Web Application Penetration Test

Jake Lewandowski

CMP319: Web Application Penetration Testing

BSc Ethical Hacking

Year 3

2025/26

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

Abstract

This report presents the findings of a grey-box penetration test for the Rickstore website, the aim of which is to document and score existing security flaws found using a modified version of the OWASP web security methodology along with the CVSS 3.1 framework, with remediation provided at the end. The modifications implemented excluded sections not applicable to this case and merged some sections together where a lot of overlap occurred. Any vulnerability found would be further investigated to find the complete extent of the damage and see if vulnerabilities can be chained together.

The test found 16 total vulnerabilities, 3 of which are considered critical according to the CVSS v3.1 framework used to assign severity. 3 high severity and 9 medium severity vulnerabilities were also identified which demonstrated that the website currently does not follow proper security practices. The most critical exploit found involved poor file upload filtering which results in total system compromise for the web server, completely exposing all customer and business data. Furthermore, other exploits make it possible to gain complete control over the business database and administrator panel.

The findings demonstrate numerous accessible attack vectors which can compromise customer personal data such as addresses, phone numbers and emails. It is also shown that there is possibility of a denial-of-service attack which can disrupt the website completely. Combined, these flaws can compromise user trust as well as customer data and business functionality.

The remediation provided focuses on improving user input filtering as this is where most vulnerabilities occur, specifically for form inputs and file upload. The website must also improve the session management system as it is not fit for purpose as well as adhering to the security principle of least privilege. Rate-limiting must also be implemented to tackle the possibility of denial-of-service attacks as in its current state, it is trivial to flood the server with requests thus causing an outage. Each vulnerability found includes recommendations for best practice.

Contents

1 INTRODUCTION

1.1 BACKGROUND

Many retailers are closing physical stores in favour of online stores. The office for National Statistics in the UK has shown that online sales rose in 2025 for the 8th consecutive year (Office for National Statistics, 2025). This makes it crucial for businesses to own and maintain websites for doing commerce. Much like how physical stores had to invest in security, so too do online stores. Online stores require so much customer data for conducting business such as email addresses, home addresses, payment details and phone numbers. It is paramount that this information is held securely, making security for online stores a greater priority than for their physical counterparts.

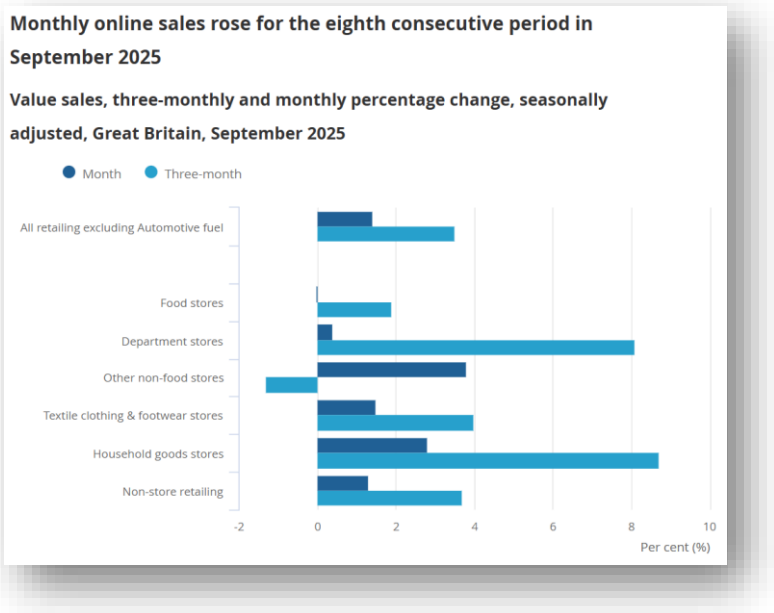

Figure 1 - Chart Displaying Percentage Change in Online Sale in the UK

All of this must be considered, especially since small to medium-sized businesses are being actively targeted. The UK government reports that 58% of small businesses reported having experienced any kind of cyber security attack (Department for Science, Innovation & Technology, 2025). The consequences for a business failing to uphold cyber security can be dire when customer data, trust and business functionality are on the line. For small businesses this can be entirely devastating, therefore, it is crucial that businesses such as Rickstore commission penetration tests to keep up on security.

The OWASP foundation's web security testing guide (OWASP Foundation, 2021) contains advice compiled by leading experts on how websites can be comprehensively tested. By following this standard, it is certain

that any website penetration test is thorough. This covers everything from information gathering through to testing user inputs, which creates assurance that after the penetration test the website has been tested comprehensively.

Once these vulnerabilities have been found, they must be scored by priority so that penetration test results are easily understandable for non-technical readers which is where the CVSS 3.1 framework comes in (FIRST, 2019). This framework grades vulnerabilities with a scoring system ranging from 0.1 – 10 by considering a variety of factors such as the following:

- Attack Vector: What kind of access is required to perform this attack?
- Attack Complexity: Is this a difficult attack to pull off?
- Privileges Required: Is any kind of authorization required to pull off this attack?
- User Interaction: Does this attack require any other user's input?

These make up the "Exploitability Factors" for a CVSS score to describe how viable a vulnerability is to execute. Along with this, the "Impact Factors" must be considered for how data could be affected. These are based on the security triad as shown below:

- Confidentiality: Can sensitive data be exposed?
- Integrity: Can attackers modify or delete important information?
- Availability: Can this cause a denial of access to data?

CVSS 3.1 will be used as opposed to the newer CVSS v4.0 as the former is simpler, which for the purposes of this report will be sufficient. Additionally, v3 includes an optional temporal score and environmental score which will also be excluded.

1.2 AIMS

The aim of this penetration test is to methodically identify security vulnerabilities present on the Rickstore website which could be used to compromise customer data or business operation. The report consists of three primary aims with sub objectives as shown:

1.2.1 Comprehensively Test Website Security Following a Systematic Industry Standard Approach

Execute a structured penetration test with a modified version of the OWASP web security testing guide V4.2, (OWASP Foundation, 2021) which excludes unnecessary tests which don't apply and merges sections with a large amount of overlap.

Sub Aims

- Execute the 6 main phases of the penetration test:
    - Enumeration
    - Configuration and Deployment Management Testing
    - Identity Management Testing
    - Authentication Testing
    - Session Management Testing
    - Input Validation Testing
- Utilize at least one tool recommended by OWASP in the testing tool resource (OWASP, 2020) per phase.
- Accurately document the process so that results can be reproduced. Evidence in the form of:
    - Screenshots
    - Code Snippets
    - Clear Instructions

1.2.2 Accurately Assess the Findings to Prioritize Identified Vulnerabilities

Once the vulnerabilities have been identified, they must be put into a list in terms of severity.

Sub Aims

- Classify vulnerability findings using the CVSS V3.1 framework
    - Calculate base scores using the CVSS V3.1 calculator available on the NIST website
- Order vulnerabilities in order of score

1.2.3 Provide Accurate Remediation Guidance

So that the business can improve cyber security in the future, it's important to provide instructions on how this can be done:

Sub aims

- Provide specific implementation solutions for each identified vulnerability
    - Explain why the remediation provided is effective

- Use code snippets and references to support instructions to support remediation instructions

2 METHODOLOGY

2.1 METHODOLOGY OVERVIEW

The Rickstore online store is a simple website built upon HTML, CSS, JavaScript and PHP with no CMS in use. Additionally, the website is not fully functional, and the scope of testing must take these factors into consideration.

The methodology for this penetration test follows a modified version of the OWASP Web Security Testing Guide. For this specific website, not all steps should be included, and some should be merged for brevity. For this reason, the following sections were cut since they were not relevant/applicable or were out of scope for this specific website.

