

WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student Name: Charan Sai Venaganti

Email: venagaci@mail.uc.edu



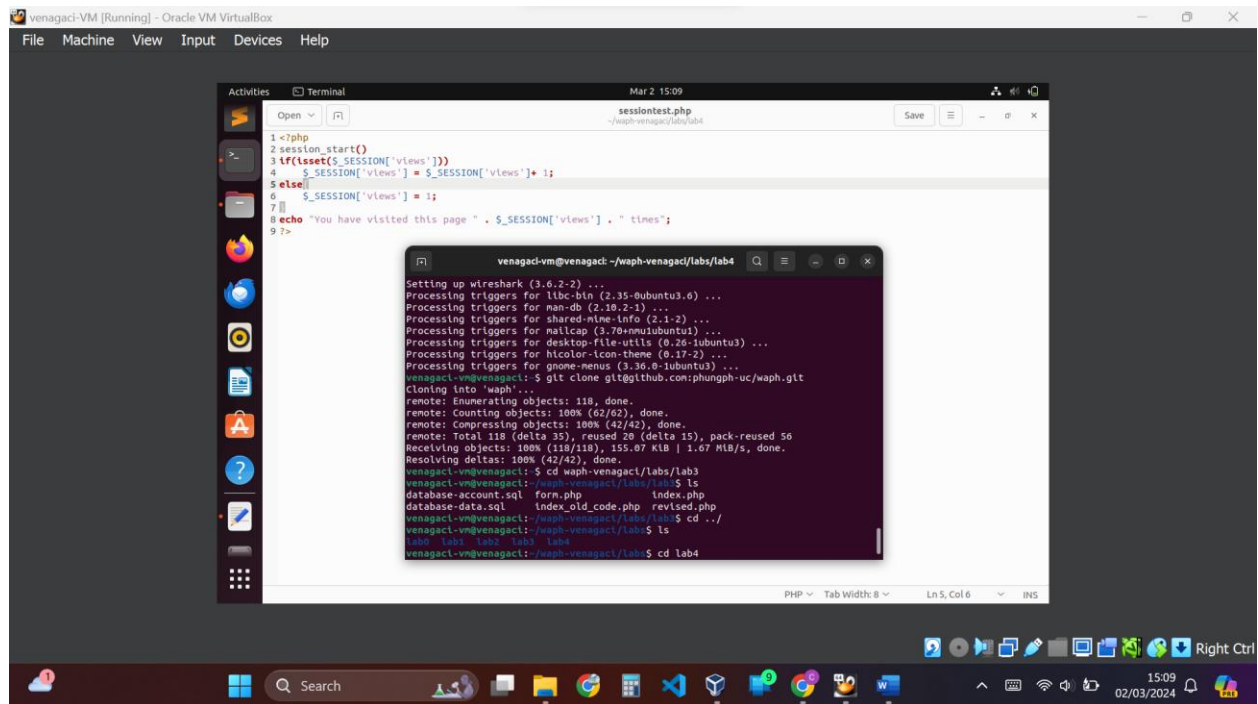
Repository URL: (<https://github.com/venagaci/waph-venagaci.git>)

Overview:

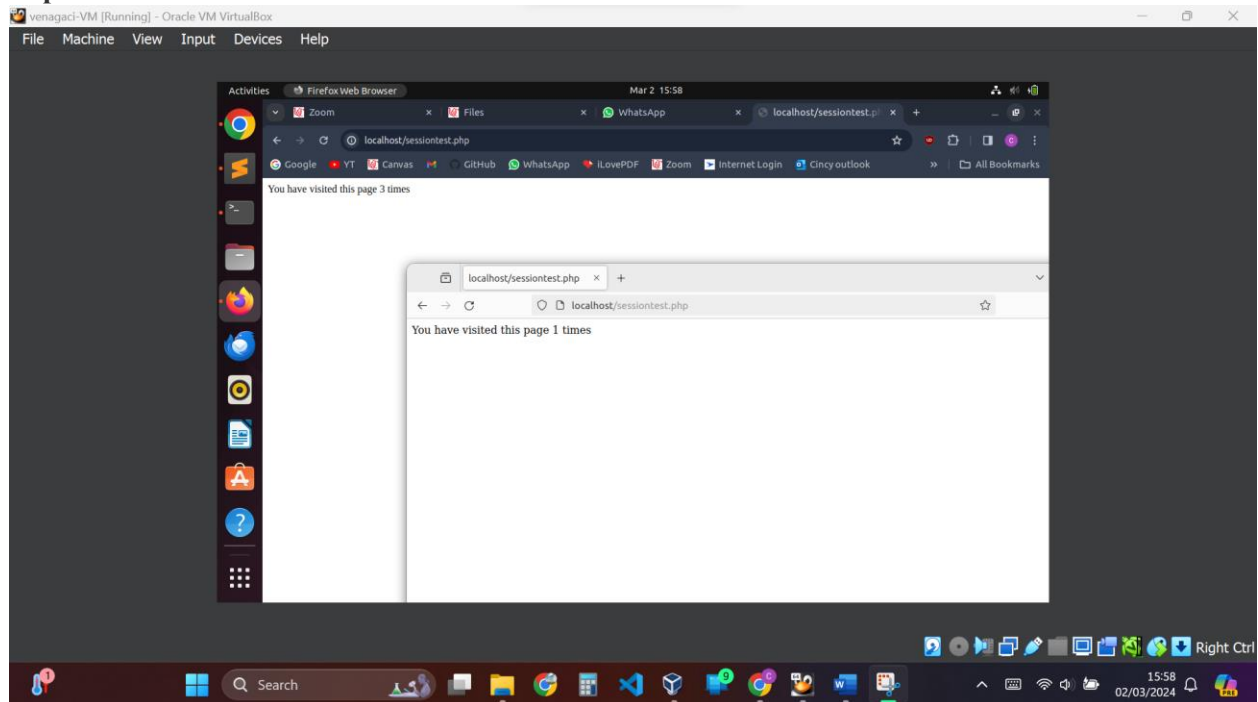
In this lab4, I gained practical experience in understanding, implementing, and securing session management in PHP web applications. The tasks encompassed deploying session management, observing session handling processes using Wireshark, identifying and mitigating session hijacking attacks, and applying secure session authentication measures. Through deploying sessiontest.php, observing session handshaking using Wireshark, and performing session hijacking attacks, I learned about session management vulnerabilities and the importance of implementing countermeasures. Additionally, revising the login system to incorporate session management, simulating session hijacking attacks, and implementing HTTPS setup with SSL certificates provided insights into securing session authentication. By setting HttpOnly and Secure flags for session cookies and implementing defense-in-depth measures to detect session hijacking, I enhanced the security of the web application. Overall, this lab provided valuable hands-on experience in securing session management and authentication in PHP web applications.

1.a. Deploy and test sessiontest.php

Sub-task:



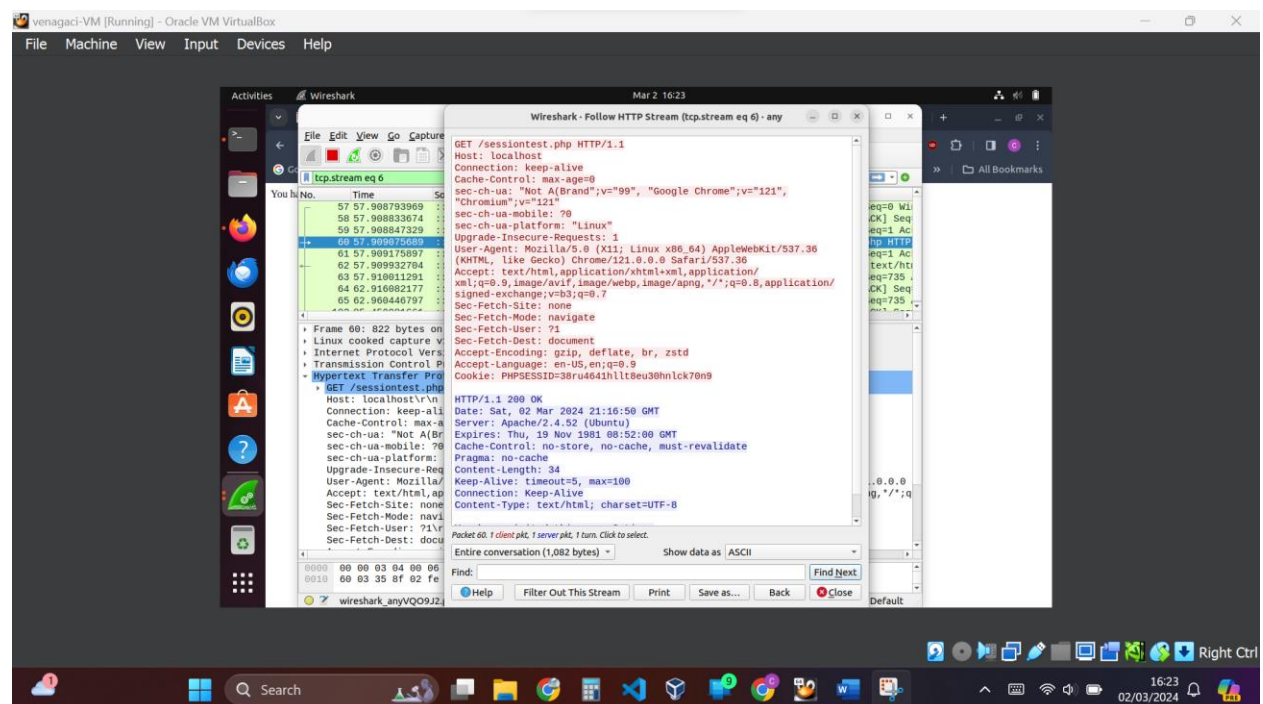
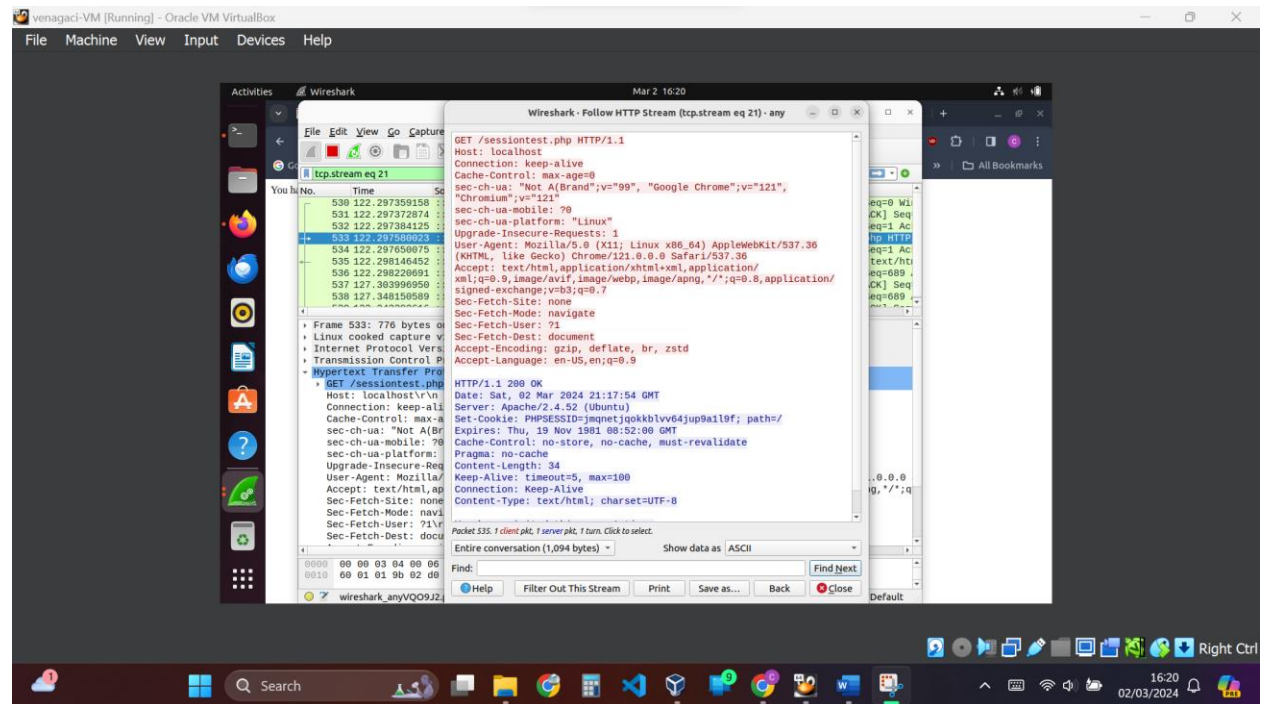
Expected Demonstration:



Summary: I cloned the course repository, revised and deployed `sessiontest.php` to my web server, and accessed it through different web browsers. This helped me understand how session values are managed and maintained across different sessions.

1.b. Observe the Session-Handshaking using Wireshark

Expected Demonstration:

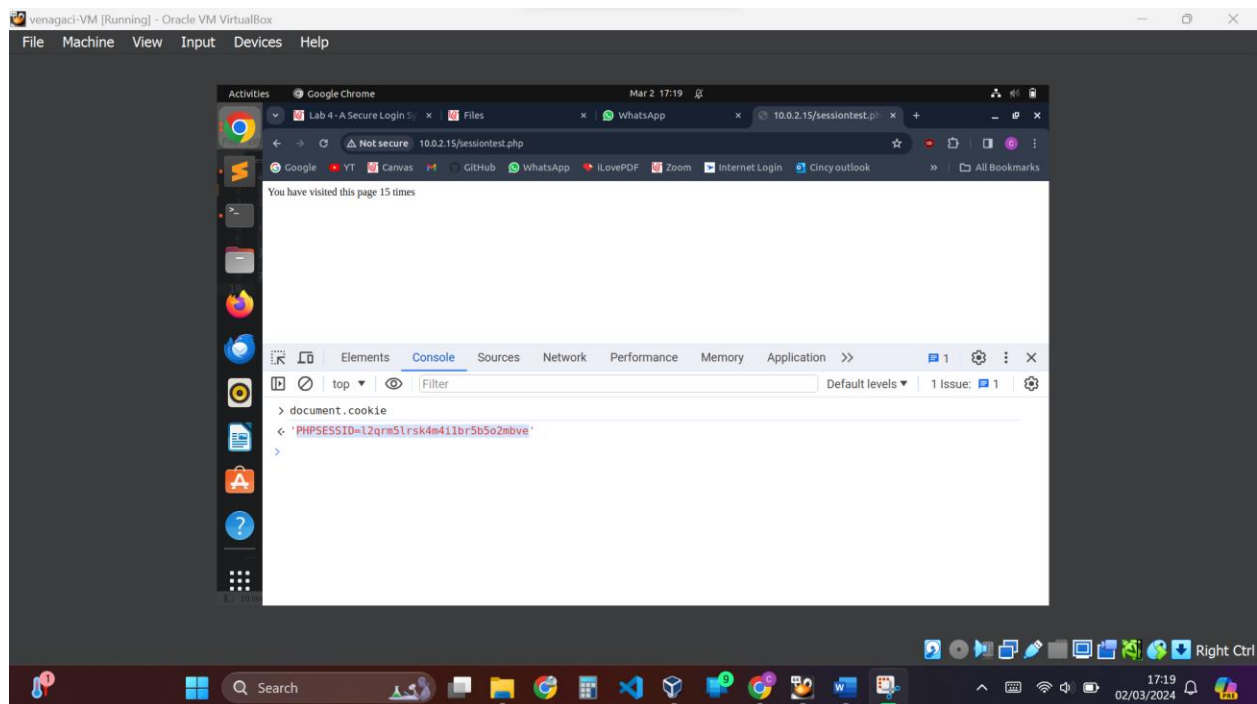


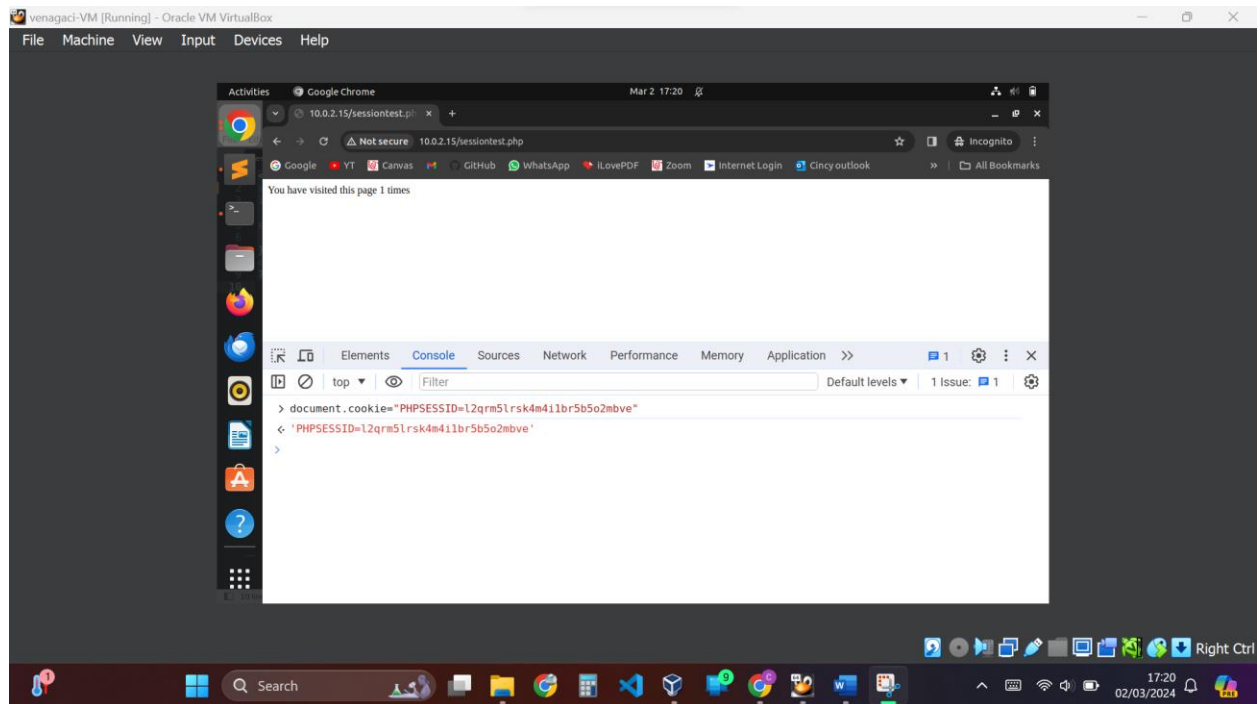
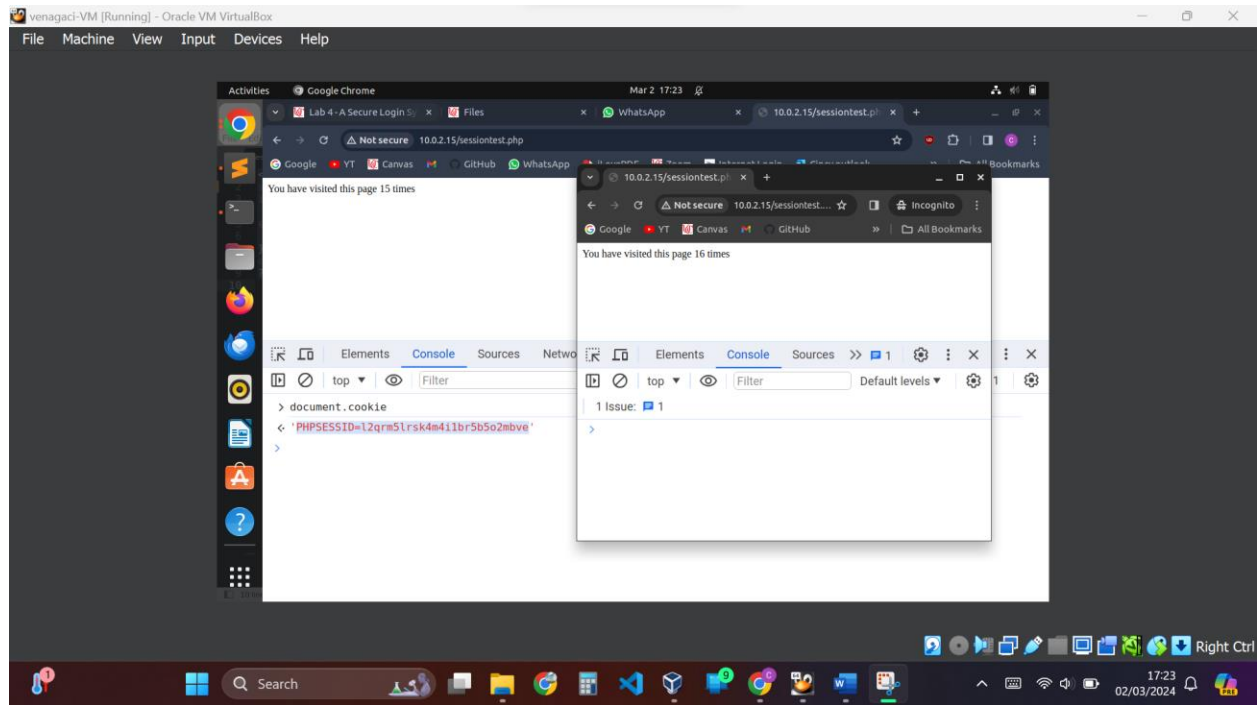
Summary: I used Wireshark to capture the traffic while accessing sessiontest.php. By analyzing the HTTP requests and responses related to session handling, I gained insights into the session handshaking process and how session cookies are exchanged between the client and server.

Discuss your understanding of this session handshaking process.

The process implements a session tracking mechanism wherein, upon the initial request to sessiontest.php, the server detects the absence of a session cookie and creates a new session, assigning a unique session identifier (PHPSESSID) and sending it back to the client via the Set-Cookie header. Subsequent requests from the client include this session cookie, allowing the server to identify and associate the request with the corresponding session data. The server updates the session data, in this case, incrementing a visit counter, and returns the updated information in the response. This process continues for each subsequent request, facilitating stateful communication between the client and server and enabling the server to maintain session-specific data across multiple interactions.

1.c. Understanding Session Hijacking





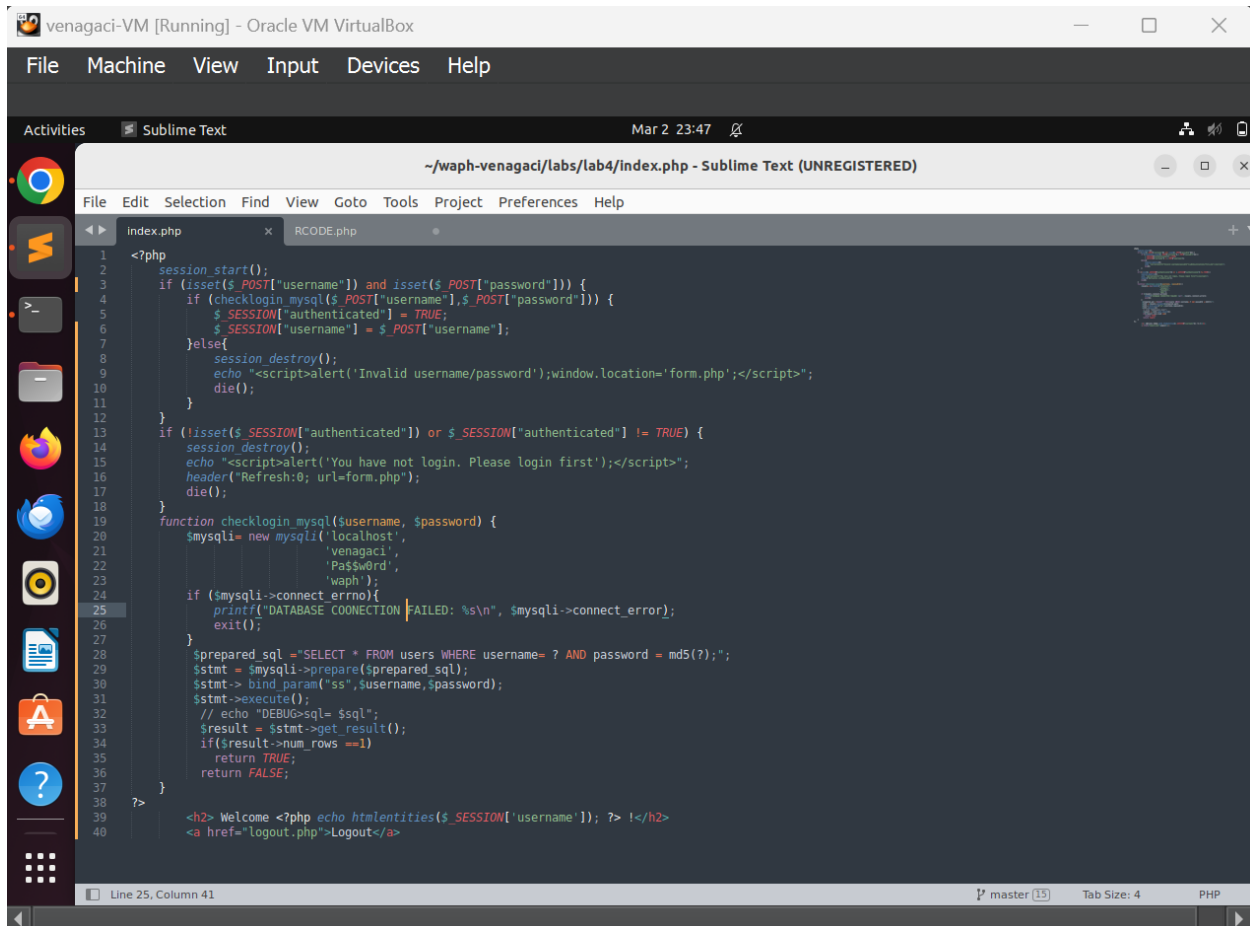
Summary: I performed a session hijacking attack by copying and pasting the session ID from one to another browser which made that particular session hijacked. This practical exercise helped me understand the vulnerabilities associated with session management and the importance of implementing countermeasures to prevent such attacks.

