AllenForum Security

AllenForum is a big web app running in the college which has thousands of records stored in its database. In all the data some information is sensitive like login credentials and some information can be public.

Hence to maintain the security of the system following are the things that are implemented to prevent several security attacks, brute force attacks, SQL Injection etc.

9.1 USE OF SESSION INSTEAD OF COOKIE

- Cookies are small files that are stored in the visitor's browser.
- Cookies can have a long lifespan, lasting months or even years.
- Cookies are limited in size depending on each browser's default settings.
- Cookies can be disabled if the visitor's browser does not allow them (uncommon).
- Cookies can be edited by the visitor. (Do not use cookies to store sensitive data.)

- Sessions are small files that are stored on the website's server.
- Sessions have a limited lifespan; they expire when the browser is closed.
- Sessions are only limited in size if you limit their size on the server.
- Sessions cannot be disabled by the visitor because they are not stored in the browser.
- Sessions cannot be edited by the visitor.

In short, cookies serve as a temporary or long-term storage unit on the visitor's computer that should not contain sensitive information, and sessions serve as a temporary storage unit not on the visitor's computer that can hide sensitive information. For most tasks I find it efficient to use sessions and cookies together.

9.2 PRVENTION FROM SQL INJECTION

SQL Injection is an attack that poisons dynamic SQL statements to comment out certain parts of the statement or appending a condition that will always be true. It takes advantage of the design flaws in poorly designed web applications to exploit SQL statements to execute malicious SQL code.

Hacking Activity: SQL Inject a Web Application

We have a simple web application at http://www.techpanda.org/ **that is vulnerable to SQL Injection attacks for demonstration purposes only.** The HTML form code above is taken from the login page. The application provides basic security such as sanitizing the email field. This means our above code cannot be used to bypass the login.

To get round that, we can instead exploit the password field. The diagram below shows the steps that you must follow

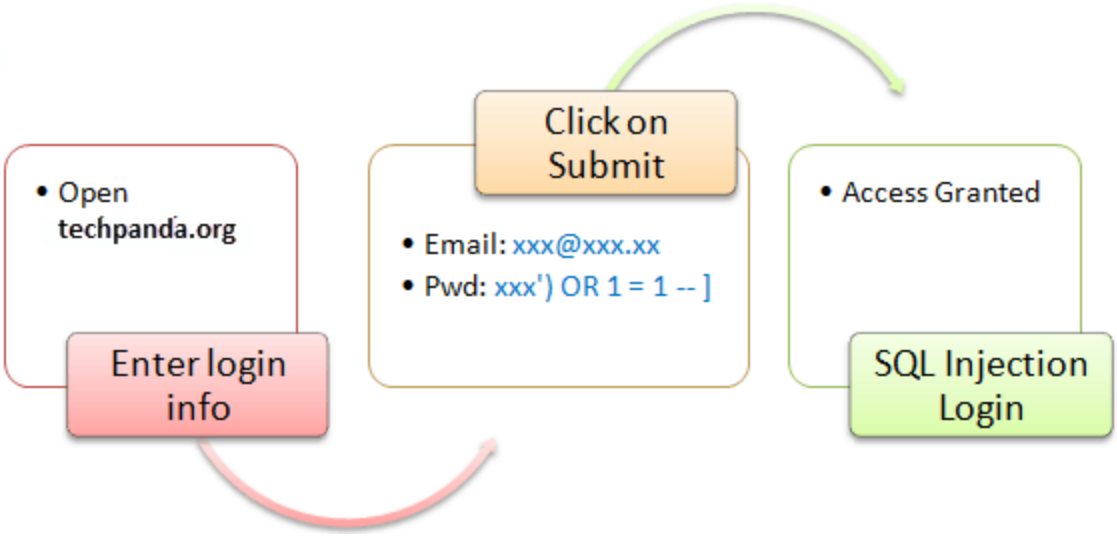


Figure 1: SQL injection

Let's suppose an attacker provides the following input

- Step 1: Enter xxx@xxx.xxx as the email address
- Step 2: Enter xxx') OR 1 = 1 --]