Table 1 - Table showing why certain OWASP WSTG sections were cut

OWASP WSTG Section	Reason for Removal
Conduct Search Engine Discovery Reconnaissance for Information Leakage	Not applicable here
Fingerprint Web Application Framework	Not applicable here
Fingerprint Web Application	Too complex for this case
Map Application Architecture	Too complex for this case
Test RIA Cross Domain Policy	Not applicable here
Test for Subdomain Takeover	Not applicable here
Test Cloud Storage	Not applicable here
Testing for Vulnerable Remember Password	Not applicable here
Testing for Weaker Authentication in Alternative Channel	Not applicable here
Testing for Session Fixation	Too complex for this case
Testing for Session Puzzling	Too complex for this case
Testing for HTTP Parameter Pollution	Too complex for this case
Testing for LDAP Injection	Not applicable Here
Testing for XML Injection	Not applicable here
Testing for XPath Injection	Not applicable here
Testing for IMAP SMTP Injection	Not applicable here
Testing for Command Injection	Not applicable here
Testing for Format String Injection	Not applicable here
Testing for Incubated Vulnerability	Not applicable here
Testing for HTTP Splitting Smuggling	Not applicable here
Testing for HTTP Incoming Requests	Not applicable here
Testing for Host Header Injection	Not applicable here
Testing for Server-side Template Injection	Not applicable here
Testing for Server-Side Request Forgery	Too complex for this case
Testing for Stack Traces	Not applicable here
Business Logic Testing	Not hugely applicable since the website has functionality issues however file upload

	validation and test of malicious files was performed in section 3.7
Client-Side Testing	Too complex for this case
API Testing	Not applicable here

Sections from the OWASP methodology were merged with others or put under other sections. This is because some topics are similar enough to be covered at once. For example, since the Nmap scan fingerprints the webserver and automatically tests for metafiles such as robots.txt, it's more readable to merge these two sections.

The result is a methodology that is specifically designed for this website, is comprehensive and backed by the expertise of industry experts and is better known to the tester.

3 PROCEDURE

3.1 OVERVIEW OF PROCEDURE

The penetration test methodology was strongly based on the OWASP standard for Web application Security testing standard (OWASP Foundation, 2021) which is the industry standard for web-application penetration testing compiled by field experts. Some sections from the original methodology have been removed, merged or modified as they are not relevant or in scope for this case, such as testing for subdomain takeovers, as this web application does not use subdomains.

The methodology underwent the following main phases:

- Enumeration
- Configuration and Deployment Testing
- Identity Management Testing
- Authentication Testing
- Session Management Testing
- Input Validation Testing

These phases comprehensively test the most common and critical attack vectors which an adversary would use. These tests were performed with industry standard tools which include but are not limited to:

- Nmap 7.94 (Lyon, 2023)
- Burpsuite (PortSwigger Ltd, 2025)
- Wappalyzer 6.10.86 (Wappalyzer, 2025)
- Gobuster 3.6 (Reeves, 2025)
- Curl 8.14.1 (Stenberg, 2025)
- Wireshark 4.0.17 (Wireshark Foundation, 2025)
- Mozilla Firefox Developer Tools 140.4esr
- Netcat 1.219-1
- SQLMap 1.8.12 (Damele and Stampar, no date)
- Zap 2.15.1 (OWASP Foundation, 2025)

This shows that the methodology performed was comprehensive and able to find all the most common attack vectors that require immediate attention.

3.2 ENUMERATION

3.2.1 Nmap Scan

Nmap, which is the industry standard for network discovery and port scanning, was used to scan the network. Initially a scan is run as root to reveal what ports are open. Scanning as root defaults Nmap to using a SYN stealth scan which sends out SYN packets without completing the handshake, this is faster and stealthier than scanning typically.

```
┌─[root@parrot]─[/home/kali]
└──╼ nmap 192.168.1.10
```

Figure 2 - Basic Nmap Scan

Then using another scan, the ports found can be further enumerated with the -sV and -sC flags. In this case, the IP address is hosting a web application with open ports for FTP and MySql.

```
┌─[root@parrot]─[~]
└──╼ $nmap -p21,80,3306 -sV -sC 192.168.1.10
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-10-03 11:33 BST
Nmap scan report for 192.168.1.10
Host is up (0.00083s latency).
Not shown: 997 closed tcp ports (conn-refused)
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)
|_http-title:  RickStore Groups
|_http-server-header: Apache/2.4.3 (Unix) PHP/5.4.7
| http-robots.txt: 1 disallowed entry
|_/info.php
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
3306/tcp open  mysql   MySQL (unauthorized)
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 6.77 seconds
```

Figure 3 - Nmap version and script scan output

Nmap found that the robots.txt file contained an entry for the phpinfo.php page. Which contains configuration data for PHP running on the website.

Another scan using the -sU flag can be performed to ensure there are no open ports on UDP. Since UDP scans take much longer, the -F flag is used to check only the top 100 most common ports. This scan returned nothing confirming that there is more than likely no UDP service running.

```
┌─[root@parrot]─[/home/kali]
└── sudo nmap -sU -F  192.168.1.10
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-10-13 22:35 BST
Nmap scan report for 192.168.1.10
Host is up (0.00024s latency).
All 100 scanned ports on 192.168.1.10 are in ignored states.
Not shown: 100 closed udp ports (port-unreach)
MAC Address: 00:0C:29:7C:B2:51 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 98.41 seconds
```

Figure 4 - Nmap UDP scan output

3.2.2 Enumerating technologies with Wappalyzer

Wappalyzer can fingerprint the website to find running technologies including service versions. The results show the use of typically found Apache uses version 2.4.3 and PHP uses 5.4.7 as shown in the screenshot below.

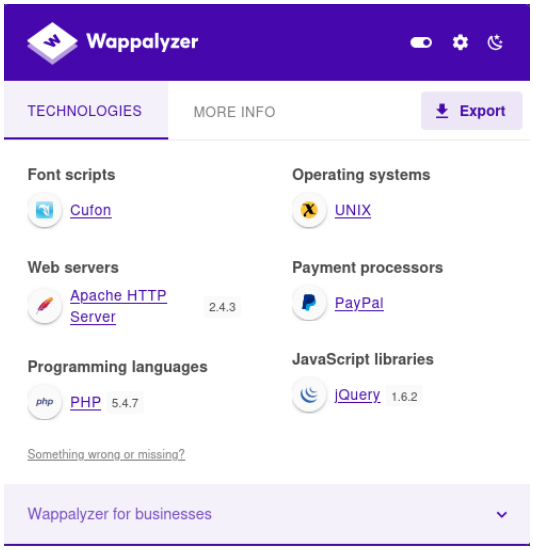

Figure 5 – Wappalyzer Output

3.2.3 Identify Application Entry Points

The application was spidered using Zap and manual searching to return the following input points:

- Login form (login.php)
- Admin login form (login.php)
- Customer Registration (customer.php)
- Contact form (contact.php)
- Profile management (profile.php)
- Password reset (changepassword.php)
- Checkout process (process.php)
- Adding to cart (index.php)
- Checking out basket (view_cart.php)

The key ZAP scan findings can be found in Appendix A.

3.3 Configuration and Deployment Management Testing

3.3.1 Reviewing Webserver Metadata and Content for Information Leakage

Gobuster (Reeves, 2025) uses a medium sized word list to search websites for existing content such as directories or files, the scan returned the results as shown in the figure below.

```
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.10
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/images               (Status: 301) [Size: 235] [--> http://192.168.1.10/images/]
/admin                (Status: 301) [Size: 234] [--> http://192.168.1.10/admin/]
/audio                (Status: 301) [Size: 234] [--> http://192.168.1.10/audio/]
/pictures             (Status: 301) [Size: 237] [--> http://192.168.1.10/pictures/]
/bin                  (Status: 301) [Size: 232] [--> http://192.168.1.10/bin/]
/css                  (Status: 301) [Size: 232] [--> http://192.168.1.10/css/]
/js                   (Status: 301) [Size: 231] [--> http://192.168.1.10/js/]
/fonts                (Status: 301) [Size: 234] [--> http://192.168.1.10/fonts/]
/font                 (Status: 301) [Size: 233] [--> http://192.168.1.10/font/]
/phpmyadmin           (Status: 401) [Size: 1222]
Progress: 220560 / 220561 (100.00%)
===============================================
```

Figure 6 – Gobuster Output

The directory listings were further investigated and found to contain files used for the website which were accessible anyway, apart from a database backup file which could not be downloaded.