Task 2:

2.a. Revised Login System with Session Management

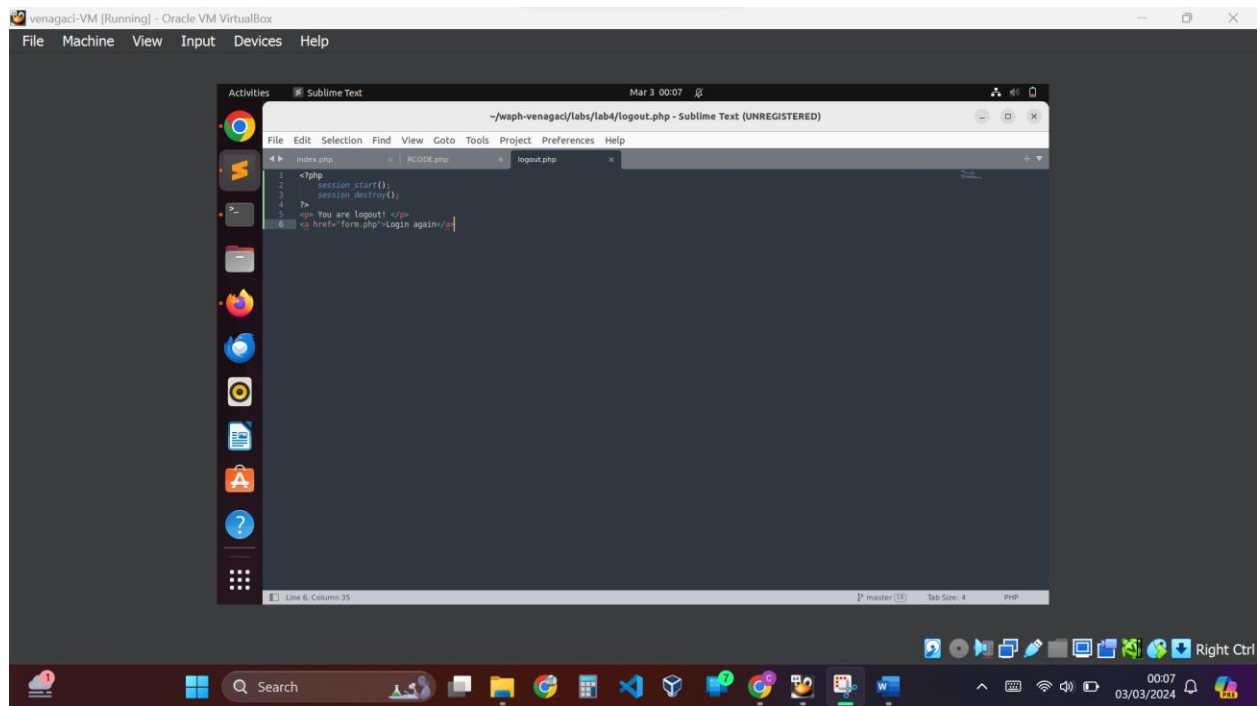
Sub-task:

Revised index.php

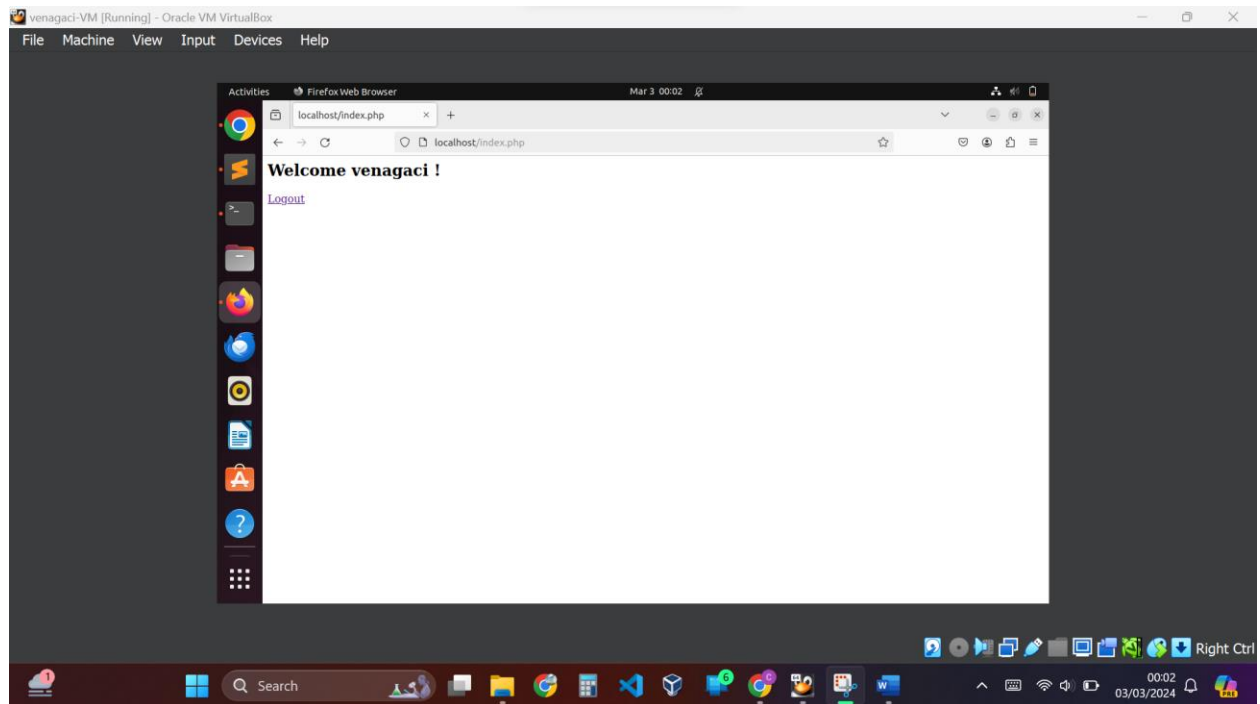


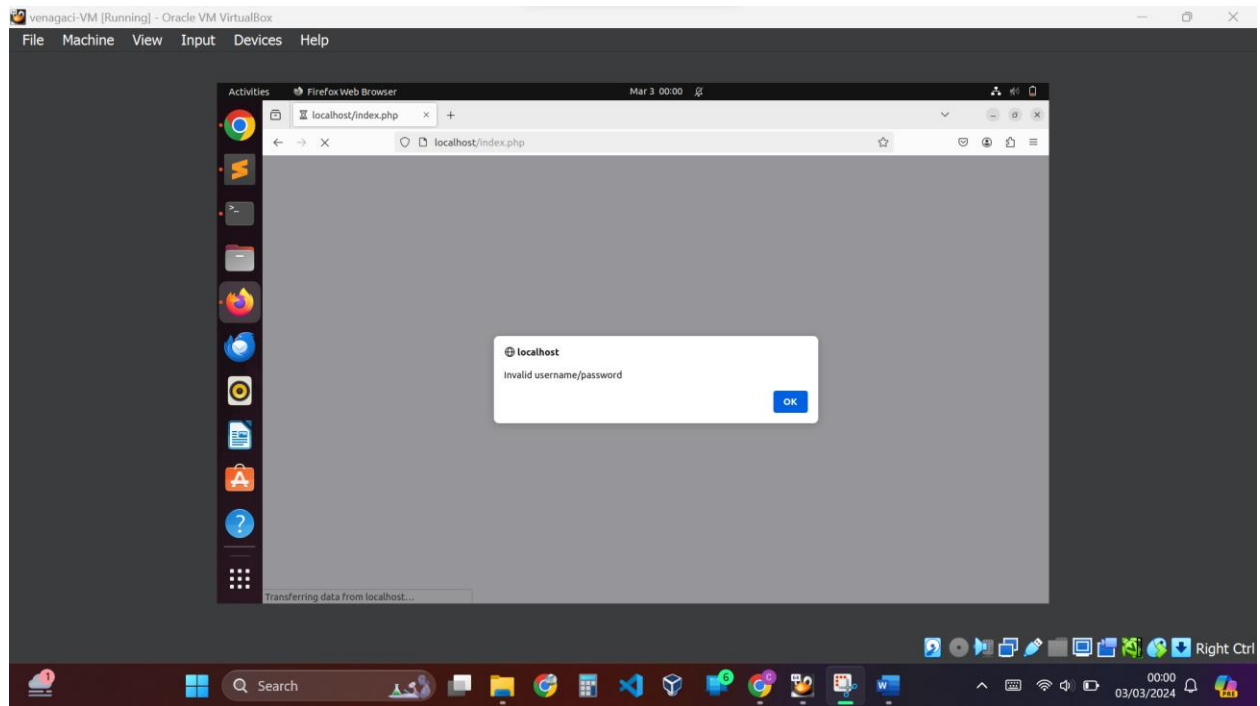
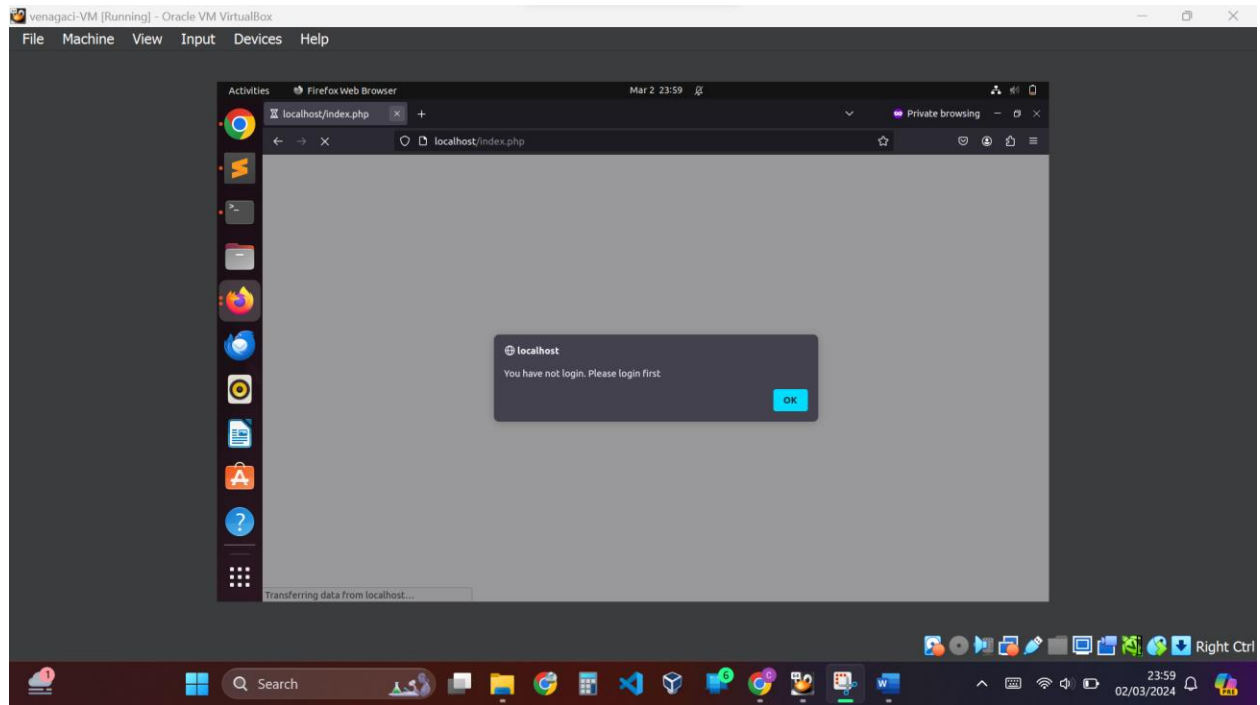
```
1 <?php
2 session_start();
3 if (isset($_POST['username']) and isset($_POST['password'])) {
4     if (checklogin_mysql($_POST['username'],$_POST['password'])) {
5         $_SESSION['authenticated'] = TRUE;
6         $_SESSION['username'] = $_POST['username'];
7     }else{
8         session_destroy();
9         echo "<script>alert('Invalid username/password');window.location='form.php';</script>";
10        die();
11    }
12 }
13 if (!isset($_SESSION['authenticated']) or $_SESSION['authenticated'] != TRUE) {
14     session_destroy();
15     echo "<script>alert('You have not login. Please login first');</script>";
16     header("Refresh:0; url=form.php");
17     die();
18 }
19 function checklogin_mysql($username, $password) {
20     $mysqli= new mysqli('localhost',
21                        'venagaci',
22                        'Passw0rd',
23                        'waph');
24     if ($mysqli->connect_errno){
25         printf("DATABASE CONNECTION FAILED: %s\n", $mysqli->connect_error);
26         exit();
27     }
28     $prepared_sql ="SELECT * FROM users WHERE username= ? AND password = md5(?)";
29     $stmt = $mysqli->prepare($prepared_sql);
30     $stmt-> bind_param("ss",$username,$password);
31     $stmt->execute();
32     // echo "DEBUG>sql= $sql";
33     $result = $stmt->get_result();
34     if($result->num_rows ==1)
35         return TRUE;
36     return FALSE;
37 }
38 ?>
39 <h2> Welcome <?php echo htmlentities($_SESSION['username']); ?> !</h2>
40 <a href="logout.php">Logout</a>
```

