Task 3 Findings

PHP Security Implementation & Penetration Testing

Student Name:	Abdullah Akram
Student ID:	DHC-3606
Date:	July 10, 2025
Task:	Week 3 - Security Implementation

1. Manual Penetration Testing Findings

✓ Completed SQL Injection Testing

Performed manual SQL injection testing using browser-based techniques to identify potential vulnerabilities in the application.

📂 *No file modifications required - Browser-based testing only*

```
// Test URL used for SQL injection detection
http://localhost/yourpage.php?id=1'
```

✓ Completed Cross-Site Scripting (XSS) Testing

Conducted XSS vulnerability testing to identify potential script injection points in the application.

```
// Test URL used for XSS detection
```

```
http://localhost/yourpage.php?name=<script>alert(1)</script>
```

2. Security Logging Implementation

✓ Implemented **Basic PHP Logging System**

Successfully implemented a comprehensive logging system to track application activities and security events.

Implementation Location: config.php / index.php (Application entry point)

```
<?php
// Security logging implementation
$logMessage = "[" . date("Y-m-d H:i:s") . "] Application started" .
PHP_EOL; file_put_contents("security.log", $logMessage, FILE_APPEND);
?>
```

3. Backend Security Implementation

Authentication & Authorization Files

login.php

Input validation and password verification implementation

register.php

User registration with password hashing

process_form.php

Form data sanitization and validation

db.php

Database connection and security configuration

✓ Enhanced **Input Validation Implementation**

Implemented comprehensive input validation using PHP filter functions across all user input points.

```
// Example: Login form validation
$username = filter_input(INPUT_POST, 'username',
FILTER_SANITIZE_STRING); $password = $_POST['password'];
// Password verification
if (password_verify($password, $hashedPasswordFromDB)) { // login
success }
```

✓ Implemented **Password Security Enhancement**

Implemented secure password hashing using PHP's built-in password_hash() function with default algorithms.

```
// Registration: Password hashing
$username = filter_input(INPUT_POST, 'username',
FILTER_SANITIZE_STRING); $password = password_hash($_POST['password'],
PASSWORD_DEFAULT);
// Store hashed password in database
```

4. Secure Index.php Implementation

✓ Deployed **Comprehensive Security Headers**

Implemented essential security headers to protect against common web vulnerabilities including XSS, clickjacking, and MIME-type sniffing.

```
// Security headers implementation
header("X-Content-Type-Options: nosniff"); header("X-Frame-Options:
SAMEORIGIN"); header("X-XSS-Protection: 1; mode=block");
header("Strict-Transport-Security: max-age=31536000;
includeSubDomains");
```

✓ Secured Search Functionality Security

Enhanced search functionality with proper input sanitization, prepared statements, and comprehensive logging.

```
// Secure search processing
$keywords = array_filter(array_map('trim', explode(',',
filter_var($raw_keywords, FILTER_SANITIZE_STRING))));
// Security logging
$log = "[".date("Y-m-d H:i:s")."] Search - Keywords: $raw_keywords |
Location: $raw_location | IP: ".$_SERVER['REMOTE_ADDR'].PHP_EOL;
file_put_contents("security.log", $log, FILE_APPEND);
```

5. Security Implementation Checklist

- ✓ Inputs validated using PHP filter functions
- ✓ HTTPS configuration ready for secure data transmission
- ✓ Passwords hashed using password_hash() with secure algorithms
- ✓ Security headers implemented (X-Content-Type-Options, X-Frame-Options, X-XSS-Protection, HSTS)
- ✓ Prepared statements used for database queries
- ✓ Comprehensive logging system implemented
- ✓ Input sanitization applied across all user input points

✓ Session security configuration enhanced