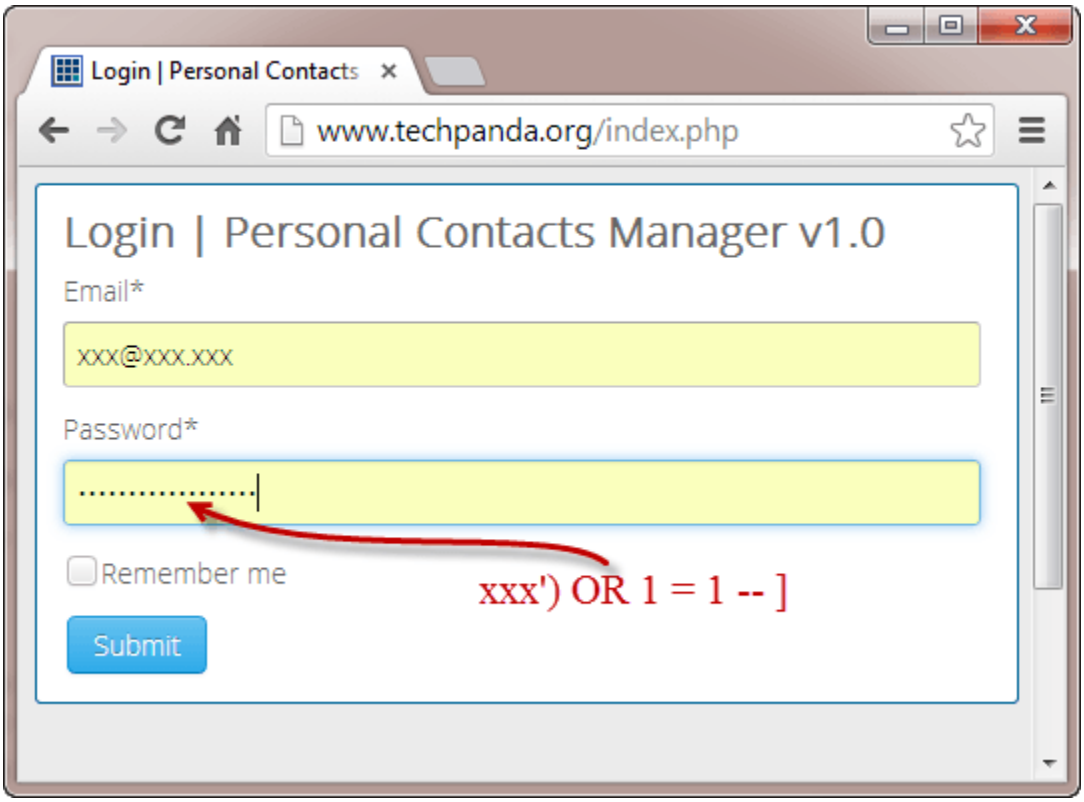


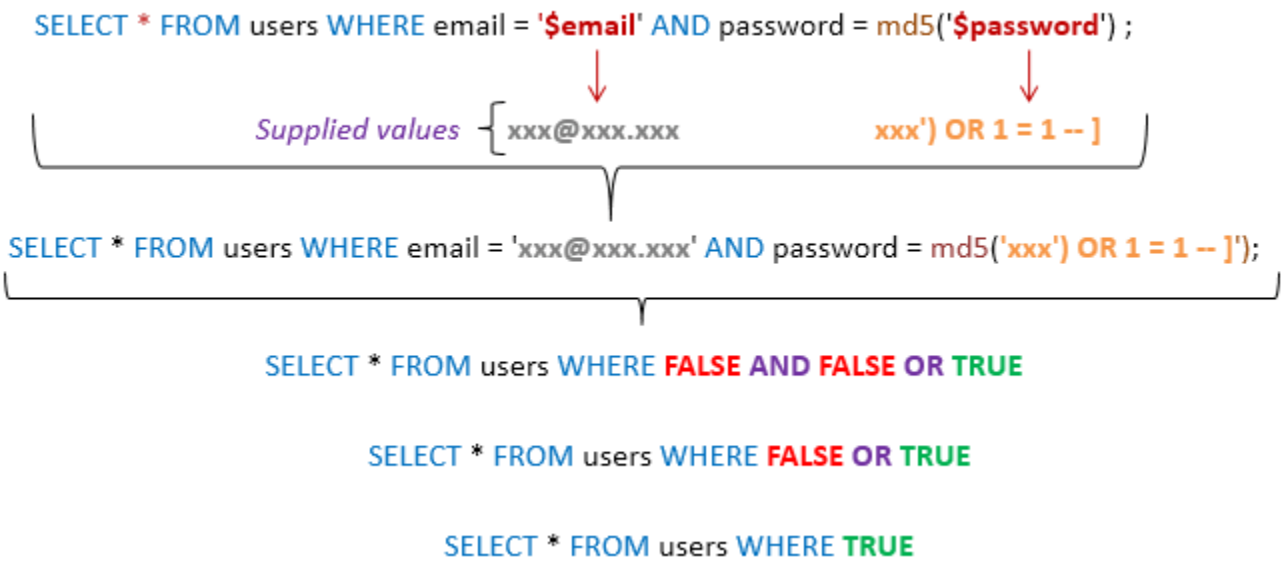
Figure 2: screenshot from techpanda.org

- Click on Submit button
- You will be directed to the dashboard

The generated SQL statement will be as follows

SELECT * FROM users WHERE email = 'xxx@xxx.xxx' AND password = md5('xxx') OR 1 = 1 --]');

The diagram below illustrates the statement has been generated.



HERE,

- The statement intelligently assumes md5 encryption is used
- Completes the single quote and closing bracket
- Appends a condition to the statement that will always be true

In general, a successful SQL Injection attack attempts a number of different techniques such as the ones demonstrated above to carry out a successful attack. [x]

9.2 USE OF PREPARED STATEMENT & SPECIAL FUNCTIONS

The prepared statement execution consists of two stages: prepare and execute. At the prepare stage a statement template is sent to the database server. The server performs a syntax check and initializes server internal resources for later use.