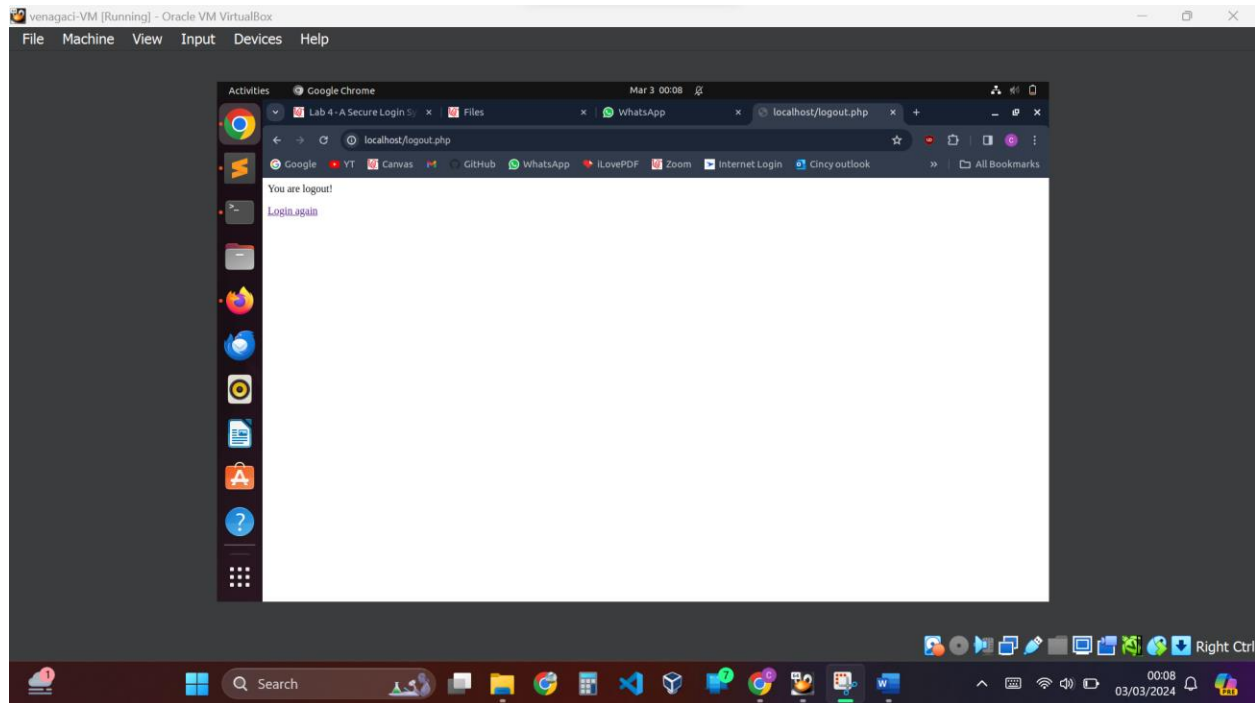

Logout.php



Expected Demonstration:



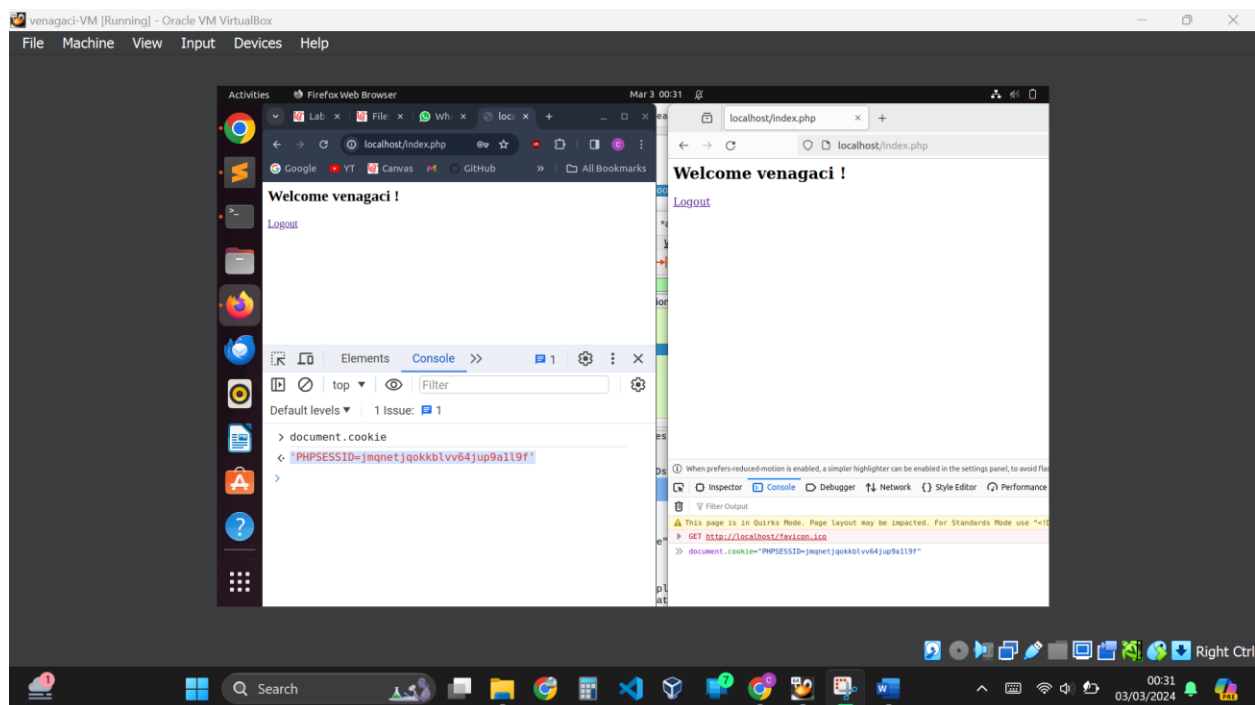
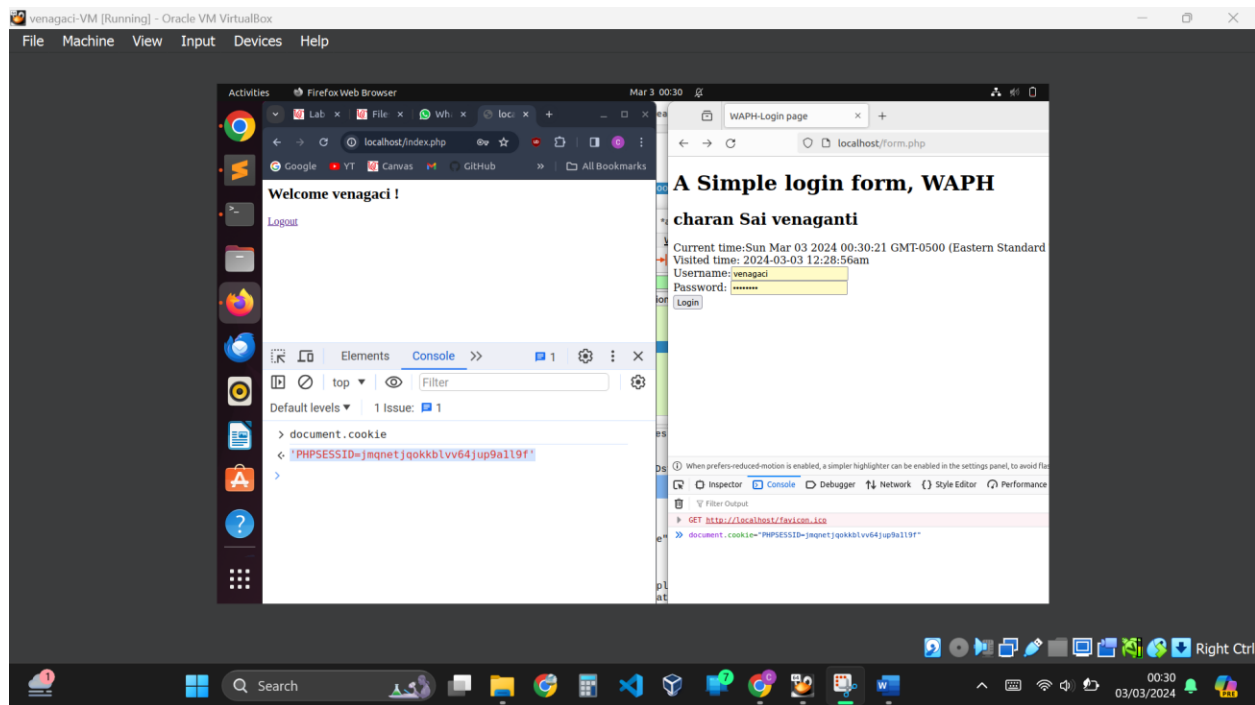




