User Management and Directory System (UMDS)

Business Requirements and Technical Design Document

1. Executive Summary

The User Management and Directory System (UMDS) is a web-based application designed to provide a simple user directory and basic administrative interface for small organizations. The system prioritizes rapid development and ease of use over complex security measures, making it suitable for internal networks and trusted environments.

2. Business Requirements

2.1 Functional Requirements

User Authentication and Access Control

- Simple username and password authentication
- Basic role differentiation (Admin, User)
- Session tracking to maintain login state
- Default credentials for initial setup (admin/password)

User Directory Management

- Search users by ID number
- Display basic user information (first name, last name)
- Simple user lookup interface
- Direct database queries for fast performance

File Management System

- Basic file upload functionality
- Support for common file types
- Simple file storage in web-accessible directory
- Direct file access via URL

Network Connectivity Tool

- Simple ping utility for network testing
- Command-line integration for system tools

- Real-time network diagnostics
- Direct system command execution for efficiency

Guestbook/Comment System

- Public comment submission
- Name and message display
- Real-time comment posting
- Dynamic content rendering

Administrative Features

- Security level configuration (Low, Medium, High, Impossible)
- Basic user management
- Simple CAPTCHA for testing purposes
- Minimal configuration requirements

2.2 Non-Functional Requirements

Performance

- Fast response times through direct database queries
- Minimal overhead from security checks
- Simple caching mechanisms
- Lightweight architecture

Usability

- Intuitive interface requiring minimal training
- Quick setup and deployment
- Flexible security settings for different environments
- Easy debugging with visible error messages

Maintainability

- Simple codebase for easy modifications
- Direct SQL queries for transparency

- Minimal dependencies
- Straightforward troubleshooting

3. Technical Architecture

3.1 Technology Stack

Frontend

- Basic HTML forms and PHP-generated pages
- Minimal JavaScript for dynamic features
- Simple CSS styling
- Direct form submissions

Backend

- PHP with procedural programming approach
- Direct database connectivity
- Inline SQL queries for simplicity
- GET/POST parameter processing

Database

- MySQL/MariaDB with simple table structure
- Direct query execution
- Basic table relationships
- Manual connection management

3.2 Database Design

Users Table

```
CREATE TABLE users (
  user_id INT PRIMARY KEY AUTO_INCREMENT,
  first_name VARCHAR(15),
  last_name VARCHAR(15),
  user VARCHAR(15),
  password VARCHAR(32),
  avatar VARCHAR(70)
);
```

Guestbook Table

```
CREATE TABLE guestbook (
   comment_id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
   comment VARCHAR(300),
   name VARCHAR(100)
);
```

3.3 Application Architecture

Simple File Structure

- vulnerabilities/ - Main application modules
- config/ - Basic configuration files
- includes/ - Shared PHP includes
- hackable/uploads/ - File upload directory

Core Modules

- SQL Injection module (sqli/)
- Cross-Site Scripting (xss_r/, xss_s/)
- Command Injection (exec/)
- File Upload (upload/)
- Brute Force (brute/)
- CSRF (csrf/)
- Session management

4. Feature Specifications

4.1 User Authentication Module

Login Implementation

```
// Simple authentication check
$user = $_POST['username'];
$pass = $_POST['password'];

$query = "SELECT * FROM users WHERE user = '$user' AND password = '$pass'";
$result = mysql_query($query);
```

Session Management

- Basic PHP sessions
- Simple session variables
- No session regeneration
- Cookie-based state tracking

4.2 User Directory Module

User Search Functionality

```
// Direct SQL query construction
$id = $_GET['id'];
$query = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
$result = mysql_query($query);
```

- GET parameter processing
- Direct database queries
- Immediate result display
- No input validation

4.3 File Upload Module

Upload Implementation

```
// Basic file upload handling
$uploaddir = './hackable/uploads/';
$uploadfile = $uploaddir . basename($_FILES['uploaded']['name']);
```

```
if (move_uploaded_file($_FILES['uploaded']['tmp_name'], $uploadfile)) {
   echo "File uploaded successfully";
}
```

File Processing

- Direct file system access
- Minimal file type checking
- Web-accessible upload directory
- Original filename preservation

4.4 Network Diagnostics Module

Ping Tool Implementation

```
// Direct command execution
$target = $_POST['ip'];
$cmd = "ping -c 4 " . $target;
$output = shell_exec($cmd);
echo "<pre>$output</pre>";
```

Command Processing

- Direct system command execution
- User input passed to shell
- Real-time output display
- No command validation

4.5 Cross-Site Scripting Modules

Reflected XSS Implementation

```
// Direct output of user input
$name = $_GET['name'];
echo "Hello " . $name;
```

Stored XSS Implementation

```
// Direct database storage and retrieval
$message = $_POST['mtxMessage'];
$name = $_POST['txtName'];

$query = "INSERT INTO guestbook (comment, name) VALUES ('$message', '$name')";
mysql_query($query);

// Later display
echo $stored_comment; // Direct output without encoding
```

