CS6008 Cryptography and Network Security Assignment – 3

SQL INJECTION ATTACK IN PHP BASED WEBSITES

DONE BY

SABHARI P

2018103582

SQL Injection:

SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve.

This might include data belonging to other users, or any other data that the application itself is able to access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behaviour.

In some situations, an attacker can escalate an SQL injection attack to compromise the underlying server or other back-end infrastructure, or perform a denial-of-service attack.

Impact of a successful SQL injection attack:

A successful SQL injection attack can result in unauthorized access to sensitive data, such as passwords, credit card details, or personal user information.

Many high-profile data breaches in recent years have been the result of SQL injection attacks, leading to reputational damage and regulatory fines.

In some cases, an attacker can obtain a persistent backdoor into an organization's systems, leading to a long-term compromise that can go unnoticed for an extended period.

Purpose of SQL injection:

• Identify injectable parameters.
• Identify the database type and version.
• Discover database schema.
• Extracting data.
• Insert, modify or delete data.
• Denial of service to authorized users by locking or deleting tables.
• Bypassing authentication.
• Privilege escalation.
• Execute remote commands by calling stored functions within the DBMS which are reserved for administrators

Methods:

There are some methods through which the SQL statements are injected into vulnerable system.

• Injected through user input.
• Injection through cookie fields contains attack strings.
• Injection through Server Variables.
• second-Order Injection where hidden statements to be executed at another time by another function.

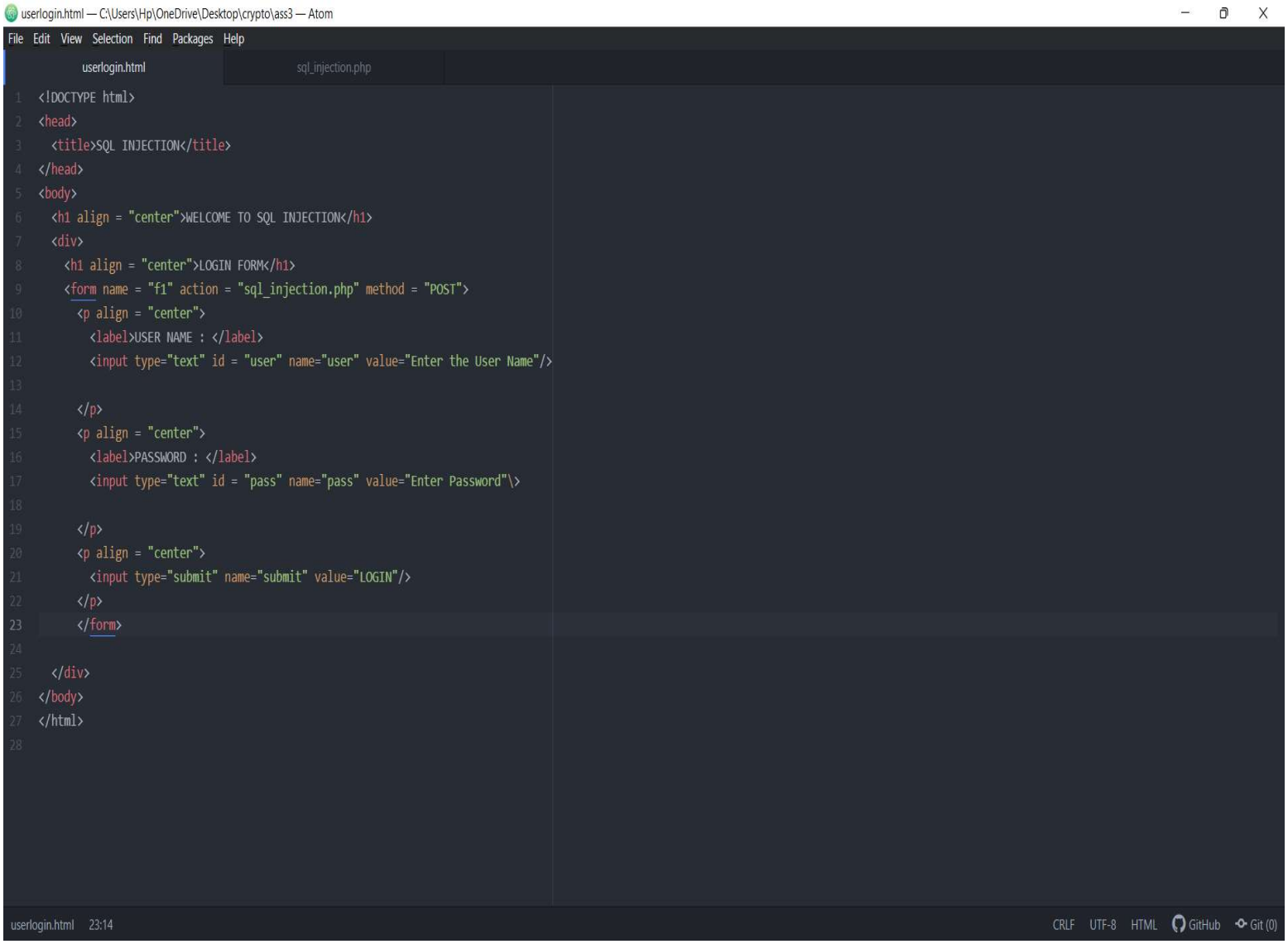
Occurrence of SQL injection:

• Your code uses unsanitized data from user input in SQL statements
• A malicious user includes SQL elements in the input in a tricky way
• Your code executes these SQL elements as part of legitimate SQL statements.

EXPLANATION WITH CODE SCREENSHOTS:

Firstly, let’s make a simple login form using html and PHP. This login form asks the user for the credentials like username and password for proceeding through and doesn’t allow access for incorrect credentials.

Userlogin.html



Sql_injection.php

sql_injection.php — C:\Users\Hp\OneDrive\Desktop\crypto\ass3 — Atom

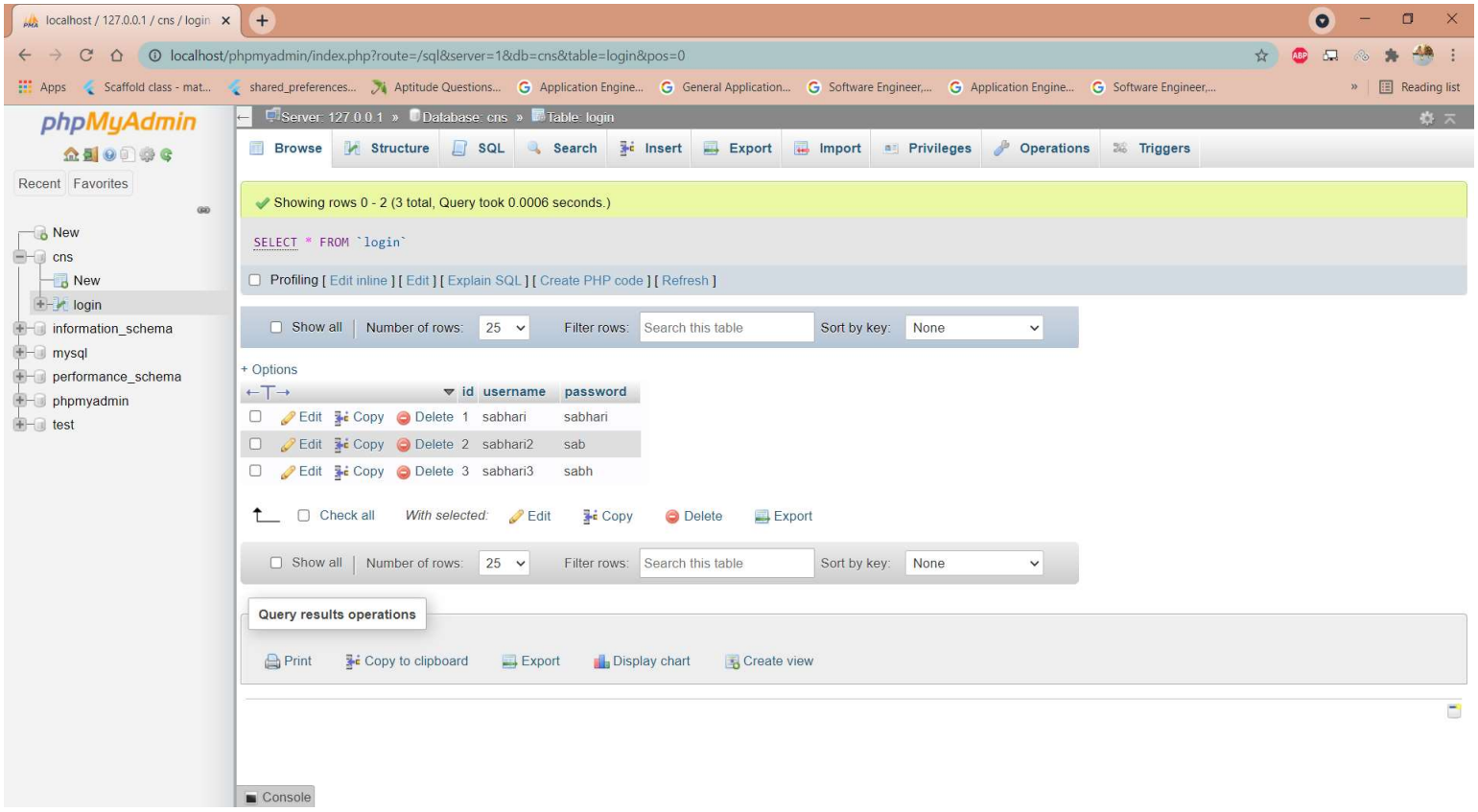
File Edit View Selection Find Packages Help