Summary: I revised the `index.php` to implement session management, ensuring that authenticated users are allowed access to the protected pages while unauthenticated users are redirected or alerted accordingly. This task reinforced the importance of securely managing user sessions to maintain the integrity of the authentication process.

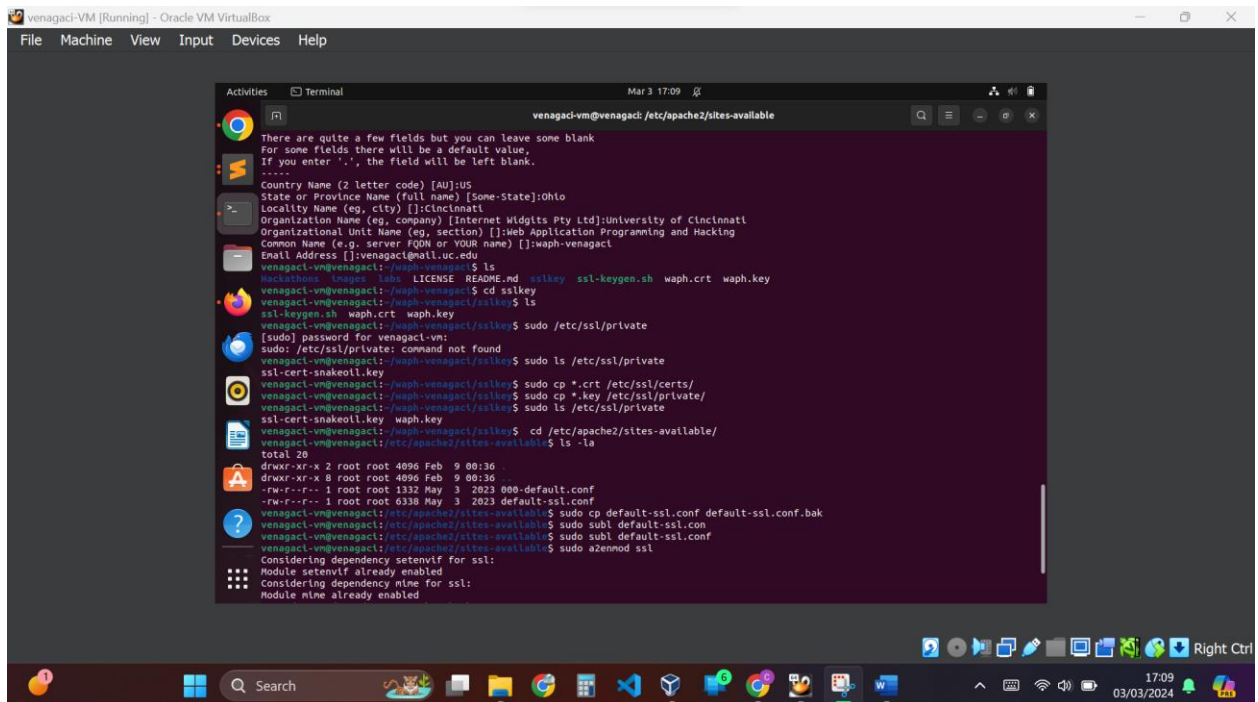
2.b. Session Hijacking Attacks

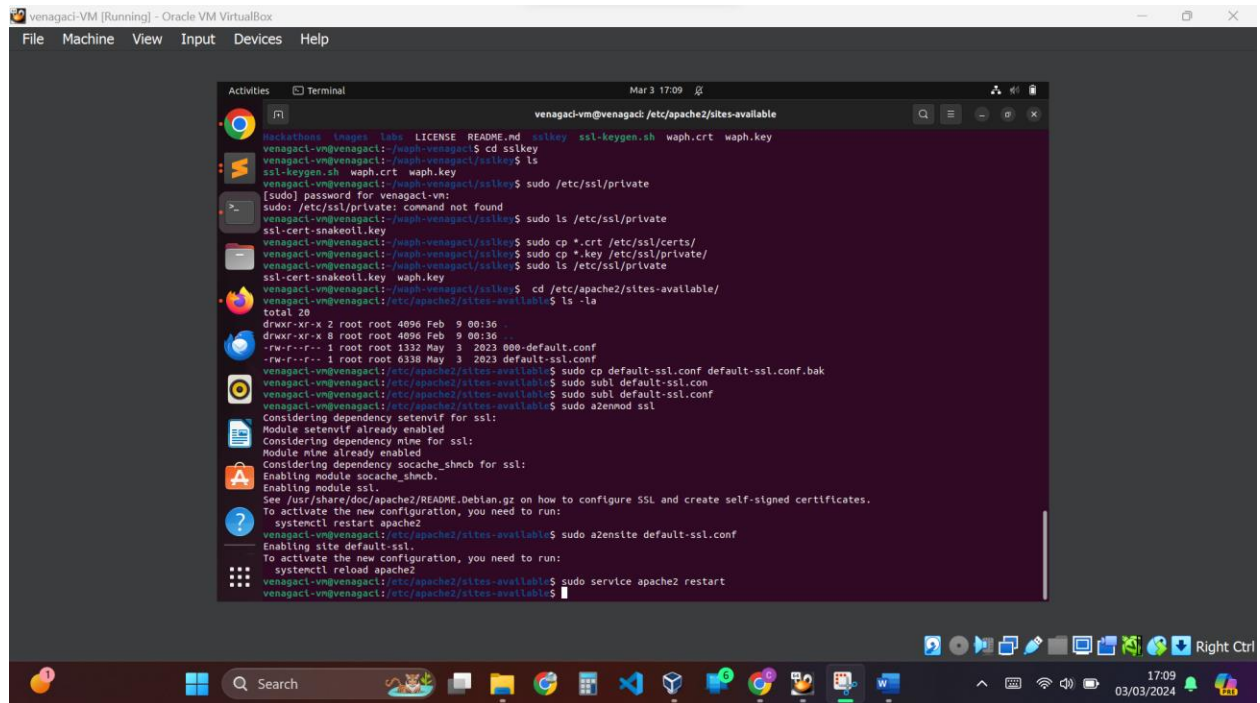
Sub-task & Expected Demonstration:



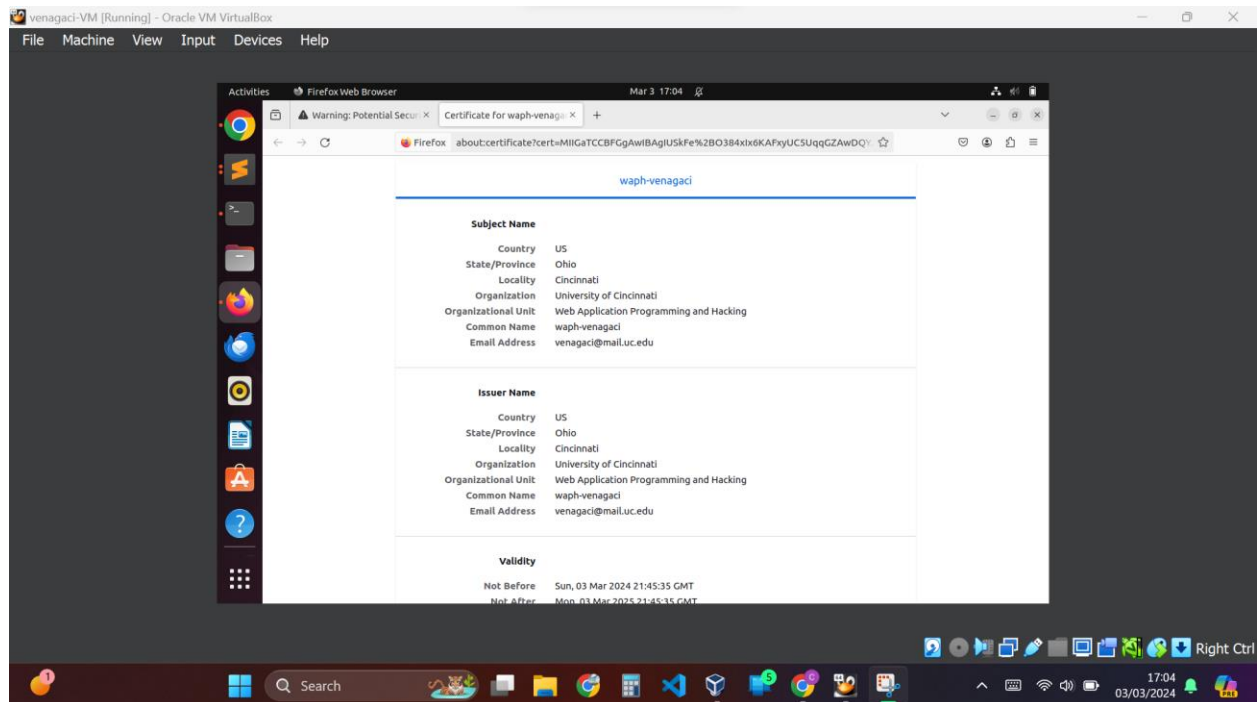
Summary: I simulated session hijacking attacks by manually copying session IDs between different browsers. By demonstrating access to session-protected pages through session hijacking, I gained a deeper understanding of the risks associated with insecure session management.

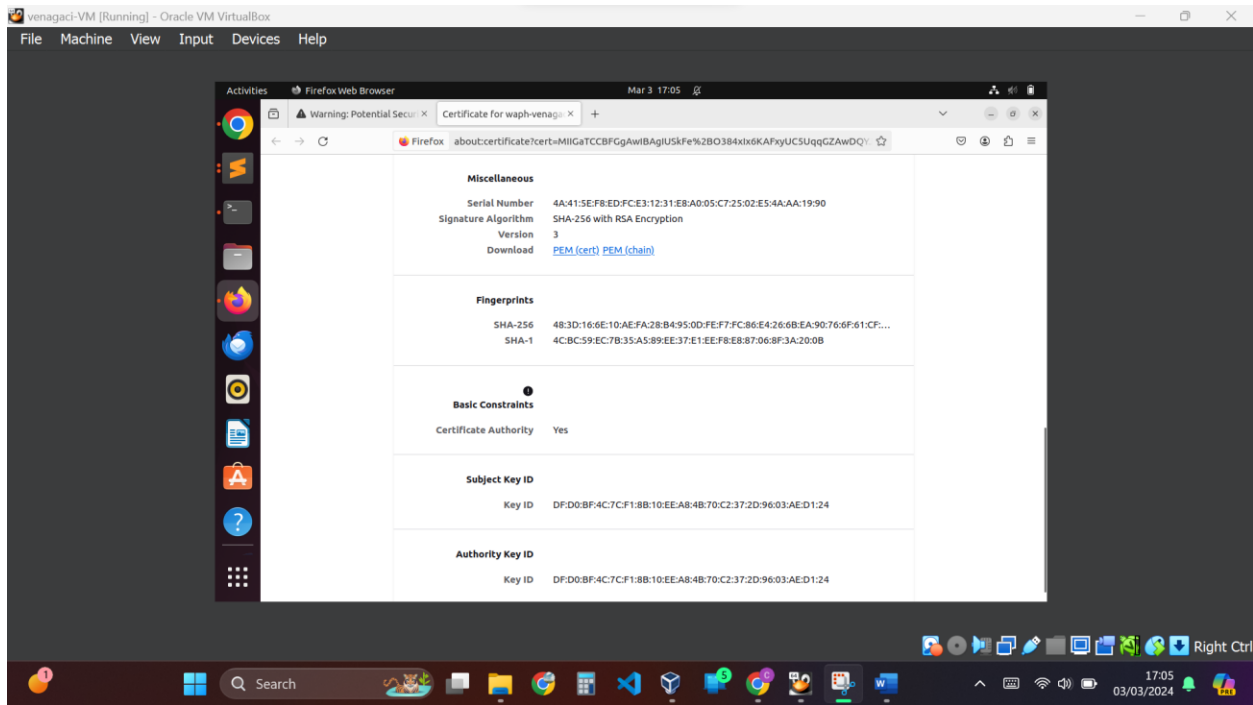
Sub-task:





Expected Demonstration:

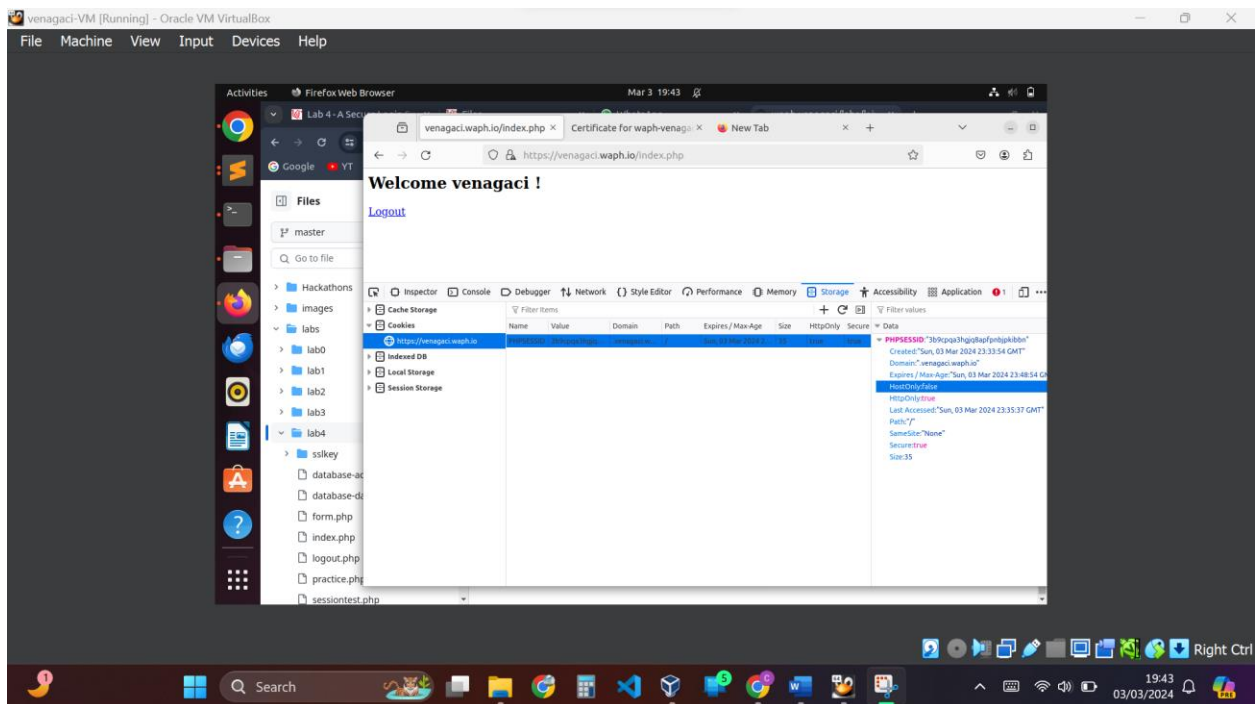
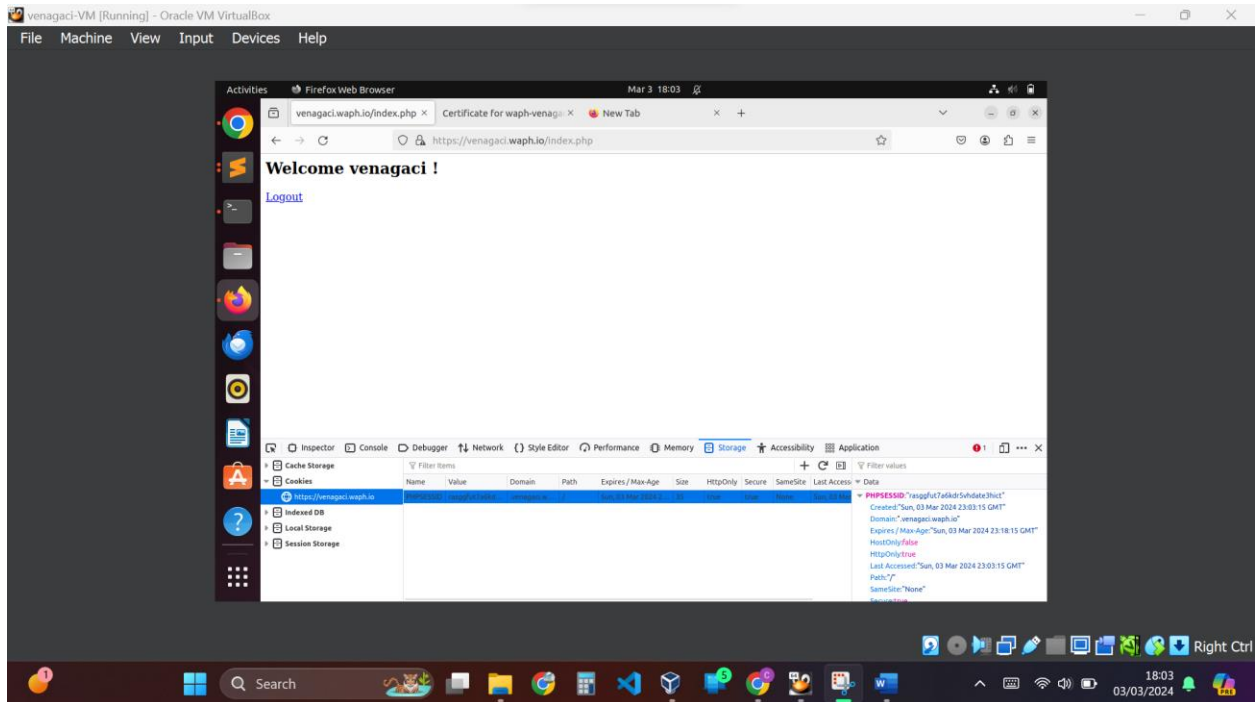