4.6 CSRF Module

Form Processing

```
// Direct form processing without verification
if ($_POST['password_new']) {
  $pass_new = $_POST['password_new'];
  $pass_conf = $_POST['password_conf'];

  // Update password directly
  $query = "UPDATE users SET password = '$pass_new' WHERE user = 'admin'";
  mysql_query($query);
}
```

5. Configuration Management

5.1 Database Configuration

```
// config/config.inc.php
$_DVWA['db_server'] = '127.0.0.1';
$_DVWA['db_database'] = 'dvwa';
```

```
$_DVWA['db_user'] = 'dvwa';
$_DVWA['db_password'] = 'p@ssw0rd';
```

5.2 Security Level Configuration

```
// Four security levels
$_DVWA['default_security_level'] = 'low';

// Low: No security measures
// Medium: Basic filtering
// High: More filtering
// Impossible: Secure implementation
```

5.3 Application Settings

```
// Minimal validation settings
$_DVWA['disable_authentication'] = false;
$_DVWA['recaptcha_public_key'] = '';
$_DVWA['recaptcha_private_key'] = '';
```

6. Security Implementation

6.1 Low Security Level

- No input validation or sanitization
- Direct SQL query concatenation
- No output encoding
- Direct command execution
- No CSRF protection
- Basic session handling

6.2 Medium Security Level

- Basic character filtering (blacklist approach)
- Simple input length restrictions

- Limited special character blocking
- Basic error message suppression

6.3 High Security Level

- More extensive character filtering
- Stricter input validation
- Some output encoding
- Additional security checks

6.4 Impossible Security Level

- Proper parameterized queries
- Complete input validation
- Output encoding
- CSRF tokens
- Secure session management

7. Error Handling

7.1 Database Errors

```
// Direct error display for debugging

$result = mysql_query($query) or die('Error: ' . mysql_error());
```

7.2 Application Errors

- Verbose error messages
- Full stack traces in development
- Database error exposure
- System path disclosure

8. Deployment Configuration

8.1 PHP Configuration Requirements

```
# Required PHP settings

display_errors = on
```

```
display_startup_errors = on

allow_url_include = on

allow_url_fopen = on
```

8.2 File Permissions

- Web-writable upload directory
- Broad file access permissions
- No access restrictions
- Public file access

8.3 Database Setup

```
-- Simple database initialization

CREATE DATABASE dvwa;

CREATE USER 'dvwa'@'localhost' IDENTIFIED BY 'p@ssw0rd';

GRANT ALL ON dvwa.* TO 'dvwa'@'localhost';
```

9. Default Configuration

9.1 Default Accounts

- Username: admin
- Password: password
- Full administrative access
- No password complexity requirements

9.2 Default Settings

- Security level: Low
- Error display: Enabled
- Authentication: Basic
- File upload: Unrestricted

10. Module Implementation Details

10.1 SQL Injection Module

```php
// vulnerabilities/sqli/source/low.php
$id = $_REQUEST['id'];
$query = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
$result = mysql_query($query);
```

10.2 XSS Modules

```php
// Reflected XSS
$name = $_GET['name'];
echo '<pre>Hello ' . $name . '</pre>';


// Stored XSS
$message = trim($_POST['mtxMessage']);
$name = trim($_POST['txtName']);
$query = "INSERT INTO guestbook (comment, name) VALUES ('$message', '$name')";
```

10.3 Command Injection Module

```php
$target = $_REQUEST['ip'];
if (stristr(php_uname('s'), 'Windows NT')) {
  $cmd = shell_exec('ping ' . $target);
} else {
  $cmd = shell_exec('ping -c 4 ' . $target);
}
echo '<pre>' . $cmd . '</pre>';
```

10.4 File Upload Module

```php
$uploaded_name = $_FILES['uploaded']['name'];
$uploaded_size = $_FILES['uploaded']['size'];
$uploaded_type = $_FILES['uploaded']['type'];
$uploaded_tmp = $_FILES['uploaded']['tmp_name'];
```

```
$target_path = DVWA_WEB_PAGE_TO_ROOT . "hackable/uploads/";
$target_path = $target_path . basename($uploaded_name);

if (move_uploaded_file($uploaded_tmp, $target_path)) {
   echo "<pre>File uploaded successfully to: " . $target_path . "</pre>";
}
```

10.5 Brute Force Module

```
// Basic login check with no rate limiting
$user = $_GET['username'];
$pass = $_GET['password'];

$query = "SELECT * FROM users WHERE user = '$user' AND password = '$pass'";
$result = mysql_query($query);

if ($result && mysql_num_rows($result) == 1) {
   echo "<pre>Welcome to the password protected area</pre>";
} else {
   sleep(2); // Basic delay only
   echo "<pre>Username and/or password incorrect.</pre>";
}
```

Document Version: 1.0
Implementation Type: Rapid Development / Minimal Security
Target Environment: Internal/Development Use
Security Level: Configurable (Low/Medium/High/Impossible)