userlogin.html | sql_injection.php

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "cns";
$con = mysqli_connect($servername,$username,$password,$dbname);
if (mysqli_connect_errno()) {
  die("FAILED TO CONNECT  :  ".mysqli_connect_error());
  // code...
}
$username = $_POST['user'];
$password = $_POST['pass'];
$sql = "select * from login where username = '$username' and password = '$password'";
$result = mysqli_query($con,$sql);
$count = mysqli_num_rows($result);
if ($count > 0) {
  echo "<h1><center>SUCCESSFULLY LOGED IN </center></h1>";
  echo "QUERY  :  ".$sql;
}
else {
  echo "<h1>Login Failed, Invalid username or password</h1>";
}
if ($count > 0) {
  echo "<div>";
  echo "<table>
  <trbgcolor = '@ccc'>
  <th>USERNAME</th>
  <th>PASSWORD</th>
  </tr></table>";
  while ($row = mysqli_fetch_assoc($result)) {
    echo "<tr align = left style = 'font-size:20px;'>";
    echo "<td align = center>".$row['username']."</td>\t";
    echo "\t";
    echo "<td align = left>".$row['password']."</td>";
    echo "</tr>";
    echo "<br></br>";  }}
 ?>
```

sql_injection.php* 36:25 CRLF UTF-8 PHP GitHub Git (0)

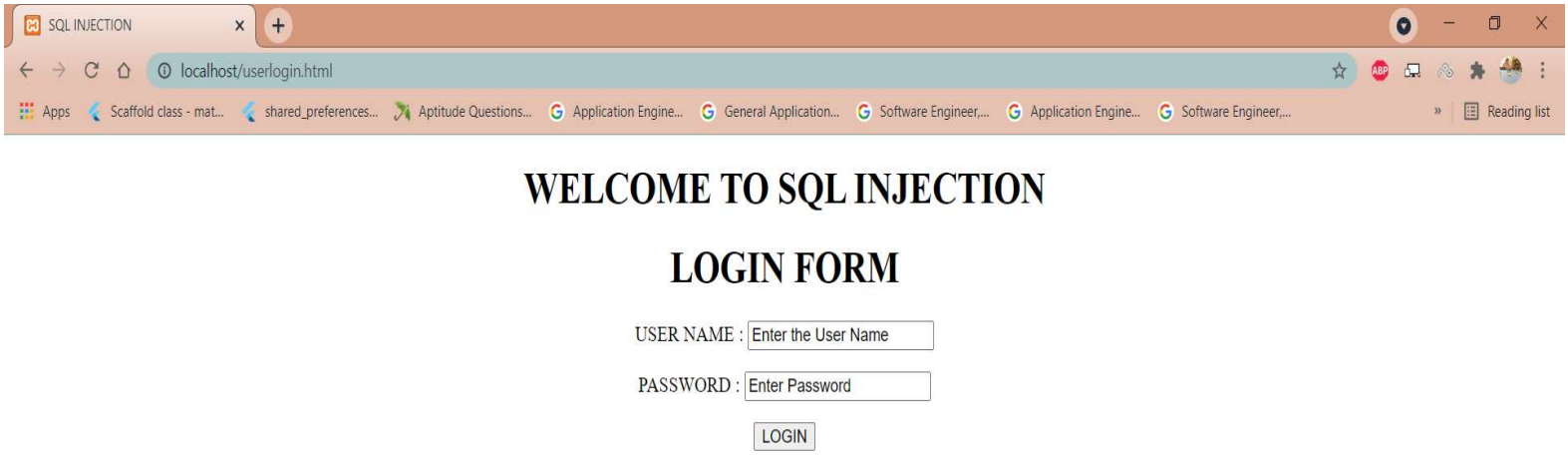
We Create a database “cns” and a table “login” with three columns “id”, “username” and “password”.



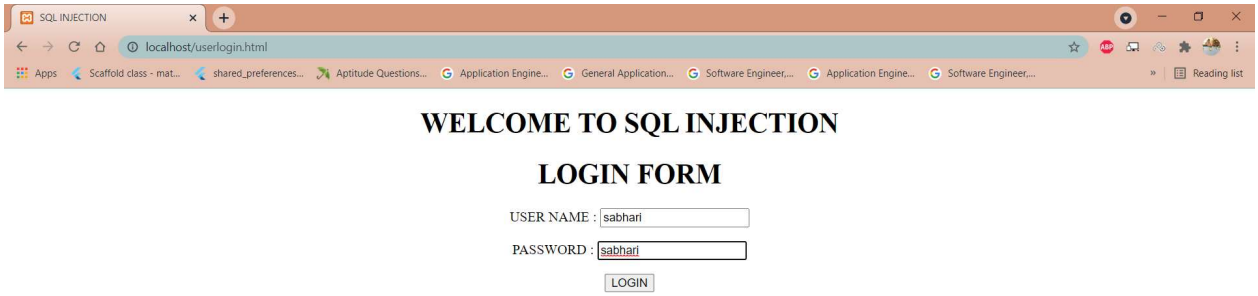
Thus, These three login credentials are valid and can be used to log in into the system.

In case of some other entry, the credentials are mismatched and the user isn't allowed to enter into the system.

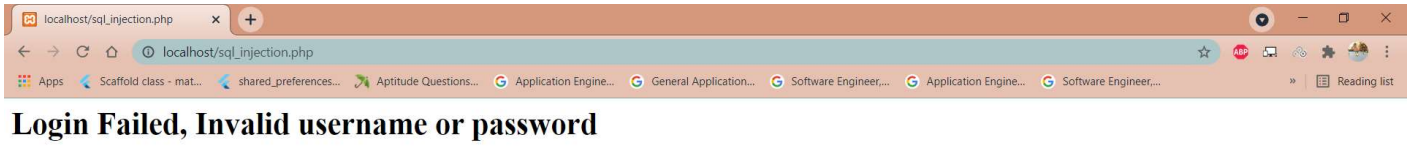
SQL INJECTION LOGIN FORM:



SUCCESSFUL LOGIN:



UNSUCCESSFUL LOGIN:



So, the unauthenticated user cannot access our webpage is clear. But, simply by injecting a slightly altered query we can access the webpage.

The attacker takes the advantage of poorly filtered or not correctly escaped characters embedded in SQL statements into parsing variable data from user input.

The attacker injects arbitrary data, most often a database query, into a string that's eventually executed by the database through a web application.

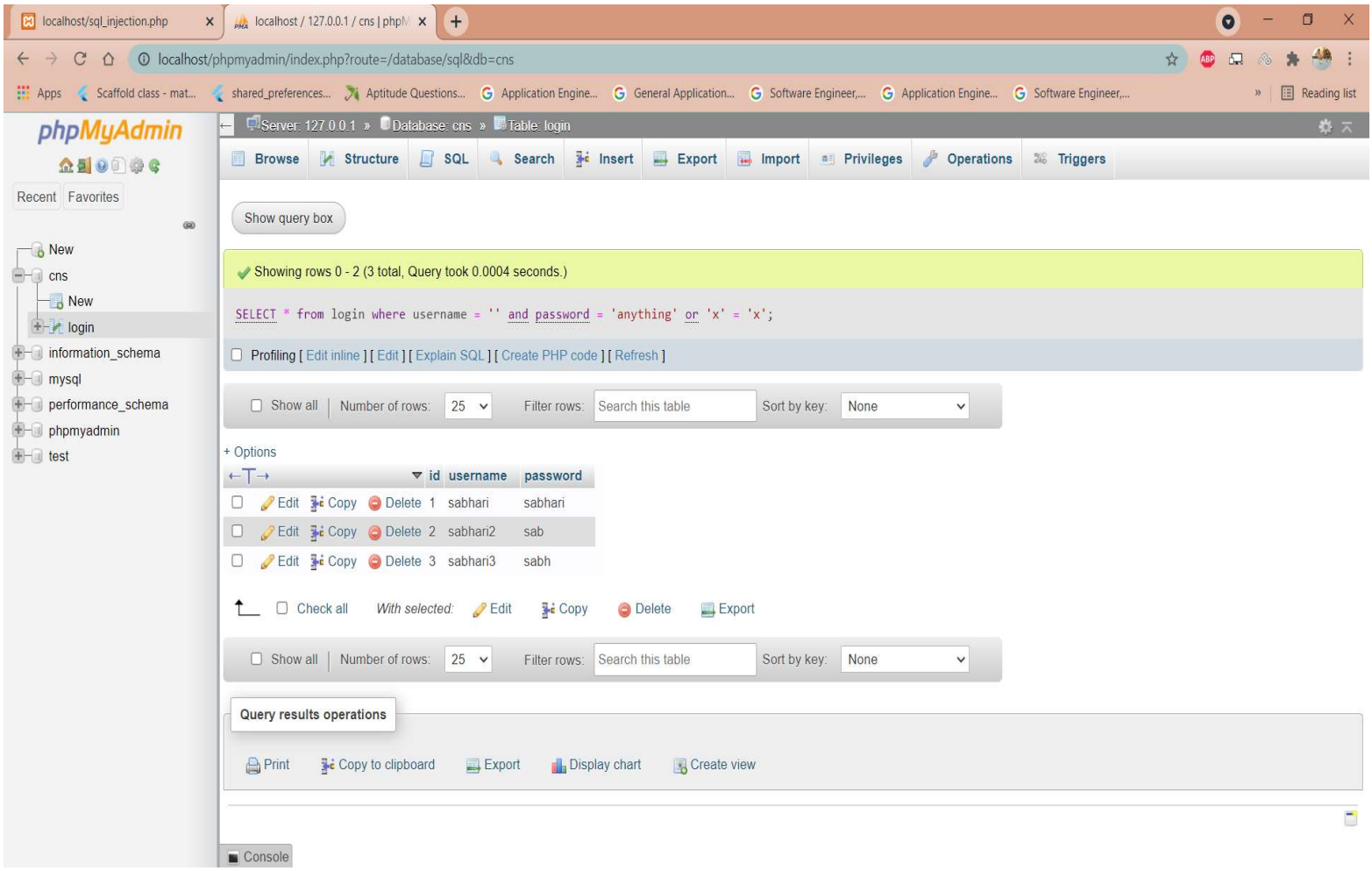
This is our query to authenticate user:

```
$sql = "select * from login where username = '$username' and password = '$password'";
$result = mysqli_query($con,$sql);
```

It contains 'and' statement saying that both the username and the password should be true to proceed the login.

The attacker tries to act smartly and inject a slightly different query in the user interface.

In the password field, if we try to add statement like anything' or 'x'='x the query will become like select * from login where username = ' ' and password = 'anything' or 'x'='x'.



So, this query displays the whole table as ‘x’ = ‘x’ turns ‘true’ and a single ‘true’ is sufficient enough for the ‘or’ condition to satisfy. The Attacker can play only with the user interface. Attackers don’t know what the database is and what the table is. But, the query is common.

Now, Lets try to access the system using “attacker1” username. This is a username not present in the database and when we try to enter we get a authentication error and our login attempt is restricted.

SQL INJECTION | localhost / 127.0.0.1 / cns | phpM

localhost/userlogin.html

Apps | Scaffold class - mat... | shared_preferences... | Aptitude Questions... | Application Engine... | General Application... | Software Engineer,... | Application Engine... | Software Engineer,... | Reading list

WELCOME TO SQL INJECTION

LOGIN FORM

USER NAME : malicious1

PASSWORD : kill

LOGIN

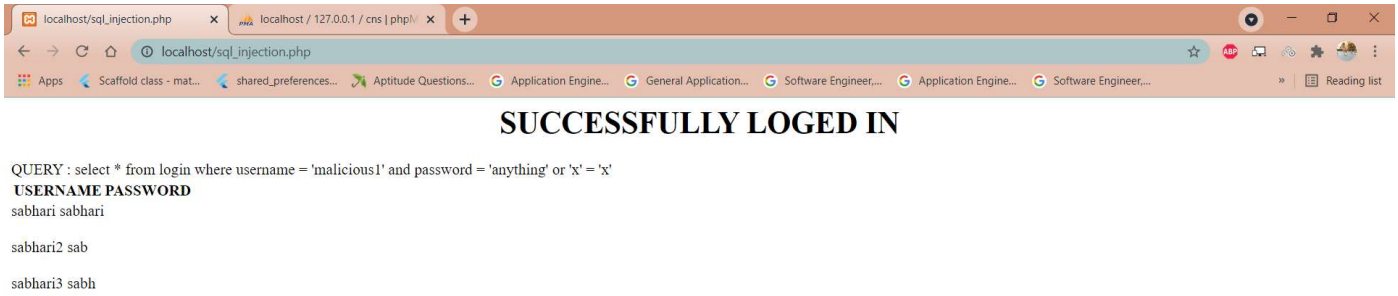
localhost/sql_injection.php

localhost/sql_injection.php

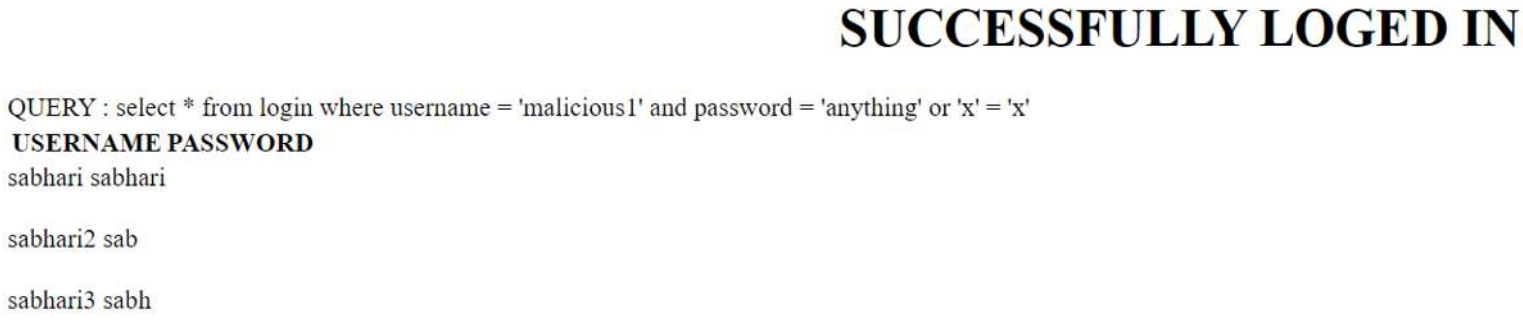
Apps | Scaffold class - mat... | shared_preferences... | Aptitude Questions... | Application Engine... | General Application... | Software Engineer,... | Application Engine... | Software Engineer,... | Reading list

Login Failed, Invalid username or password

But Now if the attacker uses sql injection then the user ‘malicious1’ which is not an authenticated user, will get access not only to the proceeding page but also get information about that database.



