Republic of the Philippines
Laguna State Polytechnic University
Province of Laguna

College of Computer Studies

Laboratory Exercise: User Account Security & Linking Permissions to User Management

Course: BSIT (Network Administrator)
Instructor: Jemar Banawa
Laboratory Title: User Account Security & Linking Permissions to User Management
Lab Number:
Duration: March 21, 2025 (10:00 AM)

Objectives:

At the end of this laboratory exercise, students should be able to:

1. Understand the importance of user account security.
2. Implement user authentication and authorization.
3. Configure user roles and permissions in a web application.
4. Manage user accounts securely following best practices.

Materials Needed:

- Computer with internet access
- Installed development environment (XAMPP/WAMP, Visual Studio Code, or any preferred IDE)
- MySQL Database
- PHP (for backend user management)
- Web browser
- Sample web application (can be provided by the instructor or created by students)

Preliminary Discussion:

In modern web applications, securing user accounts is crucial to prevent unauthorized access and potential data breaches. User account security involves authentication (verifying user identity) and authorization (granting appropriate permissions). Effective user management ensures that users only access the resources relevant to their roles.

Laboratory Procedure:

Part 1: Setting Up User Authentication

1. **Create a User Registration and Login System**
   - Set up a MySQL database with a `users` table containing fields: `id`, `username`, `password (hashed)`, `role`.
   - Develop a registration page where users can create an account.
   - Implement password hashing using PHP's `password_hash()` function.
   - Create a login page with input validation and authentication.
2. **Implement Session Management**
   - Use PHP sessions to store login state.
   - Redirect users to a dashboard after successful login.
   - Implement a logout function that destroys the session.

Part 2: Implementing User Roles and Permissions

3. **Define User Roles in the Database**
   - Modify the `users` table to include a `role` field (e.g., Admin, Editor, Viewer).
   - Populate sample user accounts with different roles.
4. **Restrict Page Access Based on Roles**
   - Create a `middleware.php` script to check user roles before granting access to specific pages.
   - Example: Admin users can access `admin_panel.php`, while regular users are restricted.
5. **Role-Based Navigation**
   - Modify the web application's navigation to display options based on user roles.
   - Example: Admins see user management options, while regular users only see general features.

Part 3: Best Practices for User Account Security

6. **Enforce Strong Password Policies**
   - Require a minimum password length and complexity (uppercase, lowercase, numbers, special characters).
   - Implement password change and recovery options securely.
7. **Implement Account Lockout for Multiple Failed Logins**
   - Track failed login attempts and temporarily lock accounts after multiple failures.
8. **Use HTTPS for Secure Data Transmission**
   - Ensure that authentication pages use HTTPS to encrypt user credentials.

Assessment Tasks:

1. **Code Review:** Submit the implemented authentication and authorization system.
2. **Security Analysis:** Identify vulnerabilities in the system and suggest improvements.
3. **Demonstration:** Present the working user management system and explain how permissions are enforced. (Explain through Documentation)

Reflection Questions:

1. Why is user authentication necessary in web applications?
2. How does role-based access control enhance security?
3. What are the common security threats in user management, and how can they be prevented?

Submission Instructions:

- Submit your source code in a `.zip` file.
- Include a short documentation explaining your implementation.
- Upload the files to the learning management system before the deadline.