As the above security attack called SQL injection, another technique is used called prepared statements and some special functions

```
if (isset($_POST['q']) == 'login'){
    $userEmail = trim($connection->real_escape_string($_POST['useremail']));
    $userPass = sha1(trim($connection->real_escape_string($_POST['userpassword'])));

    $login = $connection->query("select * from forum_users
      where user_email = '$userEmail' AND user_pass = '$userPass' ")
    or die("Login failed".$connection->error);
        if ($login->num_rows > 0){
// success, proceed
        }

}
```

Figure 3: code snippet for login

9.3 FILE UPLOAD VALIDATION

All the file upload fields are properly validated accepting only specified file extension like image can be only .png, .jpg or jpeg and some .pdf and doc or .docx. because this is also a security issue if this is not properly done the attackers can upload some malicious code to the server.

Following is the example.

```
function validateAdminProfile() {
    var allowedFiles = [".png", ".jpeg", ".jpg"];
    var fileUpload = document.getElementById("admin_profile_pic");
    var regex = new RegExp("([a-zA-Z0-9\s_\\.\-:])+(" + allowedFiles.join('|') +")$");
    if (!regex.test(fileUpload.value.toLowerCase())) {
        alert("Please Upload Valid File");
        document.getElementById("admin_profile_pic").value = "";
        return false;
    }else {
        return true;
    }
}
```

Figure 4: code snippet for file upload validation

9.4 SECURITY ALERT EMAIL & SECURITY CODE

In the current system of *AllenForum* this functionality has been implemented and working properly. This is done to alert the user if somebody tries to access other users' account. And the same email will be sent to admin whenever any user fails to login in 3 consecutive attempts.

In future if this is not much capable then we can make it **2 step login process** means

First user will enter his login details; if the details are correct then a security code like OTP will be sent to the users email or contact. After that user will be asked to enter security code if code is matched the user will be logged into system.

The diagrammatic representation of 2 step login process is shown in the flowchart given below in figure -x

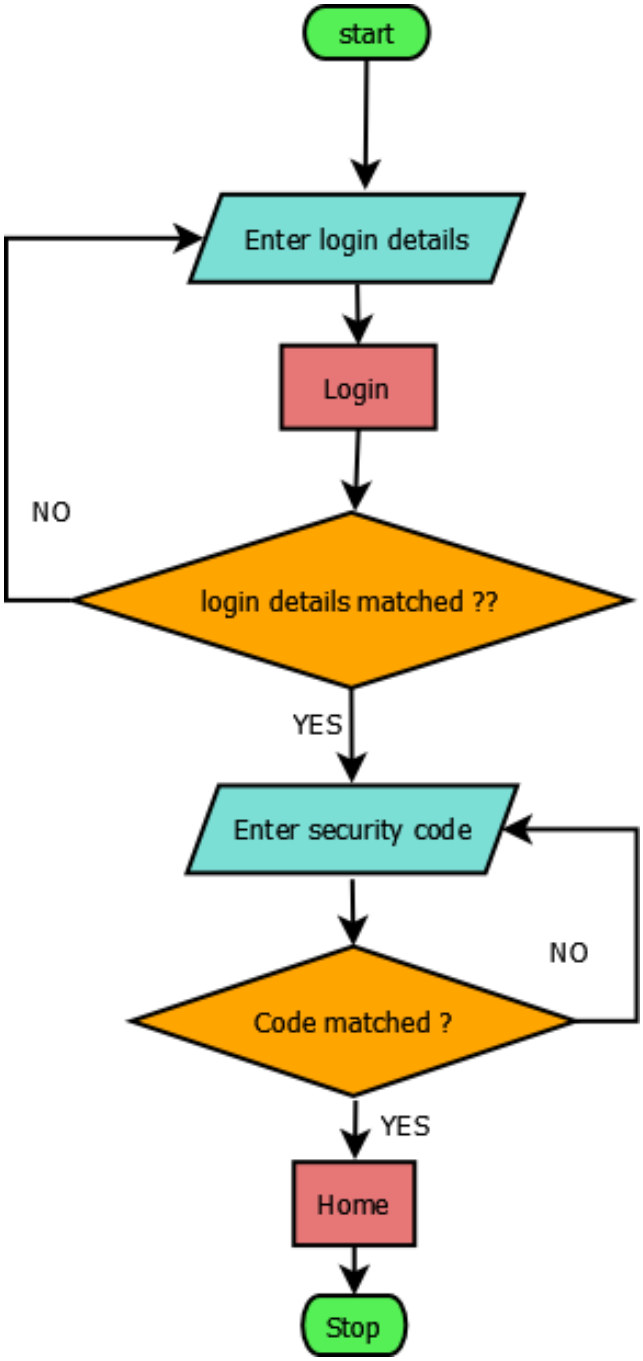


Figure 5: 2 Step Login Process

To enhance more security we can also keep the **IP address** of that user who is trying to login or doing activity after login in the system. This step can be done in future if needed.