Thus the attacker has not only got the access to the proceeding page, but also the information about the database (username and password).



Hence SQL Injection has been done and attacker had gained access to the system and can cause potential harm.

SQL injection prevention techniques:

Developers can avoid vulnerabilities by applying the following main prevention methods.

Input validation:

The validation process is aimed at verifying whether or not the type of input submitted by a user is allowed. Input validation makes sure it is the accepted type, length, format, and so on. Only the value which passes the validation can be processed. It helps counteract any commands inserted in the input string.

Parameterized queries:

Parameterized queries are a means of pre-compiling an SQL statement so that
you can then supply the parameters in order for the statement to be executed. This method makes it possible for the database to recognize the code and distinguish it from input data. The user input is automatically quoted and the supplied input will not cause the change of the intent, so this coding style helps mitigate an SQL injection attack.

Stored procedures:

Stored procedures (SP) require the developer to group one or more SQL statements into a logical unit to create an execution plan. Subsequent executions allow statements to be automatically parameterized. Simply put, it is a type of code that can be stored for later and used many times. So, whenever you need to execute

the query, instead of writing it over and over, you can just call the stored procedure.

Escaping:

Always use character-escaping functions for user-supplied input provided by each database management system (DBMS). This is done to make sure the DBMS never confuses it with the SQL statement provided by the developer.

These are some of the techniques that can be used to prevent SQL Injection.!

THE RESPECTIVE CODES ARE SUBMITTED ALONG WITH THIS FILE.