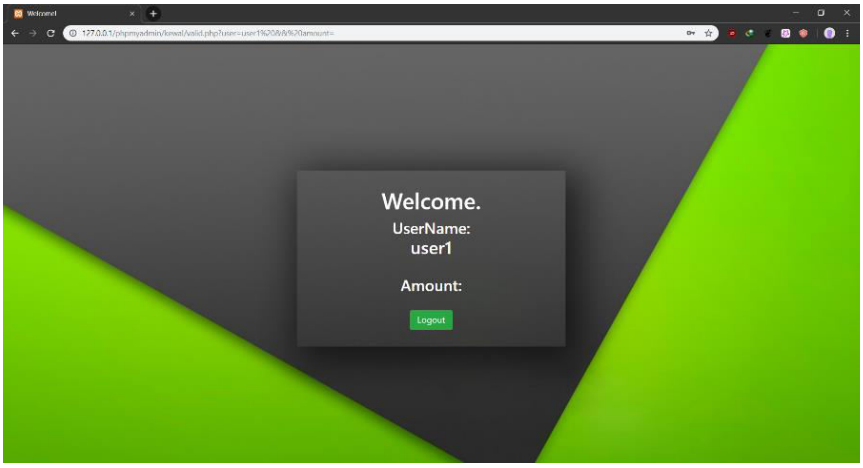
127.0.0.1/phpmyadmin/keval/index.php

Online Banking

1' or '1' = '1

••••••••••••••

>Here the entered logic statement is same for both username and password. Since the logic is true, the attacker gets access to the database.


Welcome!

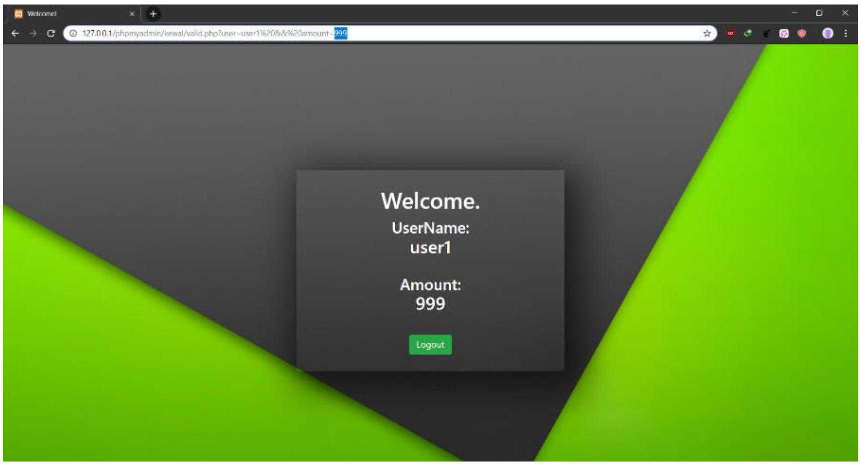
127.0.0.1/phpmyadmin/kewal/valid.php?user=user1%20&&%20amount=

Welcome.

UserName:
user1

Amount:

>Here the attacker got access without the proper username and password.


Welcome!

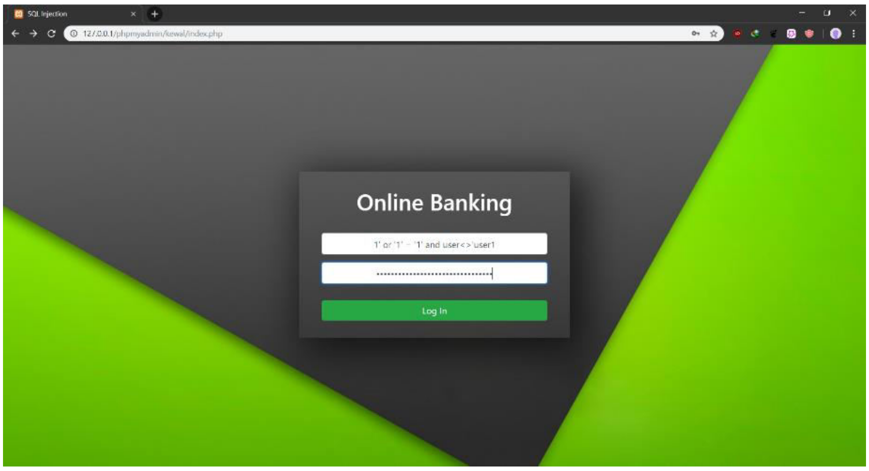
127.0.0.1/phpmyadmin/kewal/valid.php?user=user1%20&&%20amount=999

Welcome.

UserName:
user1

Amount:
999

>Attacker changes the amount value, after getting access to the database.



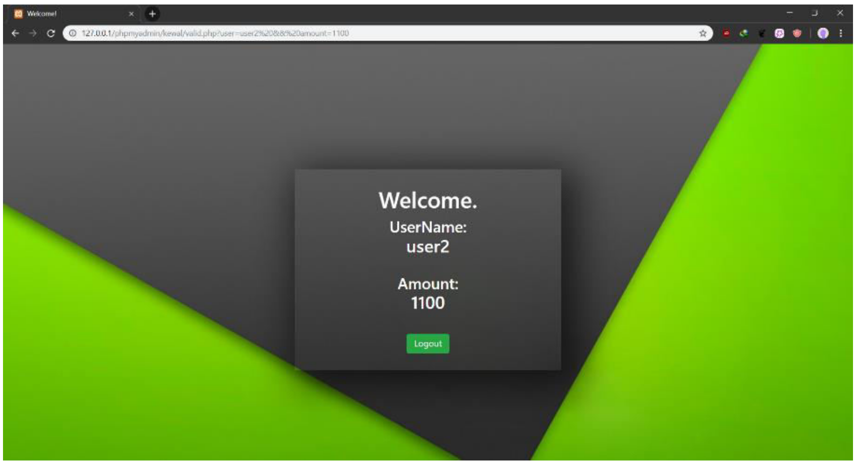
127.0.0.1/phpmyadmin/kewal/index.php

Online Banking

1' or '1' = '1' and user<>'user1

••••••••••••••••••••••••••••••••

> Here the attacker tries to enter a logic statement which makes use of NOT operator <>. This operator helps the attacker to get access to the user2 which is next in the database after user1.



127.0.0.1/phpmyadmin/kewal/valid.php?user=user2%20&%20amount=1100

Welcome.

UserName:
user2

Amount:
1100

> Attacker got access to the user2's data.

This is the working of SQL Injection.