Cross-Site Request Forgery (CSRF) Protection

Overview

In this lab, you will learn about Cross-Site Request Forgery (CSRF) attacks and how to implement protection mechanisms to safeguard web applications. You will explore how CSRF attacks exploit the trust that a web application has in the user's browser.

Prerequisites

- Basic knowledge of web application security concepts.
- Familiarity with session management and HTTP requests.
- Access to a web application environment (e.g., DVWA) for testing.

Lab Objectives

- Understand how CSRF attacks work and their impact on web applications.
- Implement CSRF protection using tokens.
- Test the effectiveness of CSRF defenses.

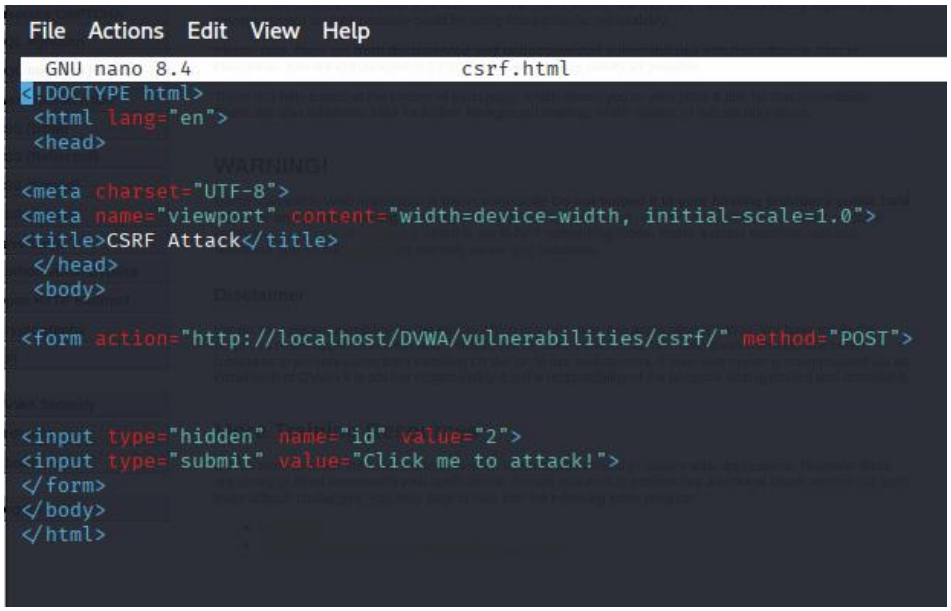
Exercise 1: Understanding CSRF Attacks

1. **Explore DVWA:**

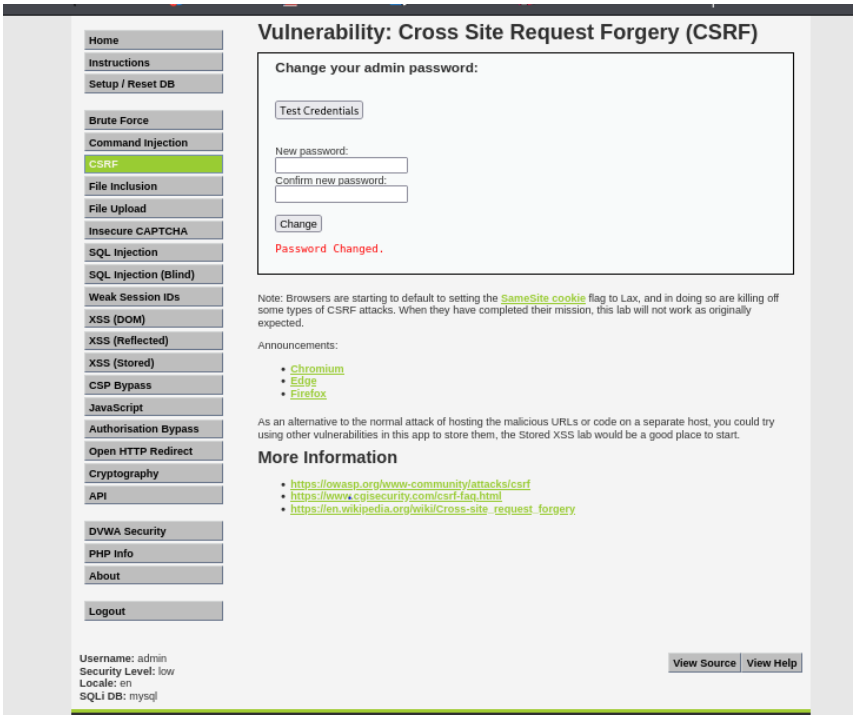
- Open DVWA in your browser and log in with your credentials (admin/password).
- Navigate to the CSRF section of the application.

2. Simulate a CSRF Attack:

- Open a new browser tab or window and create a simple HTML form to simulate a CSRF attack.
- Replace http://your-dvwa-url with your actual DVWA URL.



- Open this HTML file in your browser and click the submit button after logging into DVWA



.

3. Evaluate the Attack:

- Yes it successfully change the user settings or execute a sensitive action?
  - Yes, it did work because it displayed “Password Changed”
  - Also, when i log out i was able to log in with the new password i change to

Discuss how this demonstrates the importance of CSRF protection.

- CSRF exploit the trust between a website and its user
- It shows DVWA doesn’t check for csrf validation because it accept the forged request
- It poses a real-world danger

Exercise 2: Implementing CSRF Protection

1. Modify the Application Code:

- In your DVWA environment, navigate to the CSRF protection implementation area. If you don’t have one, you can add a CSRF token to forms manually.
- PHP Code Example for generating and validating CSRF tokens:

```php
<?php
define( 'DVWA_WEB_PAGE_TO_ROOT', '../../' );
require_once DVWA_WEB_PAGE_TO_ROOT . 'dvwa/includes/dvwaPage.inc.php';

dvwaPageStartup( array( 'authenticated' ) );

// Start session (DVWA does this, but ensure active)
if (session_status() === PHP_SESSION_NONE) {
    session_start();
}

// Generate CSRF token
if (empty($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}

$page = dvwaPageNewGrab();
$page[ 'title' ]   = 'Vulnerability: Cross Site Request Forgery (CSRF)' . $page[ 'title_separator' ].$page[ 'title'
$page[ 'page_id' ] = 'csrf';
$page[ 'help_button' ]   = 'csrf';
$page[ 'source_button' ] = 'csrf';

dvwaDatabaseConnect();

$vulnerabilityFile = '';
switch( dvwaSecurityLevelGet() ) {
```

2. Validate CSRF Tokens:

- Add validation logic to check the CSRF token upon form submission:

```php
session_start();

 // Validate the CSRF token
 if ($_SERVER['REQUEST_METHOD'] === 'POST') {

if (!hash_equals($_SESSION['csrf_token'],
$_POST['csrf_token'])) {
}
```

3. Reflection:

- Discuss how CSRF tokens work to prevent unauthorized actions. What other security measures could complement this?

How CSRF Tokens Work:

CSRF (Cross-Site Request Forgery) tokens are unique, unpredictable values generated by the server and tied to a user's session. Each time a form or state-changing request is made, the token is embedded in the request (usually as a hidden form field or in a header). When the server receives the request, it validates the submitted token against the one stored in the user's session.

- If the token is valid the request will be accepted.
- If the token is missing, invalid, or reused the request will be rejected.

This prevents attackers from performing unauthorized actions on behalf of a logged-in user because they cannot predict or obtain the valid token.

Session Management and Web Security

Challenge Title: Testing for Session Hijacking on AIVTIC Portal

Objective

The objective of this CTF challenge is to enhance your practical skills in identifying and exploiting session hijacking vulnerabilities. You will use your login credentials to attempt to detect and exploit potential session hijacking issues on the AIVTIC portal.

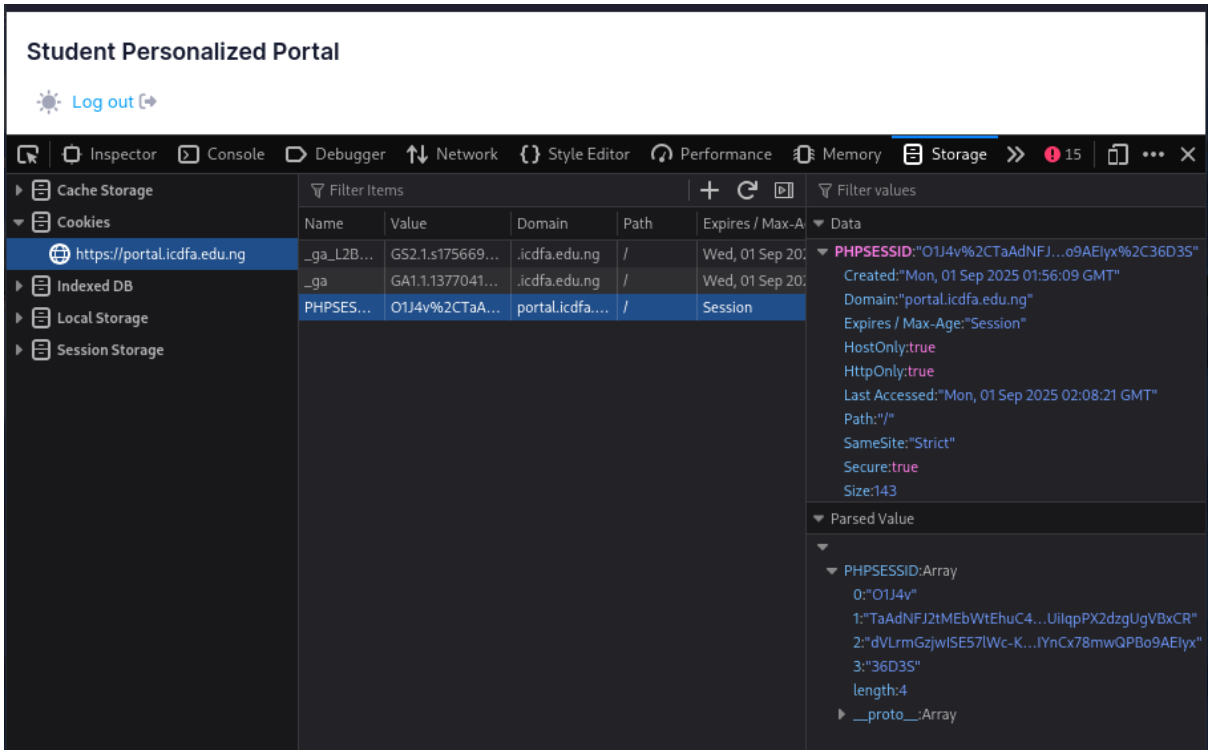
Challenge Overview

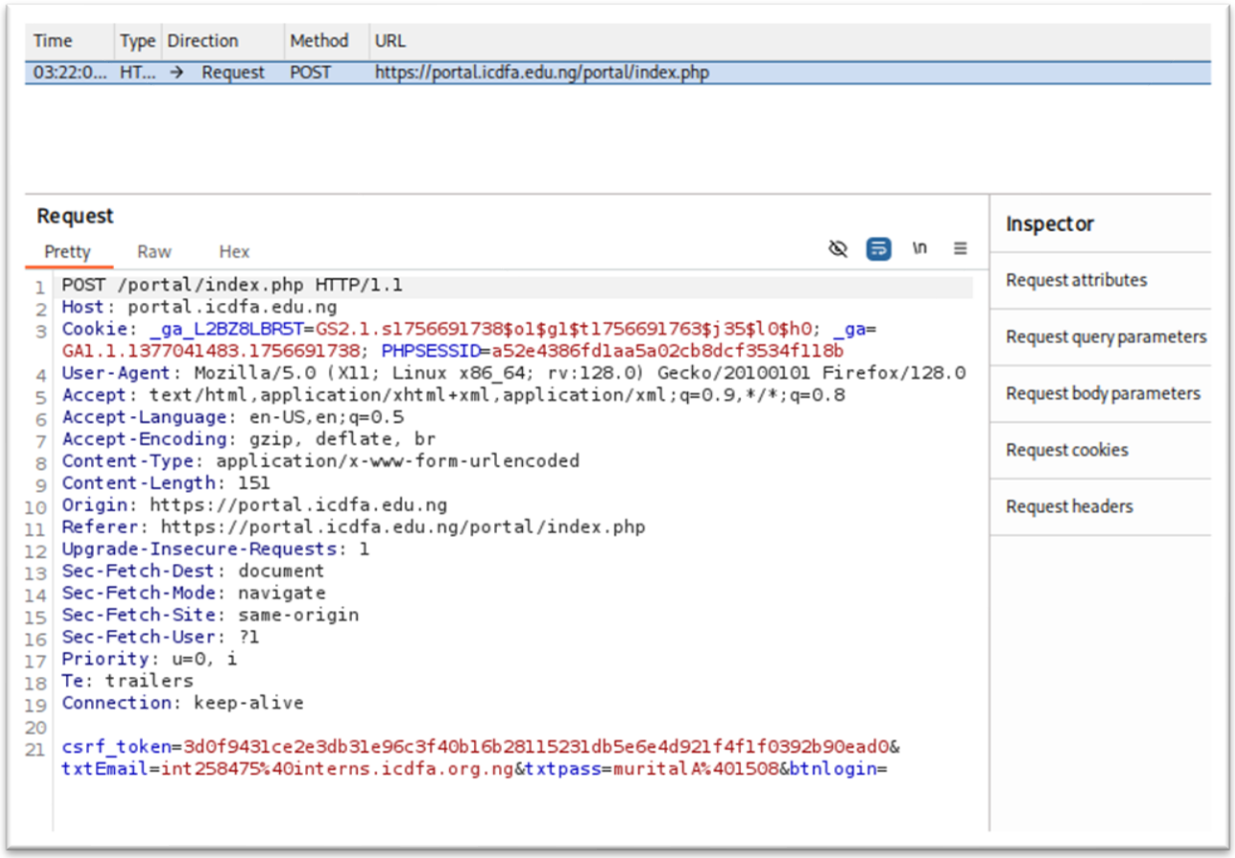
1. Understanding Session Hijacking

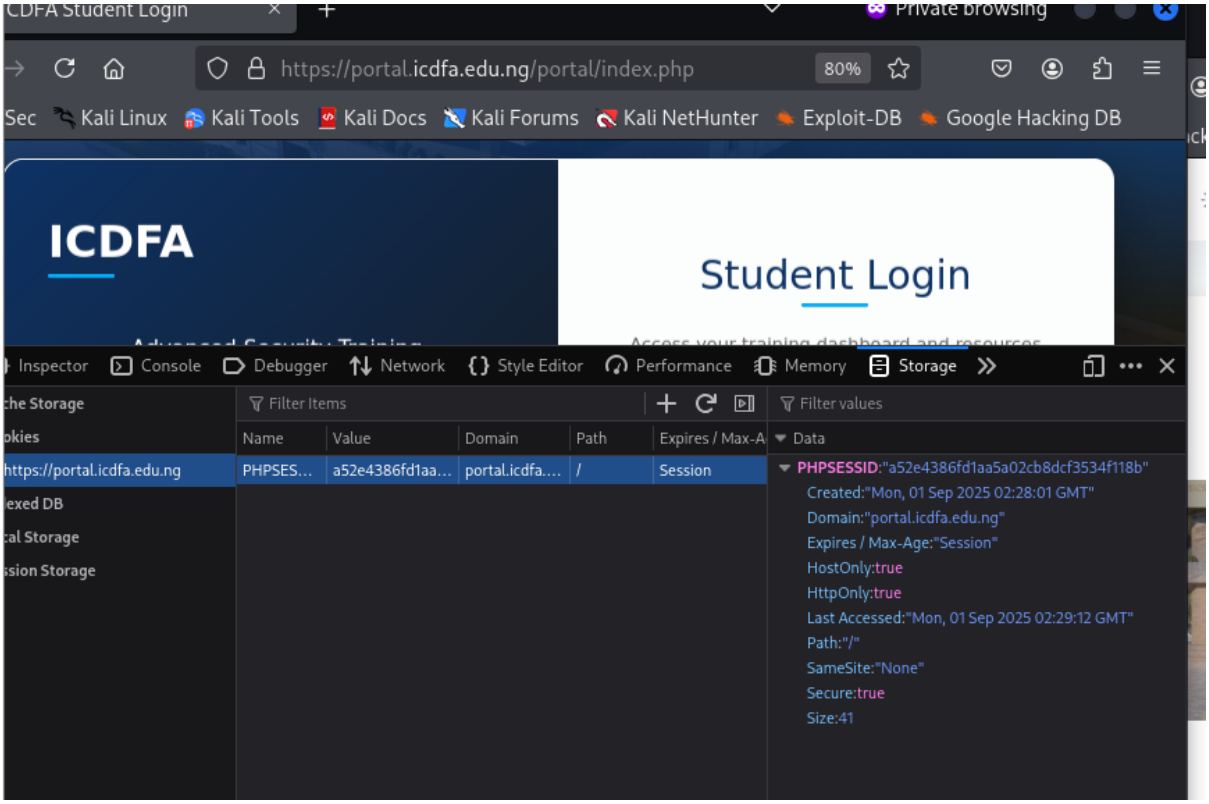
- Research: Start by reviewing concepts of session hijacking, focusing on how attackers exploit weaknesses in web session management. Understand the tools and methods commonly used to detect session-related vulnerabilities.

2. Test Setup

- Access: Log in to the AIVTIC portal using your assigned credentials.
- Observation: Use browser Developer Tools, Burp Suite, or OWASP ZAP to observe session cookies and their management in your browser.

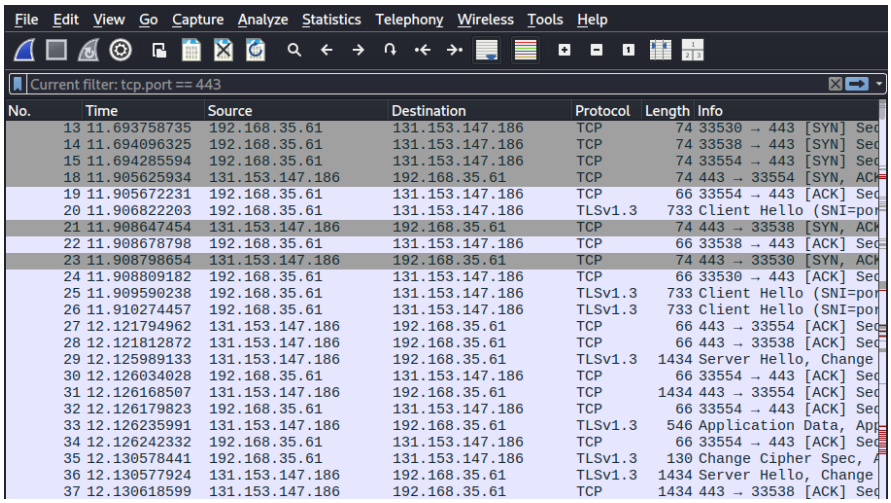


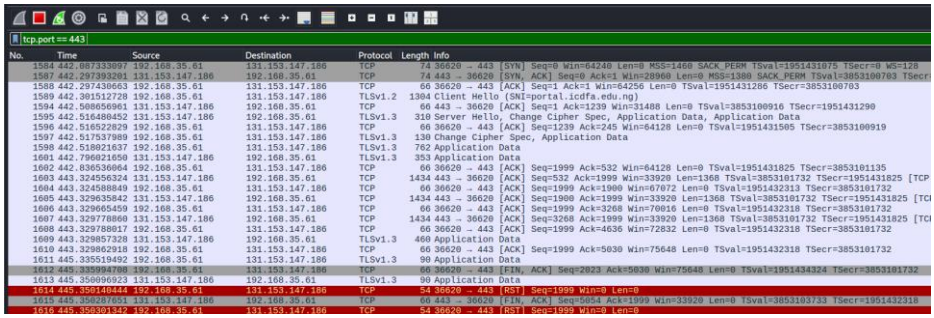

Time Type Direction Method URL
03:22:0... HT... → Request POST https://portal.icdfa.edu.ng/portal/index.php
Request
Pretty Raw Hex
1 POST /portal/index.php HTTP/1.1
2 Host: portal.icdfa.edu.ng
3 Cookie: _ga_L2BZ8LBR5T=GS2.1.s1756691738$o1$g1$t1756691763$j35$l0$h0; _ga=GA1.1.1377041483.1756691738; PHPSESSID=a52e4386fd1aa5a02cb8dcf3534f118b
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 151
10 Origin: https://portal.icdfa.edu.ng
11 Referer: https://portal.icdfa.edu.ng/portal/index.php
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 Priority: u=0, i
18 Te: trailers
19 Connection: keep-alive
20
21 csrf_token=3d0f9431ce2e3db31e96c3f40b16b28115231db5e6e4d921f4f1f0392b90ead0&txtEmail=int258475%40interns.icdfa.org.ng&txtpass=muritalA%401508&btnlogin=
Inspector
Request attributes
Request query parameters
Request body parameters
Request cookies
Request headers


CDFA Student Login
Private browsing
https://portal.icdfa.edu.ng/portal/index.php
80%
Sec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB
ICDFA
Student Login
Inspector Console Debugger Network Style Editor Performance Memory Storage
Filter Items
Filter values
Name Value Domain Path Expires / Max-A
PHPSES... a52e4386fd1aa... portal.icdfa.... / Session
Data
PHPSESSID:"a52e4386fd1aa5a02cb8dcf3534f118b"
Created:"Mon, 01 Sep 2025 02:28:01 GMT"
Domain:"portal.icdfa.edu.ng"
Expires / Max-Age:"Session"
HostOnly:true
HttpOnly:true
Last Accessed:"Mon, 01 Sep 2025 02:29:12 GMT"
Path:"/"
SameSite:"None"
Secure:true
Size:41

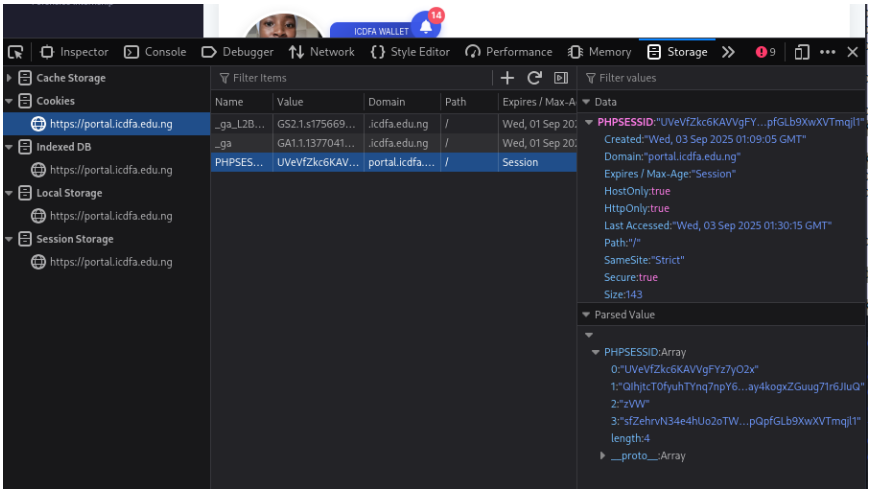
3. Session Hijacking Attempt

- Simulate an Attack: Utilize the knowledge gained to simulate a session hijacking attack on your own session. Use tools like Wireshark to monitor session traffic, intercept cookies, or test session expiration scenarios.





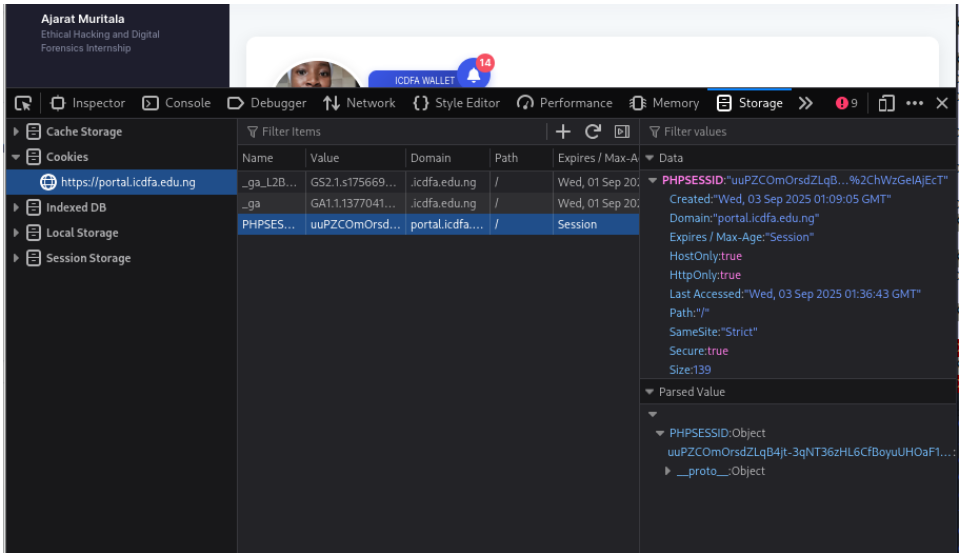
- Key Focus Areas: Session ID Predictability: Investigate if the session ID can be easily guessed or regenerated.
    - The session ID can not be easily guessed so its not predictsble
    - The session ID generated after login appeared random and non-sequential.
    - No pattern or predictable structure was identified.
    - reduces the risk of attackers guessing or brute-forcing valid session IDs.



Below are the two cookies i get after logging out and they did not look predictabe

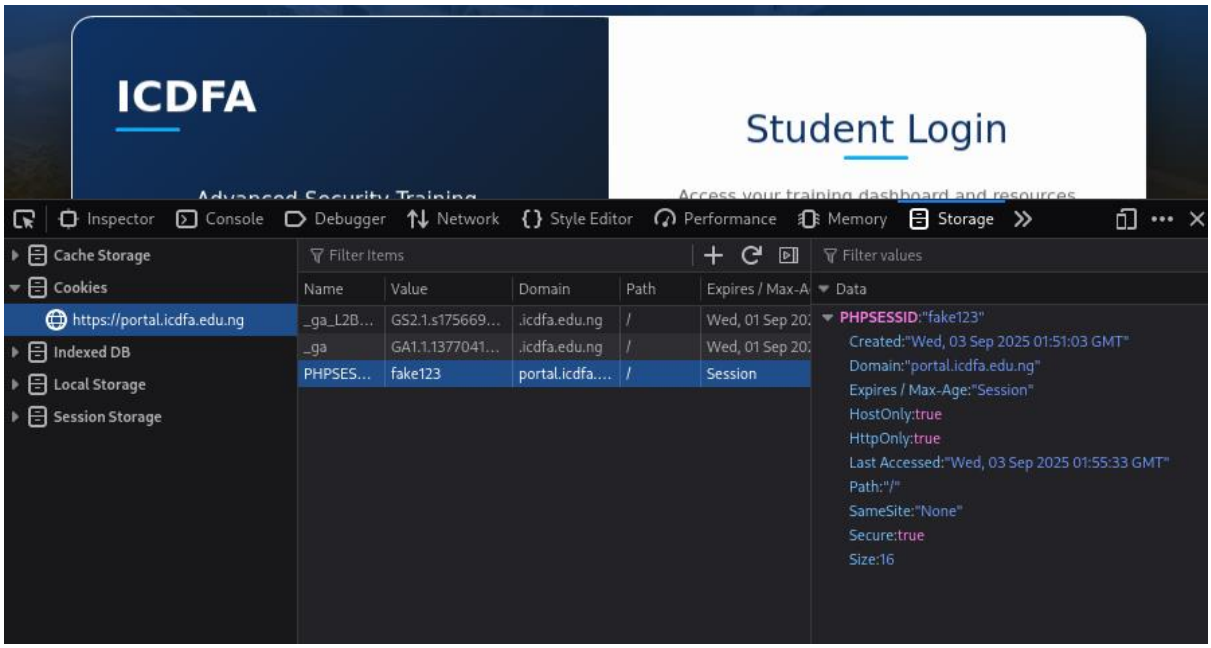
- UVeVfZkc6KAVVgFYz7yO2x%2CQIhjtcT0fyuhTYnq7npY66GHkay4kogxZGuug71r6JIuQ%2CzVW%2CsfZehrvN34e4hUo2oTWbYUtOGFDjLct6KEepKpQpfGLb9XwXVTmqjl1

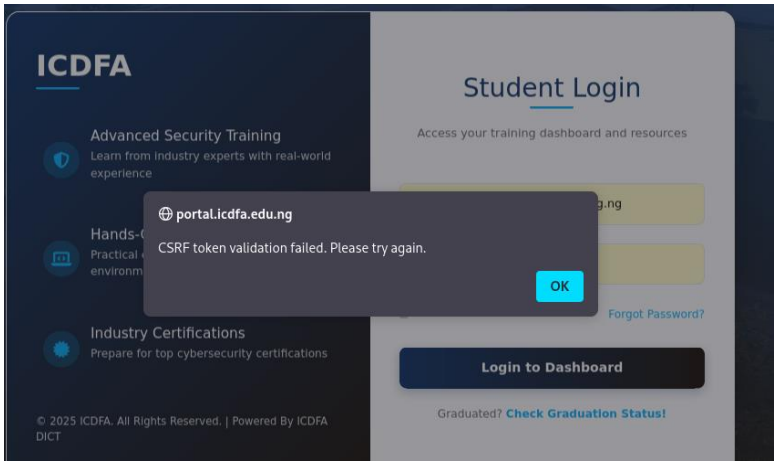
- uuPZCOmOrsdZLqB4jt-3qNT36zHL6CfBoyuUHOaF1CnuQcgvAYIllcz7MOAM9jzDbGv1WNMaM7ZzFNoEh9Z7qgYSF0dbFGpIej29bEFZOnesneUiZX1O%2ChWzGeIAjEcT



Session Fixation: Test whether you can set a user's session ID before login and hijack the session.

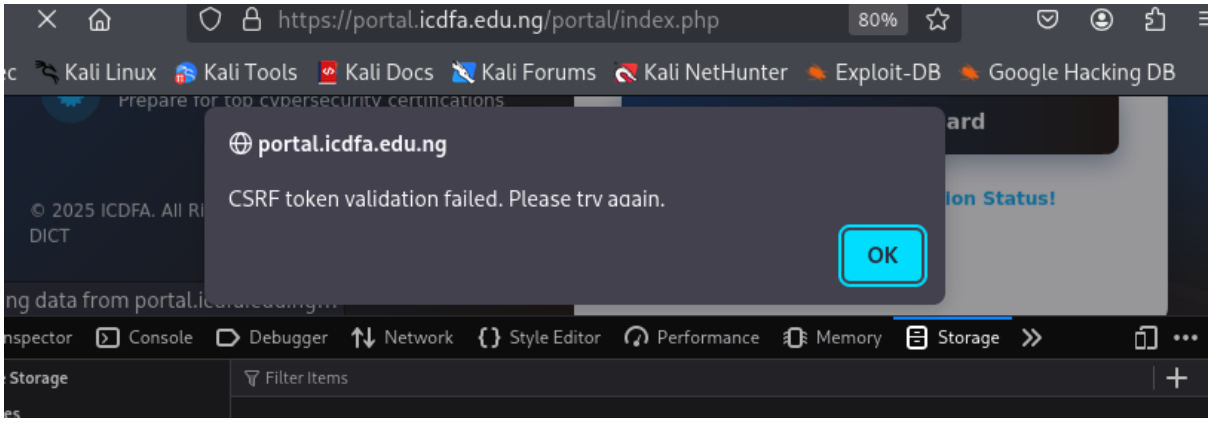
- No you cant set a users session id before log in so you cant hijack the session
- The system returned “CSRF token validation failed”, preventing fixation attempts.
- The CSRF token mechanism is functioning as an additional security layer





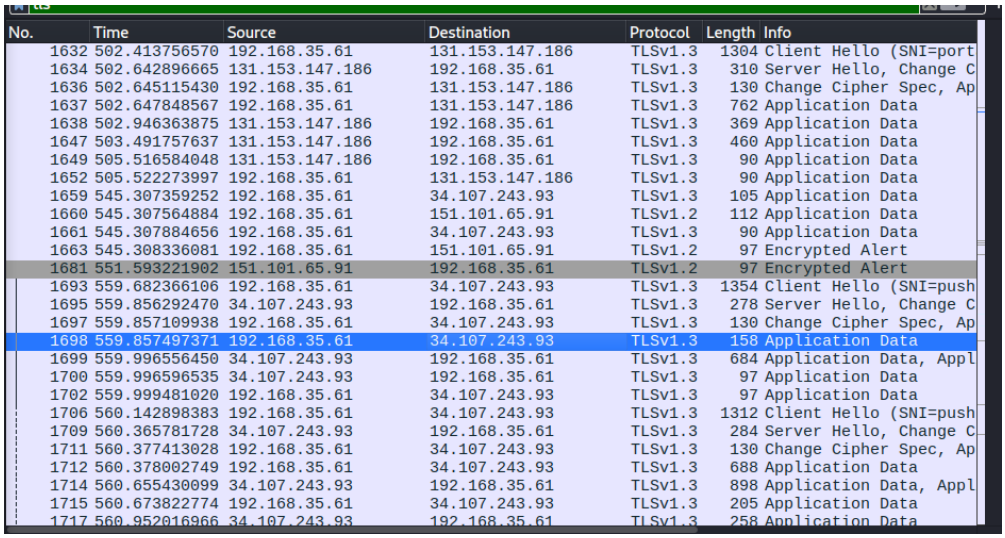
Session Expiration: Confirm if the session terminates correctly after logout or inactivity.

- Yes the session terminates correctly afterog out or inactivity
- Session reuse attempts also triggered “CSRF token validation failed.
- The portal correctly invalidates old sessions, preventing hijacking via expired cookies.



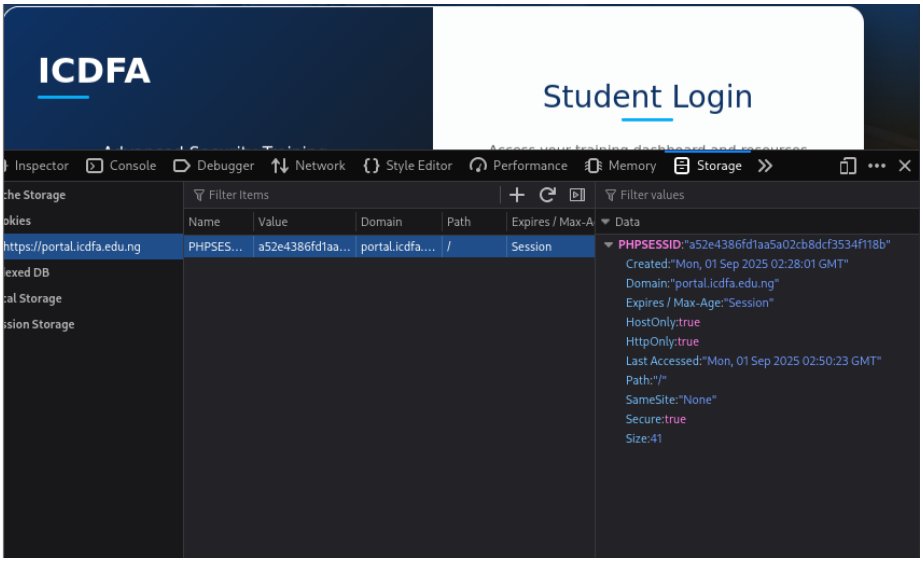
Transport Security: Evaluate if session tokens are transmitted securely (via HTTPS).

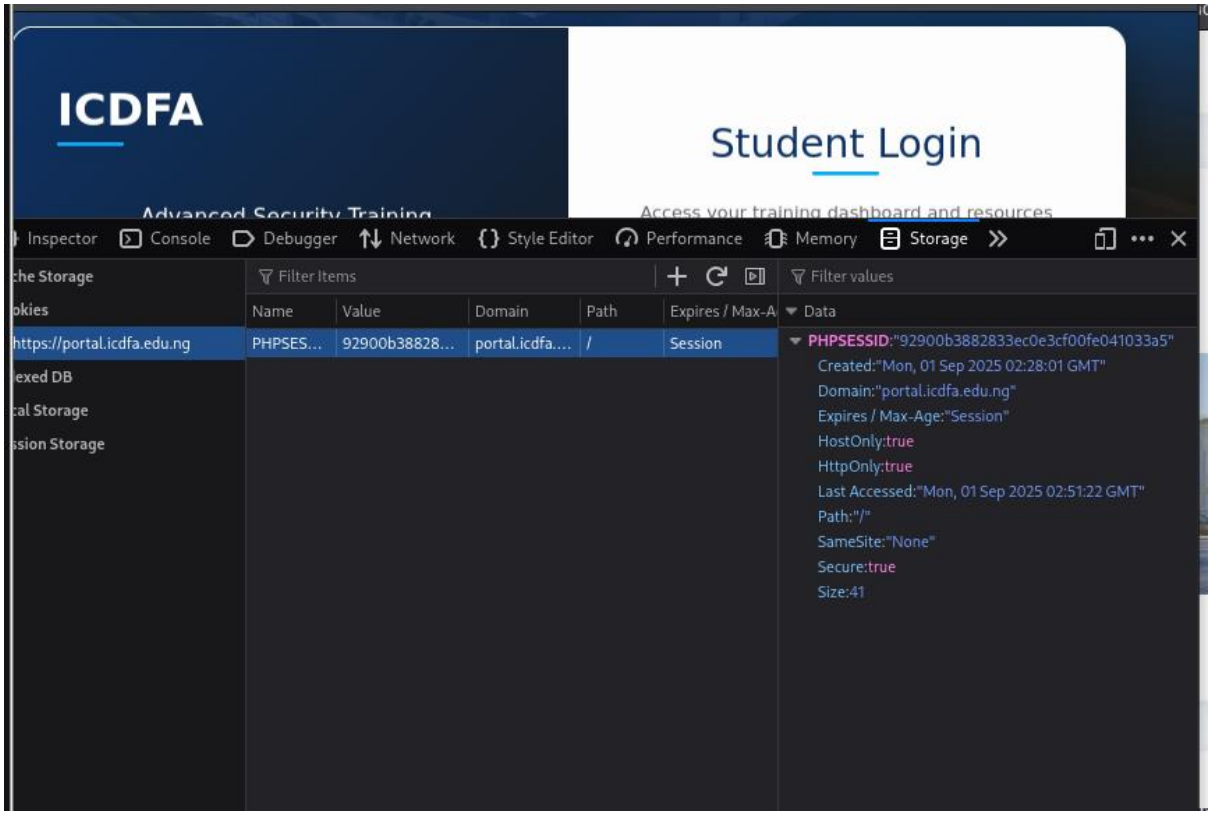
- Using Wireshark, login and session-related traffic appeared as TLS 1.2/1.3 “Application Data” packets, indicating encryption.
- Credentials and tokens are not exposed over the network, preventing packet-sniffing attacks



No.	Time	Source	Destination	Protocol	Length	Info
1632	502.413756570	192.168.35.61	131.153.147.186	TLSv1.3	1304	Client Hello (SNI=port
1634	502.642896665	131.153.147.186	192.168.35.61	TLSv1.3	310	Server Hello, Change C
1636	502.645115430	192.168.35.61	131.153.147.186	TLSv1.3	130	Change Cipher Spec, Ap
1637	502.647848567	192.168.35.61	131.153.147.186	TLSv1.3	762	Application Data
1638	502.946363875	131.153.147.186	192.168.35.61	TLSv1.3	369	Application Data
1647	503.491757637	131.153.147.186	192.168.35.61	TLSv1.3	460	Application Data
1649	505.516584048	131.153.147.186	192.168.35.61	TLSv1.3	90	Application Data
1652	505.522273997	192.168.35.61	131.153.147.186	TLSv1.3	90	Application Data
1659	545.307359252	192.168.35.61	34.107.243.93	TLSv1.3	105	Application Data
1660	545.307564884	192.168.35.61	151.101.65.91	TLSv1.2	112	Application Data
1661	545.307884656	192.168.35.61	34.107.243.93	TLSv1.3	90	Application Data
1663	545.308336081	192.168.35.61	151.101.65.91	TLSv1.2	97	Encrypted Alert
1681	551.593221902	151.101.65.91	192.168.35.61	TLSv1.2	97	Encrypted Alert
1693	559.682366106	192.168.35.61	34.107.243.93	TLSv1.3	1354	Client Hello (SNI=push
1695	559.856292470	34.107.243.93	192.168.35.61	TLSv1.3	278	Server Hello, Change C
1697	559.857109938	192.168.35.61	34.107.243.93	TLSv1.3	130	Change Cipher Spec, Ap
1698	559.857497371	192.168.35.61	34.107.243.93	TLSv1.3	158	Application Data
1699	559.996556450	34.107.243.93	192.168.35.61	TLSv1.3	684	Application Data, Appl
1700	559.996596535	34.107.243.93	192.168.35.61	TLSv1.3	97	Application Data
1702	559.999481020	192.168.35.61	34.107.243.93	TLSv1.3	97	Application Data
1706	560.142898383	192.168.35.61	34.107.243.93	TLSv1.3	1312	Client Hello (SNI=push
1709	560.365781728	34.107.243.93	192.168.35.61	TLSv1.3	284	Server Hello, Change C
1711	560.377413028	192.168.35.61	34.107.243.93	TLSv1.3	130	Change Cipher Spec, Ap
1712	560.378002749	192.168.35.61	34.107.243.93	TLSv1.3	688	Application Data
1714	560.655430099	34.107.243.93	192.168.35.61	TLSv1.3	898	Application Data, Appl
1715	560.673822774	192.168.35.61	34.107.243.93	TLSv1.3	205	Application Data
1717	560.952016966	34.107.243.93	192.168.35.61	TLSv1.3	258	Application Data

a52e4386fd1aa5a02cb8dcf3534f118b





4. Defense Mechanisms

- Identify Vulnerabilities: After testing, propose practical countermeasures to prevent session hijacking attacks. This may include enhancing session management, adjusting cookie attributes, or implementing encryption.

Identified Vulnerabilities

- Burp Suite interception showed username and password in plaintext inside the HTTPS request body.
- No evidence that cookies are set with HttpOnly, Secure, or SameSite.
- Lack of these attributes increases risk of theft via XSS or CSRF.
- While expired sessions returned **"CSRF token validation failed,"** there was no strict idle timeout confirmation.
- Long-lived sessions may remain active if a user forgets to log out.
- The portal does not appear to monitor for unusual session behavior (e.g., IP changes, concurrent logins).
- Attackers could exploit valid tokens without being detected.

Countermeasures to prevent session hijacking attacks.

- Enforce Strong TLS Security
- Enable HTTP Strict Transport Security (HSTS).
- Protect Login Credentials
- Implement Multi-Factor Authentication (MFA).
- Improve Session Timeout Policies
- Reauthentication required for sensitive actions.
- Monitor Active Sessions
- Detect anomalies (IP changes, multiple logins).
- Allow users to view/revoke active sessions.
- Alert or block suspicious concurrent logins.