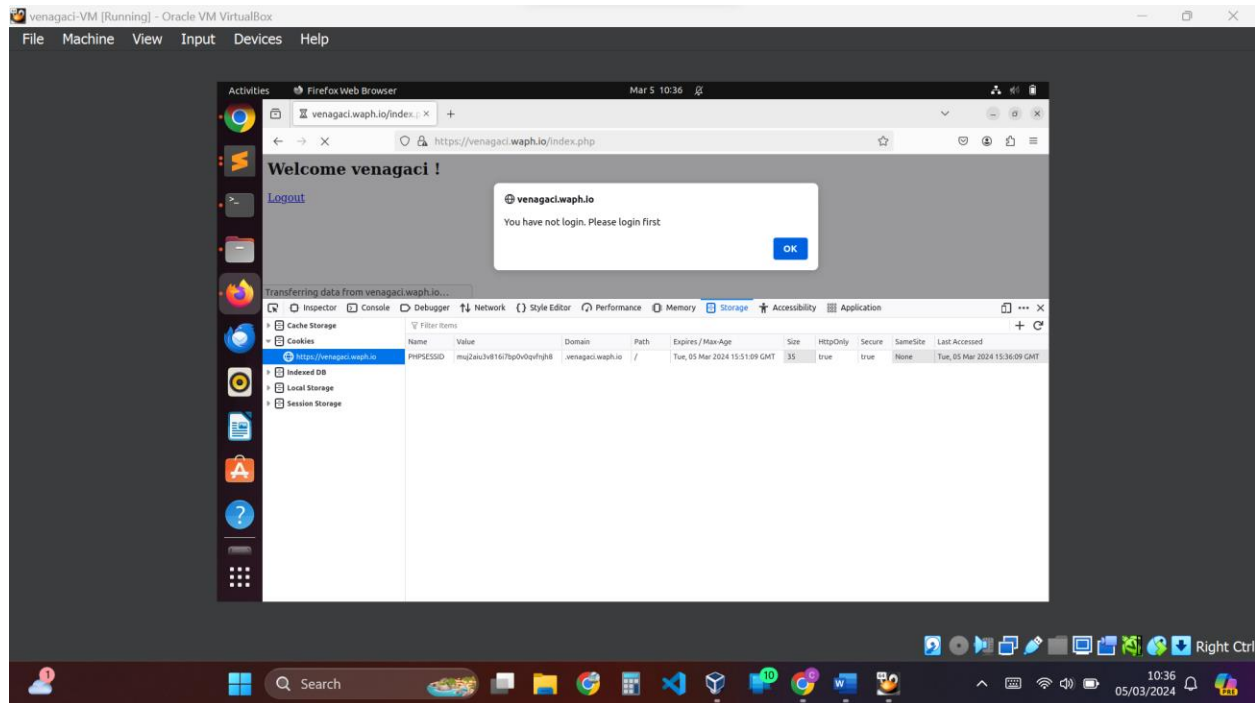


Summary: *I generated SSL certificates and configured my web server to use HTTPS, thereby encrypting the data exchanged between the client and server. This step enhanced the security of the web application by preventing eavesdropping and man-in-the-middle attacks.*

3.b. Securing Session Against Session Hijacking Attacks - setting HttpOnly and Secure flags for cookies

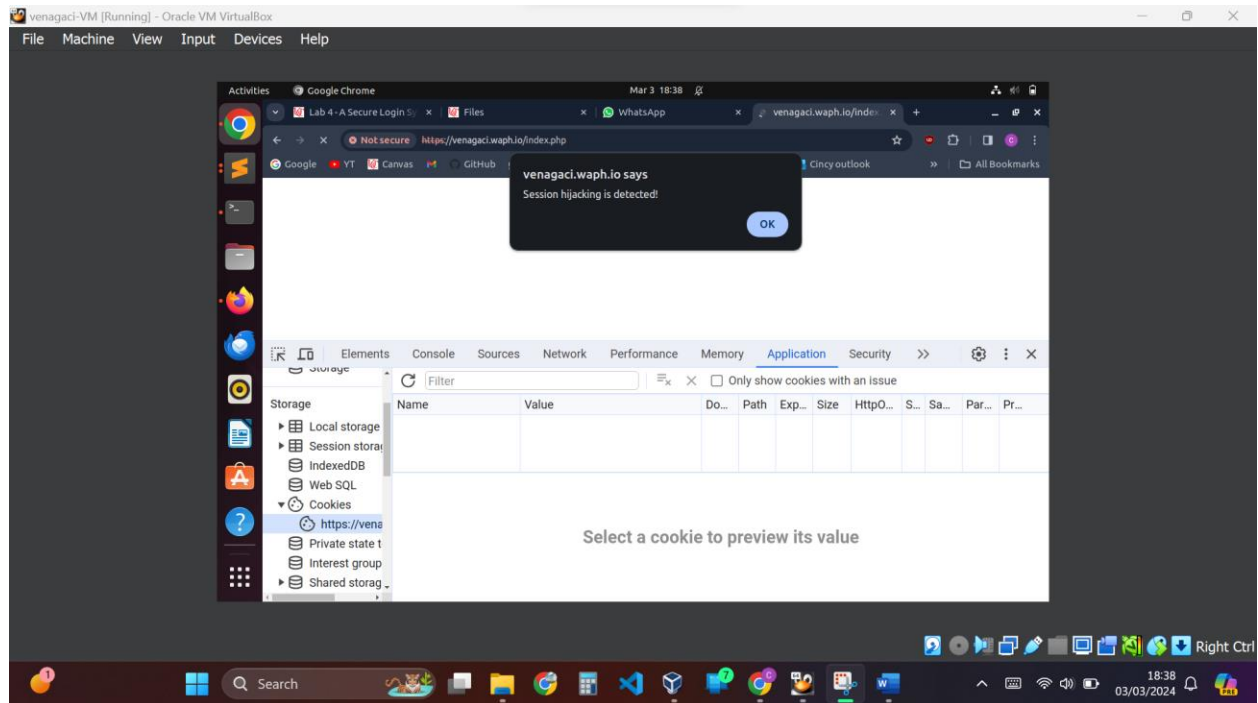
Expected Demonstration:





Summary: I implemented measures to secure sessions by setting the HttpOnly and Secure flags for session cookies. This prevented client-side scripts from accessing session cookies and ensured that they are only transmitted over secure connections, mitigating the risk of session hijacking attacks

3.c. Securing Session Against Session Hijacking Attacks - Defense In-Depth



Summary: I revised the `index.php` to store additional session variables containing browser information. By comparing this information with the session data, I could detect and prevent session hijacking attacks, thereby adding an extra layer of defense to the authentication process.

Files in Repository:

