# Lab Exercise: PHP Security Practices

Duration: 1:00 hours

## Objective:

To apply the concepts of sending emails in PHP, managing sessions securely, and protecting against XSS attacks.

## Requirements:

- A local development environment like XAMPP or a similar server setup.
- Basic knowledge of PHP, HTML, and web security concepts.
- A text editor or PHP IDE.

## Part 1: Secure Session Management

**Goal:** Create a secure login and logout mechanism using PHP sessions.

### Task:

1. **Create Login Script (`login.php`):**
   - Start a session and check for user-submitted login credentials.
   - Implement session ID regeneration upon successful login.
   - Redirect the user to a protected page (**`dashboard.php`**).
2. **Create Dashboard Page (`dashboard.php`):**
   - Ensure the session is active and the user is logged in.
   - Provide a logout option that destroys the session.
3. **Implement Logout Functionality (`logout.php`):**
   - Destroy the session and redirect the user back to the login page.

### Detailed Instructions:

- Use **`session_start()`** at the beginning of each script.
- Validate hardcoded username and password in **`login.php`**.
- In **`dashboard.php`**, check if a session variable (e.g., **`$_SESSION['loggedin']`**) is set.
- Use **`session_destroy()`** and **`session_unset()`** in **`logout.php`** for ending the session.

## Part 2: Preventing XSS Attacks

**Goal:** Safely process and display user input from an HTML form.

**Task:**

1. **Create an HTML Form (`form.html`):**
    - Include a text input and a submit button.
2. **Process Form Input (`process_input.php`):**
    - Capture user input and sanitize it to prevent XSS attacks.
    - Display the sanitized input back to the user.

**Detailed Instructions:**

- In **`process_input.php`**, use the **`htmlspecialchars()`** function to sanitize user input.
- Ensure that the form data is displayed in a way that any HTML tags are rendered harmless.

## Part 3: Implementing CSRF Protection

**Goal:** Implement CSRF protection in a PHP form.

**Task:**

1. **Create a Form with CSRF Token (`csrf_form.php`):**
    - Start a session and generate a unique CSRF token.
    - Store the token in the session and include it as a hidden field in the form.
2. **Process the Form Submission (`process_csrf.php`):**
    - Validate the CSRF token from the form against the session token.
    - Display a success message if the token is valid, or an error if not.

**Detailed Instructions:**

- Use **`bin2hex(random_bytes(32))`** to generate a CSRF token.
- Store and compare the CSRF token securely to prevent token leakage or tampering.

## Part 4: Setting Up Content Security Policy (CSP)

**Goal:** Configure CSP headers to enhance security.

**Task:**

1. **Implement CSP in a PHP Page (`csp_setup.php`):**

- Create a PHP script that outputs a webpage.
- Set CSP headers to restrict resource loading (scripts, styles, images).

2. **Test CSP Implementation:**
    - Try including external scripts or styles and observe CSP in action.
    - Check browser's console for CSP violation reports.

### Detailed Instructions:

- Use the `header()` function to set `Content-Security-Policy`.
- Start with a restrictive policy and gradually allow specific sources as needed.
- Example policy: `header("Content-Security-Policy: default-src 'self'; script-src 'self'; style-src 'self';");`