The phpMyAdmin page prompted for authentication, login attempts with default credentials were attempted but access could not be gained.

3.3.2 Searching HTML Source code

All the pages and .js files were manually checked for any leftover comments. The products.php page contains the following comment at the top of the HTML source code:

```
*** Note document root is /mnt/sda2/swag/output/vulnerable/site. Tidy
this up later.
```

Figure 7- Leftover HTML Code

Other leftover comments found did not disclose important information.

3.3.3 Test HTTP Methods

The Nmap tool allows for the creation of scripts that can be used to test various services throughout a network. Specific scripts can be used with the "-- script" flag. To test the HTTP Methods available, the "http-methods" script can be used on port 80 to automate the process as shown below:

```
┌─[kali@parrot]─[~]
└──$ nmap -p 80 --script http-methods 192.168.1.10
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-10-23 20:13 BST
Nmap scan report for 192.168.1.10
Host is up (0.00035s latency).

PORT   STATE SERVICE
80/tcp open  http
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS

Nmap done: 1 IP address (1 host up) scanned in 0.21 seconds
```

Figure 8 – Nmap HTTP Method Scan Output

It was found that the HTTP methods available for a user were secure.

3.3.4 Test HTTP Strict Transport Security

The application does not implement HTTPS/TLS encryption. All traffic including authentication credentials, session cookies, and personal data form inputs are transmitted in clear text over HTTP.

3.4 IDENTITY MANAGEMENT TESTING

3.4.1 Test Role Definitions

The website defines two user roles:

Administrator

- Has access to the Administrator panel
- Must log in through the admin login form
- Can create customer/administrator accounts
- Can add/modify existing products in the store

Customer

- Can register account
- Can login
- Can manage their own details and upload profile picture
- Can create support tickets

It is crucial to note that due to poor authorization by the web application, a customer does not have to be logged in or registered to execute the same functions as a logged in one.

Website users are identified using the "secret cookie" cookie. This is insecure as the secret cookie can be easily decoded and manipulated on the client-side.

3.4.2 Test User Registration Process

The /customer.php page was used to test the user registration process. The following issues were found:

- The input form for email address is not filtered so users can use any text without any domain such as "test". Because the user email address is shown on the profile page, testing an "email" such as `<script>alert(1)</script>` reveals that the email address for an account can be used for cross site scripting.
- Capturing a request with Burp-Suite and sending it repeatedly also showed that there is no rate-limiting in place which makes account registration an avenue for a Denial-of-Service attack.
- It was found that the web application does not properly check if an email address already exists, allowing for multiple accounts under one email address. This means that the password reset function can affect the wrong account if used, meaning user data is not safe and can be accessed by others.

3.4.3 Testing for Account Enumeration and Guessable User Account

Manual testing of the login form showed that the website discloses if a login attempt contains the incorrect login or password. Curl, a tool for interfacing websites using a command line, is used to prove this and the command used is broken down in the bullet points below:

- -X: This tells curl that a POST request is to be sent, as typically done for a form.
- -d: Specifies the parameters to be used. The username parameter is magaca and will contain a non-existing user to prove the vulnerability exists. The password which is part of the furaha parameter does not matter in this case so “test” is used.
- -v: Gives a verbose output for additional information.
- | grep “alert”: This takes the output from curl and looks for the alert which discloses that the username attempted does not exist in the database.

This command is first run with a username that does not exist in the database and then run with a username that we know does. The difference in response will show if a username disclosure vulnerability exists. The code snippet below shows the output from the command with a non-existing username.

```
┌──[kali@parrot]─[~]
└─── $curl -X POST http://192.168.1.10/userValidate.php   -d
"magaca=nouserhere&furaha=test&submit=+Login"   -v | grep "alert"
<script language="javascript">alert ("Username not found");window.history.back();</script>
```

Figure 9 – Curl output showing user existence disclosure vulnerability

And below is shown an attempt with an existing username:

```
┌─[kali@parrot]─[~]
└─── $curl -X POST http://192.168.1.10/userValidate.php   -d
"magaca=hacklab@hacklab.com&furaha=test&submit=+Login"   -v | grep "alert"
```

Figure 10 – Curl output comparing existing user to non-existing user

These code snippets prove the existence of the username disclosure vulnerability on the customer login page. The same command was attempted for the admin login page however the vulnerability did not exist.

3.5 AUTHENTICATION TESTING

3.5.1 Testing for Credentials Transported over an Encrypted Channel

The website uses HTTP 1.1 which sends packets to the website in plain text, these can be intercepted by an adversary on the same network by using a tool like Wireshark.

Wireshark is a tool that lists all packets sent over a network interface. This includes any communication with a website or service that uses the internet. In this case, the packets are readable as they are not end to end encrypted and make sensitive data easily readable as shown below:

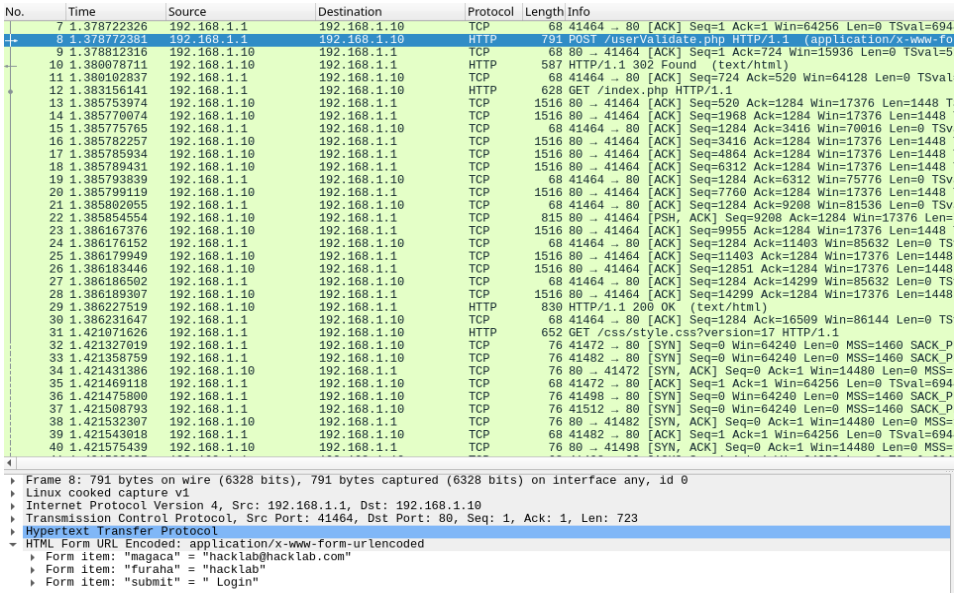

Figure 11 – Wireshark Screenshot Showing Captured Packet with Plain-Text Credentials

The packet can be read without any obfuscation or decryption. In this case it contains the details of a user login attempt.

3.5.2 Testing for Weak Credentials

Since it was possible to dump the database and look at the unencrypted passwords, the admin account credentials were checked to analyze the credentials in use.

Table 2 – Table containing all admin credentials

Username	Password
admin	chat
testadmin	testadmin

The passwords found are extremely weak as they are both available as part of the rockyou.txt wordlist, which is often used by attackers, especially on websites with no rate-limiting. Furthermore, testadmin's password is the same as the username which is easily guessable for an attacker.

3.5.3 Testing for Weak Lock Out Mechanism

To test if administrator passwords can be brute forced, the tool Hydra was used. Hydra is a tool for going through a list and testing credentials. In this case, the default username admin was guessed to exist, and the following flags were applied to test the viability of a bruteforce attack.

- -l : The username to use for bruteforce attempts. In this case, “admin”
- -P: The password list to be used, in this case rockyou.txt. This word list contains most weak credentials that are used and is quite comprehensive at over a million different passwords.
- -T: The number of threads to use, 16 is the maximum amount and makes the process quicker
- -V: A verbose output gives more information for debugging and testing purposes
- -I: Ignores previous attempts and starts a new attack

These flags were used as part of the command as shown in the code snippet below:

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt 192.168.1.10 http-post-form
"/employeeValidate.php:magaca=^USER^&furaha=^PASS^:F=Invalid" -V  -T 16 -I
```

Figure 12 – The hydra command used to bruteforce administrator credentials

This ran for some time and provided the following result:

```
[ATTEMPT] target 192.168.1.10 - login "admin" - pass "chata7" - 52378 of 14344399 [child 13]
(0/0)
[ATTEMPT] target 192.168.1.10 - login "admin" - pass "chata69" - 52379 of 14344399 [child 6]
(0/0)
[ATTEMPT] target 192.168.1.10 - login "admin" - pass "chata07" - 52380 of 14344399 [child 3]
(0/0)
[80][http-post-form] host: 192.168.1.10   login: admin   password: chat
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-11-11 17:32:26
```

Figure 13 – Hydra outputs showing brute forced password “chat”

This shows that weak credentials are in use, which makes the administrator panel extremely insecure. Furthermore, after 50,000 requests some form of rate-limiting should have kicked in to stop further requests, however, this was not the case. This shows that the administrator login

panel is not secure and that the website suffers from a lack of rate-limiting, making it prone to denial-of-service attacks.

3.5.4 Testing for Bypassing Authentication Schema

The admin page was tested to see if it could be accessed by an unauthenticated user. During testing with the browser, it was shown to redirect to the index.php page. Curl was used to fetch the admin page to see the response from the website as shown in the code snippet below.

```
curl http://192.168.1.10/admin/ -v
```

Figure 14 – Curl Command used for Authentication Bypass Vulnerability

Since curl only returns the first response from the webserver, it returned the admin page before the redirect could occur. A similar result could be obtained with the use of Burp-Suite.

The admin page reveals the functions that an admin has. These are executed with the use of various .php pages, so using curl, these functions were used to see if they contained any sort of authentication.

The curl command to test this goes into the admin panel and uses the “employeereports.php” function. This returned a pdf file which was not readable with curl, so the “--output” flag was used to save the response to a file which could be opened.

```
┌─[kali@parrot]─[~]
└──╼ $curl http://192.168.1.10/admin/EmployeeReports.php --output file.pdf -v
```

Figure 15 – Curl Command used to Generate Employee Report with No Authentication

The file can be browsed to and then opened using a pdf viewer, providing the result shown below:

SOMSTORE EMPLOYEE REPORS
EMPLOYEE REPORTS
Friday, September 26, 2025

Employee Record: 2

Employee ID	Employee Full Name	EMPLOYEE USER NAME
52	Mr Admin	admin
53	testadmin	testadmin

Figure 16 – Employee Report Retrieved Through Vulnerability

Using the provided hacklab@hacklab.com account the reset password feature was tested by attempting to change the password to "1". Furthermore, the password "1" was tested during account registration and was accepted. This proves no password policy is in place.

3.5.5 Testing for Weak Password Change or Reset Functionalities

The method used for password changing is insecure since the user email is passed through as a parameter rather than cookie. This allows for the password reset request to be modified to any user, and the current password parameter brute-forced to change the password. Furthermore, due to an existing cross site scripting vulnerability in the form, the current password parameter can be escaped allowing for changing the password of any known user on the database as shown on the figure below:

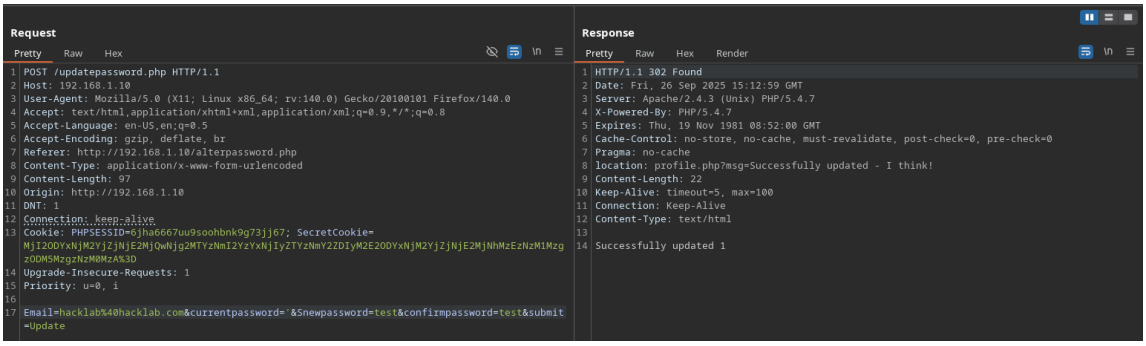

Figure 17 – Burp-Suite Output showing currentpassword parameter bypass vulnerability

3.6 SESSION MANAGEMENT TESTING

3.6.1 Testing for Session Management Schema

Browser developer tools were used to analyze the Session Management system. This resulted in the finding that although the website assigns a PHP session ID, it uses its own secret cookie to manage sessions as shown below:

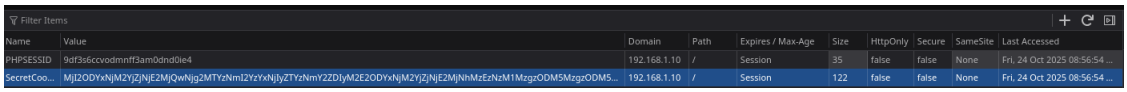

Figure 18 – Screenshot of Browser Developer Tools Displaying Cookies in Use

The secret cookie is set only during login, so when a user registers, they are not automatically logged in. The SecretCookie string was identified as a base64 string. This was decoded using Burp-Suite's decoder tool. The resulting string was then identified as being hex, which was decoded to reveal the cookie plain-text as shown below.

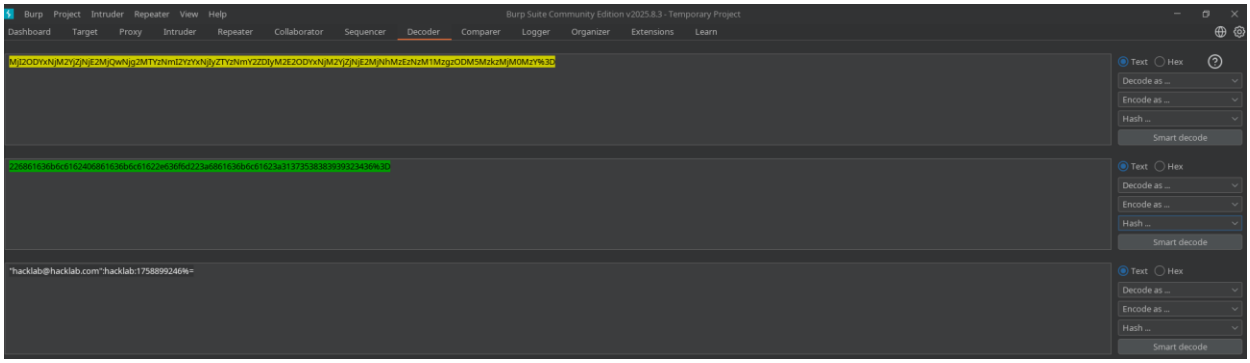

Figure 19 – Burp-Suite Decoder Output Showing Decoded SecretCookie

The cookie consists of email and password as well as a unix timestamp which makes it predictable.

3.6.2 Testing for Cookie Attributes

Browser tools also found that the SecretCookie cookie also holds the following insecure attributes as shown in figure and listed below:

- HttpOnly: Because this is set to false, user cookies are vulnerable in case of an XSS attack, and the cookie can be read and then decoded easily.
- Secure: Because this attribute is set to false, cookies can be sent over HTTP instead of HTTPS/TLS which is encrypted. This means that cookies can be read unencrypted over the network, similarly to the demonstration in section 4.4.1.
- Max-Age: Max-Age is set to session which seems to break website functionality as sometimes the user is logged out for no reason. This means that a stolen cookie cannot be used by someone else, however because it can be decoded an attacker can use the credentials to log in.

3.6.3 Testing for Exposed Session Variables

All pages were checked for exposed session parameters using Zap and manual testing. No such vulnerability was found, making the website safe from Insecure Direct Object Reference vulnerabilities. Testing for Cross Site Request Forgery

3.6.4 Testing for Cross Site Request Forgery

The web application does not use CSRF Tokens, so requests can be made by anyone from anywhere. This is demonstrated in section 3.5.4 where admin panel functions could be executed from curl.

3.6.5 Testing for Logout Functionality

The test account was logged out from and then the browser back button was used, this is to test that the session is properly terminated. In this case it was proven that account resources became unavailable until the account was logged back in to show that the session is adequately ended.

3.6.6 Testing for Session Hijacking

As shown in other sections, session hijacking is possible through many different methods:

- Man in the Middle Attack on the same network as the web-app does not use HTTPS.
- Stored XSS payloads can fetch user cookies since the HTTPOnly flag is set to false, so user cookies can be sent to an attacker's IP.
- The "SecretCookie" used by the website is predictable as it follows a username:password:timestamp format.

3.7 INPUT VALIDATION TESTING

3.7.1 Testing for Cross Site Scripting

Zap was used to find points where XSS is possible. These findings were confirmed with manual testing in this section, and all form inputs were also tested manually in case Zap missed anything.

To manually test for XSS, first it would be observed if any user input was displayed on the website. If it was, then it would try to be escaped by using special characters such as: <,",>. After this special character the "alert("xss")" payload would be used to confirm the finding.

For inputs which do not reflect user input, the standard apostrophe and greater than symbol payloads would be used. In this case, for this website, this escaped the input and proved a XSS vulnerability.

Table 3– Table Containing XSS Manual and Automated Testing Results

Page	XSS Vulnerable?	Notes
login.php	No	
customer.php	Stored XSS (All Parameters)	The (') symbol allows for payloads to be placed after for XSS. When the customer details are viewed from the Admin panel, these can escape and lead to cookie theft or other XSS vulnerabilities.

process.php	Stored XSS (All Parameters)	The (') symbol allows for payloads to be placed after for XSS. When orders are viewed from the admin panel, parameters can escape. This can lead to cookie theft or other XSS vulnerabilities.
Contact.php	Reflected XSS (name parameter) And Possibly Stored	Name parameter is reflected on the thank you page after submitting form. This allows for reflected XSS and likely stored XSS when ticket is opened by admin.
profile.php	Reflected & Stored XSS (All Parameters)	The string "> Allows for escaping form inputs including setting input titles which are executed when the profile page is viewed.
changepassword.php	Reflected XSS	XSS Vulnerability in currentpassword parameter allows for bypassing checks to ensure password is correct. This means that any user account can have the password reset without authentication by editing the POST Request. More details available in section 2.2.1.3.

The contact form reflects the name used in the form in the url as such:

```
http://192.168.1.10/thankyou.php?id=user
```

This was found to be an avenue for cross site scripting, as the id parameter would be reflected on the website as shown in the figure below:

3.7.2 Testing for SQL Injection

To determine if a parameter was SQL injection Vulnerable, a POST request was first captured with burp-suite. The input parameters would be tested by attempting to input an apostrophe one by one. This would get around client-side restrictions on inputs such as emails. If an error was returned it would be clear that SQL injection is possible.

To ensure thorough testing, ZAP also checked inputs for SQLi, these findings would be manually tested.

The table below shows the output of each page tried.

Table 4– Table containing SQL Injection Manual and Automated Testing Results

Page	SQLi Vulnerable?	Payload
login.php	No	N/A
customer.php	All Parameters, Error Based	'
process.php	All Parameters, Error Based	'
Contact.php	All Parameters, Error Based	'
profile.php	No	N/A
changepassword.php	Email Parameter Error Based	'

To further test the extent of the damage resulting from SQLi, SQLmap would be used to navigate the database. The request captured with burp-suite can be used as an input for SQLmap, and since enumeration revealed the web-app back-end is running MySQL, a flag can be used to save time and ensure correct payloads are being attempted.

```
sqlmap -r contact.txt -p email --dbms=MySQL --level=3 --risk=3 -D somstore --tables
```

Figure 20 – SQLMap Command Used to Dump Database Contents

This made all the database information available, notably the employee and customer details which were unencrypted and readable as shown below:

```
Database: somstore
Table: employee
[2 entries]
+-------------+-------------+-----------+-----------+---------------+
| Employee_ID | Picture     | Password  | Username  | Employee_Name |
+-------------+-------------+-----------+-----------+---------------+
| 52          | jananka.jpg | chat      | admin     | Mr Admin      |
| 53          | jananka.jpg | testadmin | testadmin | testadmin     |
+-------------+-------------+-----------+-----------+---------------+
```

Figure 21 – SQLMap Output Containing Dumped Employee Table

3.7.3 Testing for Local File Inclusion

Zap revealed the existence of a Local File Inclusion vulnerability on the attachment.php page with the "type" parameter as was manually confirmed in the screenshot below:

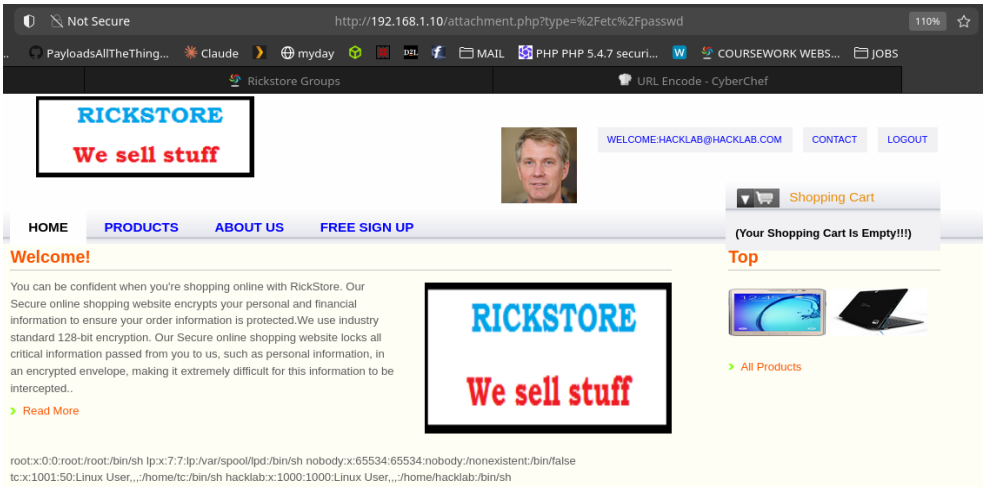

Figure 22– Screenshot of the attachment.php Page Showing File Inclusion Vulnerability

Any error on the website causes browser to redirect to this page, allowing the user to click the hyperlink to go back to the page that caused the error as shown below.

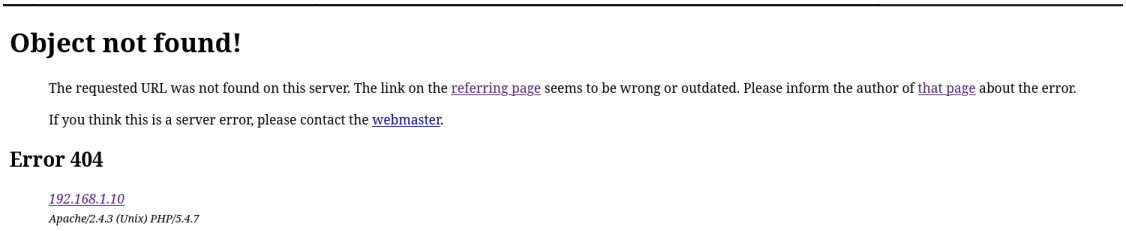

Figure 23 – Screenshot Showing 404 Error that Discloses Service Version Information

The error page discloses the service versions running on the web application which could make it easier for an attacker to enumerate.

3.7.4 Test Upload of Unexpected File Types

The website allows file upload in the form of a profile picture, this input is filtered as demonstrated in the image below, where the upload of a .php file is forbidden.

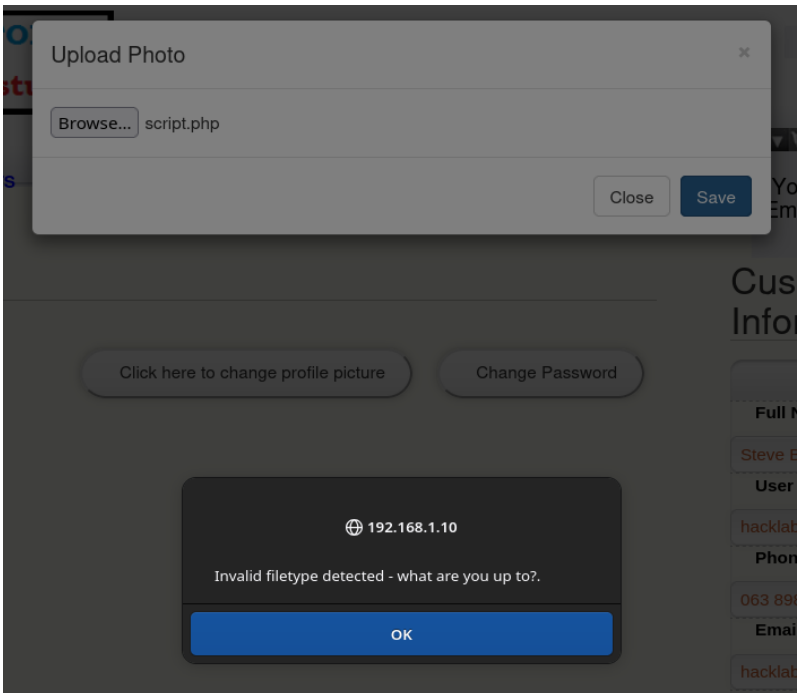

Figure 24 – Screenshot Showing Unsuccessful Attempt to Upload PHP file

To test if the filtering only occurred on the client-side, the POST request for the file upload was captured with Burp Suite (PortSwigger Ltd, 2025). This allowed for modification of the "Content-Type" parameter in the file upload functionality. This was adjusted to "image/jpeg" which is the string given to image files, and a PHP reverse shell was created and uploaded.

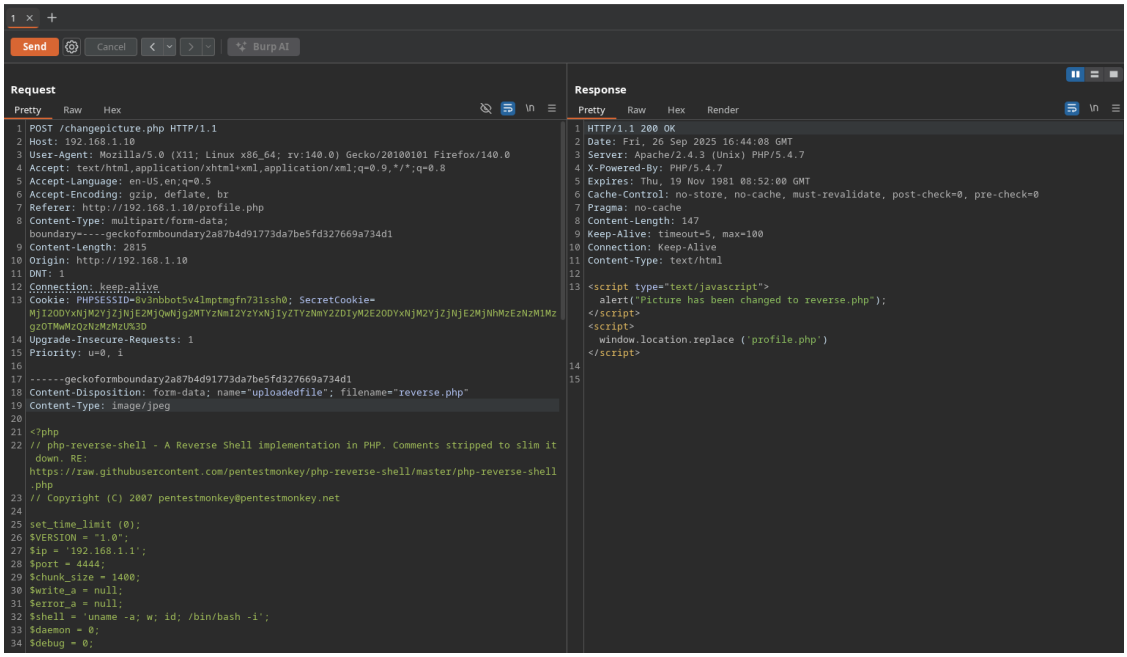

Figure 25– Burp-Suite Screenshot Showing Successful Upload of PHP file with Modified Content-Type Header

The response from the server implied that the profile picture was set to the reverse shell, demonstrating that the file upload filter can be bypassed.

3.7.5 Test Upload of Malicious Files

Since it was possible to upload PHP code as the profile picture, it would be executed just by loading the index page while logged in. A local listener was set up using netcat, the flags used were used for the following reasons:

- Listen (l) – Makes netcat look out for incoming connections instead of initiating them
- Verbose (V) – Provides extra details about the connection such as the IP address
- Numeric (N) – Skips DNS resolution. In our case this is appropriate as 192.168.1.10 is a local IP
- Port (P) – This specifies the port netcat is listening on. Because the reverse shell is using port 4444, netcat is set to the same port. Notably, any port under 1000 requires root and therefore cannot be used.

Once the page is refreshed, the website tries to use reverse.php as a profile picture. This loads the file which executes the code and creates a connection to the attacking machine, creating a terminal session as the user “nobody”.

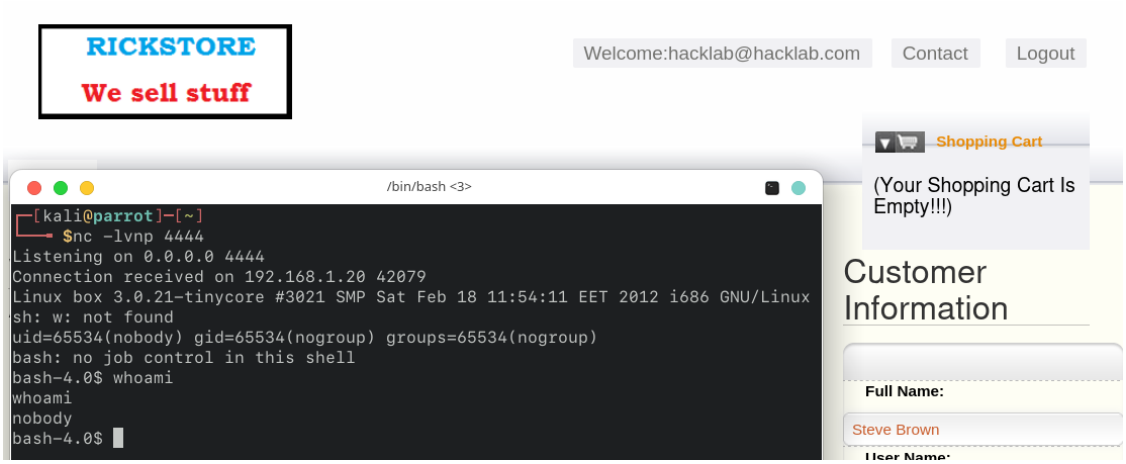

Figure 26 – Terminal Output Showing Successful NetCat Connection

Privileges can be further escalated by using the sudo -l command. This command shows what sudo commands are available to the “nobody” user as shown in the screenshot below:

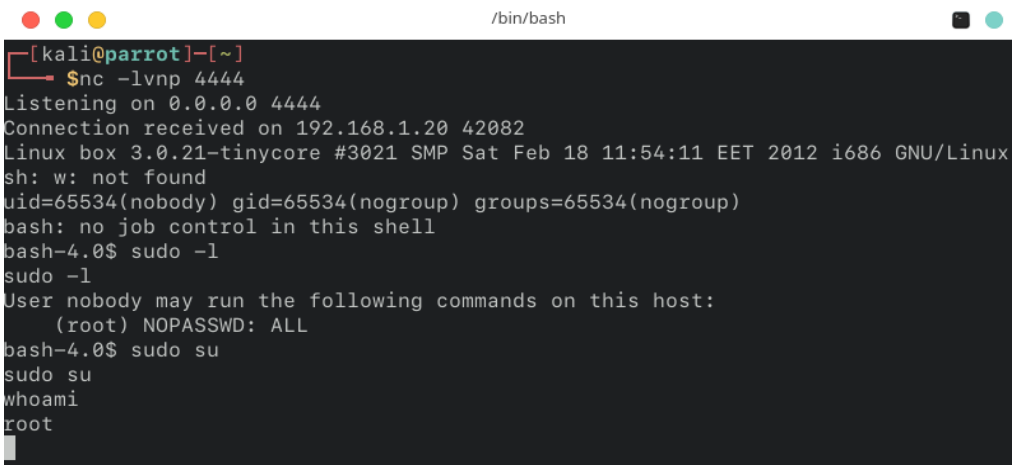

Figure 27 – Terminal Screenshot Showing Established Root Shell on the Website

The command returns that root can be used with no password. Executing the command “sudo su” gives the nobody user, and in this case attacker, full access to the system.

4 RESULTS

4.1 RESULTS SUMMARY

The vulnerabilities found are as shown in the table below:

Table 5 - Results Summary Displaying Existing Vulnerabilities

Severity	CVSS Range	Count	Key Vulnerabilities
Critical	9.0-10.0	3	• RCE + Root Shell • SQL Injection • Admin Panel Bypass
High	7.0-8.9	3	• Password Reset Bypass • XSS (Stored and Reflected) • Local File Inclusion
Medium	4.0-6.9	7	• No HTTPS • Weak Session Management • No Rate-Limiting/DoS Vulnerabilities
Low	0.1-3.9	3	• Directory Browsing • Information Disclosure • Outdated PHP Version

There was a total of 16 vulnerabilities present on the RickStore website. Many serious vulnerabilities were discovered including the possibility of remote code execution as root on the website server. The vulnerabilities found allow an adversary to:

- Gain full control over customer and business data
- Modify and delete existing products on the website
- Establish backdoors for long term access
- Attack the network that the server is running on, increasing the scope of a potential attack

Overall, the security of the Rickstore website is extremely poor and should be improved with stronger user input filtering, stronger authentication and session management. Vulnerabilities are listed from most to least critical and include remediation for how each can be addressed.

4.2 CRITICAL VULNERABILITIES

4.2.1 Critical - File Upload Remote Code Execution Vulnerability

Vulnerability Class: Improper Input Validation / Broken Access Control

CVSS 3.1 Base Score: 9.0 (Critical)

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

Reference: Section 3.7.5

4.2.1.1 Description

Because file inputs are only filtered on the client side, the security mechanism for this feature is easily bypassed with the use Burp-Suite. Furthermore, the system user running the website is granted too many privileges and has access to the “sudo -l” command with no password which compromises the system and threatens the network.

4.2.1.2 Impact

This vulnerability allows for root control over the device running the server and increases the scope of attack to the network that the server is attached to. This represents the most critical security failure identified in this penetration test. Unauthenticated attackers can achieve complete system compromise, including:

- Remote code execution as 'nobody' user
- Immediate privilege escalation to root via password-less sudo
- Full control over web server, database, and hosted data
- Potential for lateral movement across the network infrastructure
- Ability to establish persistent backdoors

4.2.1.3 Remediation

To avoid unrestricted file upload, server side verification should be implemented. This could be done with the use of PHP’s getimagesize() function which checks actual file content (OWASP Foundation, 2021).

Additionally, the user running the web services should contain the only permissions needed, following the principle of least privilege. This would make privilege escalation in the event of a reverse shell much more unlikely.

Finally, there must be a file size limit put in place to avoid the possibility of a large file being uploaded, taking up extra storage space.

4.2.2 Critical - Admin Panel Access Control Vulnerability

Vulnerability Class: Broken Access Control

CVSS 3.1 Base Score: 9.8 (Critical)

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

Reference: Section 3.5.4

4.2.2.1 Description

A critical exploit was found in the admin panel, which loads the page before it verifies if a user is authenticated. This means that with the use or curl or Burp-Suite, the admin page and its functions can be read. This can be further exploited as the functions of the admin page do not check authentication and allows for pages such as “employeereport.php” to be used for fetching employee credentials.

4.2.2.2 Impact

This allows for complete control over admin page functions such as viewing existing customers and admin accounts. Creating, deleting and modifying products as well as viewing and deleting existing orders.

4.2.2.3 Remediation

All requests to the admin panel must be authenticated before loading any content. If the user cannot be authenticated, a 403 Forbidden error message or a redirect should be returned instead. Furthermore, **all** admin panel functions should verify account credentials to protect against unauthorized requests.

4.2.3 Critical -- SQL Injection Vulnerabilities

Vulnerability Class: Injection

CVSS 3.1 Base Score: 9.8 (Critical)

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

Reference: Section 3.7.2

4.2.3.1 Description

4 user form input points were found to contain an SQLi vulnerability, namely the pages:

- Customer.php
- Process.php
- Contact.php
- Changepassword.php

Manual testing found that the SQL query could be escaped with a simple apostrophe payload due to poor user input filtering.

4.2.3.2 Impact

As confirmed with SQLmap, this exposes all business data to an adversary, allowing them to modify, add or remove data. This includes data such as customer details entered during registration, employee credentials and product details.

4.2.3.3 Remediation

The SQLi found on the website is critical as it allows for complete control over the database and compromises all employee, business and customer data. This could be solved with the use of prepared statements in PHP (The PHP Documentation Group, 2025) as such:

```php
$stmt = $conn->prepare("INSERT INTO customers (name, email) VALUES (?, ?)");
$stmt->bind_param("ss", $name, $email);
$stmt->execute();
```

Figure 28 – PHP Prepared Statement

This way, the parameter cannot be escaped, paired with filtering characters which are not used in form inputs, SQL injection would not be possible in forms. Better error handling would further mitigate the change of SQLi, where raw SQL errors would not be shown to the user.

Lastly, user data stored on the database should be encrypted. This would provide a fallback in case of an exposed database where user data would not be available. PHP's password_hash() function would effectively encrypt user password and can be easily integrated into the DBMS.

4.3 HIGH SEVERITY VULNERABILITIES

4.3.1 High - Reset Password Bypass Vulnerability

Vulnerability Class: Improper Input Validation / Broken Authentication

CVSS 3.1 Base Score: 8.1 (High)

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

Reference: Section 3.5.5

4.3.1.1 Description

The password reset function uses a form with an email input. The parameters are not filtered properly, which provides an injection point for XSS, this allows for the current password field to be skipped and a new password to be applied. If the request is captured with burp-suite is captured, this allows for modifying any user account password.

4.3.1.2 Impact

This allows for resetting the password of any known username, thus compromising this user's personal information.

4.3.1.3 Remediation

The solution to this would be to use the session cookie to determine the account on which the password change is occurring, as well as filtering the input to address the current password parameter bypass as mentioned in section 3.3.2.3. This would make for a secure password reset function without sending out emails to users.

4.3.2 High -- XSS Vulnerabilities

Vulnerability Class: Injection (XSS)

CVSS 3.1 Base Score: 7.1 (High)

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

Reference: Section 3.7.1

4.3.2.1 Description

Apart from the login form, every user input resulted in reflected or stored XSS. This is due to poor filtering during user input since many of the payloads required just a quotation mark to escape.

4.3.2.2 Impact

The password reset function requires an email, current password and new password. By using a single quotation mark, the current password requirement is ignored. If an attacker crafts a packet with a target email address, the password can be changed, and the account is compromised.

By escaping the process.php form, a <script> payload can be sent to the database table with processed orders. If this is viewed from the administrator panel, cookies can be stolen and account credentials compromised. This would also work with the contact.php form.

Additional attack scenarios include:

- Session hijacking through cookie theft (enabled by HttpOnly flag set to false)
- Phishing attacks through DOM manipulation
- Keylogging to capture sensitive user input
- Defacement of user-facing pages
- Distribution of malware to site visitor

4.3.2.3 Remediation

The poor input validation can be rectified by using the "htmlspecialcharacters" function in PHP for all user input. This would encode all user inputs rendering XSS payloads ineffective. Further action would also require setting the HTTPOnly flag for cookies to true, this would make cookie theft through Javascript unfeasible.

4.3.3 High - Local File Inclusion Vulnerability

Vulnerability Class: Path Traversal

CVSS 3.1 Base Score: 7.5 (High)

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

Reference: Section 3.7.3

4.3.3.1 Description

The local file inclusion vulnerability in the type parameter in attachment.php allows for directory traversal and any file on the system to be read.

4.3.3.2 Impact

Combined with the leftover comment in the index.php page makes it possible for the database configuration file to be read which would compromise all business data.

4.3.3.3 Remediation

This could be mitigated with filtering path traversal characters and their URL-encoded counterparts. The realpath() function in PHP prevents escape, limiting the potential consequences if such a vulnerability is found.

4.4 MEDIUM SEVERITY VULNERABILITIES

4.4.1 Medium - HTTP Vulnerability

Vulnerability Class: Cryptographic Failure

CVSS 3.1 Base Score: 6.5 (Medium)

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

Reference: Section 3.5.1

4.4.1.1 Description

The website uses HTTP 1.1 which is insecure as the communication between user and server is not end to end encrypted.

4.4.1.2 Impact

This allows for MitM attacks for users on the same network as an attacker. Since communication is not encrypted, packets which contain plain text credentials and data can be read.

4.4.1.3 Remediation

HTTPS/TLS encryption should be implemented to make communication of customer information such as login credentials or personal information substantially more secure. A free SSL/TLS certificate can be obtained from services such as “Let’s Encrypt”. Once this has been obtained, all traffic should be redirected to port 443 so that HTTPs is in use.

4.4.2 Medium – Username Enumeration

Vulnerability Class: Information Disclosure

CVSS 3.1 Base Score: 5.3 (Medium)

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

Reference: Section 3.4.3

4.4.2.1 Description

Using curl, it was demonstrated that the user login form would confirm if a username during a login attempt exists.

4.4.2.2 Impact

This makes it possible to enumerate user accounts for a password brute-forcing attack and compromises user privacy by disclosing if an email is tied to an account at Rickstore.

4.4.2.3 Remediation

The response from the login form for both admins and users should be adjusted so that the result is consistent when either a password or username is incorrect. If the response content length is the same, then the opportunity for information disclosure here would be eliminated.

4.4.3 Medium – Weak Session Management

Vulnerability Class: Security Misconfiguration

CVSS 3.1 Base Score: 5.9 (Medium)

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

Reference: Section 3.6.1

4.4.3.1 Description

Currently an attacker has a variety of methods to perform session hijacking:

- MitM Attack to steal credentials/cookie
- Stored XSS payload to steal user cookies
- Creating a "SecretCookie" which follows a Username:Password:Timestamp format.

4.4.3.2 Impact

This could allow an attacker to impersonate a user account through a variety of means and could compromise user details given during registration such as address, phone number and name becomes available.

4.4.3.3 Remediation

The session management system should be overhauled by implementing better cookie obfuscation, fixing existing XSS possibilities as outlined in section 3.3.2 and implementing HTTPS as outlined in section 3.4.2. A way to accomplish this would be to use PHP sessions, which come obfuscated by default. Additionally, user passwords should never be stored in cookies to minimize the attack vector available.

Once this has been overhauled, the sessions can be given a lifetime so that they automatically expire and can be terminated when needed, such as when a user password has been changed or a user has logged out.

4.4.4 Medium -- Insecure Cookie Attributes

Vulnerability Class: Security Misconfiguration

CVSS 3.1 Base Score: 5.9 (Medium)

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

Reference: Section 3.6.2

4.4.4.1 Description

The "SecretCookie" cookie which is used to authenticate the user throughout the website was found to be extremely insecure as the only obfuscation in place was to encode the cookie into hex and then into base64. Once decoded it was found to contain the username, password and unix timestamp of when the cookie was created. This means that a stolen cookie could be decoded to disclose user credentials and data.

The cookie attributes are as shown below:

- HttpOnly was set to false, leaving user cookies unprotected in case of cookie theft.
- The Secure flag was set to false. This is because the website does not use HTTPS and means cookies sent over HTTP are susceptible to a MitM attack.
- The Max-Age flag is set to “session” which is when the user closes the browser. This setting is secure and does not require remediation.

4.4.4.2 Impact

This means that stolen cookies reveal the password which could be reused across other websites and compromises user security out with the website.

The lack of a HTTPOnly flag means that user cookies are compromised in case of a stored XSS attack which exists on the website.

4.4.4.3 Remediation

The HttpOnly and Secure Flags should be set to true.

4.4.5 Medium - Account Registration DoS Vulnerability

Vulnerability Class: Insufficient Anti-Automation

CVSS 3.1 Base Score: 5.3 (Medium)

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

Reference: Section 3.4.2

4.4.5.1 Description

Testing with burp-suite showed that the form does not implement any sort of rate-limiting.

4.4.5.2 Impact

This allows for a denial-of-service attack by repeatedly sending requests and overburdening the web server.

4.4.5.3 Remediation

The website must implement a form of rate-limiting to address the denial-of-service attack possibility. This would make it more difficult to flood the server with requests from one IP and protect data availability.

4.4.6 Medium - Weak Email Verification Vulnerability

Vulnerability Class: Improper Input Validation

CVSS 3.1 Base Score: 4.3 (Medium)

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

Reference: Section 3.4.2

4.4.6.1 Description

The form does not properly check email parameter inputs. This allows for multiple users under the same email address or no email address at all.

4.4.6.2 Impact

This breaks the password reset functionality which relies on each user having a unique email address and can compromise personal user data provided during registration. This includes phone numbers and addresses.

4.4.6.3 Remediation

The registration form requires stronger input filtering to prevent the registration of multiple accounts under the same email address. This could be done by ensuring that an email address does not exist in the database during the registration process. Furthermore, by imposing stronger filtering, only real emails can be used for an account.

4.4.7 Medium - Weak Password Policy and Credentials

Vulnerability Class: Broken Authentication

CVSS 3.1 Base Score: 5.3 (Medium)

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

Reference: Section 3.5.5

4.4.7.1 Description

The account system suffers from a weak password policy. The current password policy allows for guessable passwords which can be found in the rockyou wordlist.

4.4.7.2 Impact

The password found for the main admin account is present on the rockyou.txt wordlist and paired with the lack of rate-limiting, which means that an attacker could gain access to the admin panel with a dictionary brute-force attack. This is further exacerbated by the lack of a password policy and lockout mechanism which makes user and administrator data much less secure.

4.4.7.3 Remediation

A lockout mechanism for logging in would mitigate attempts for brute-forcing the login panel by stopping requests from an IP address after several incorrect login attempts. This would also help in addressing the lack of rate-limiting on the website, which is expanded upon in section 3.4.2.

NIST recommends a password policy with a minimum of 8 characters with no complexity requirements, however 15 characters is even better (National Institute of Standards and Technology, 2025). Implementation of multi-factor authentication is also recommended if possible.

Finally, user login functionality should be suspended after too many login attempts. This would work as insurance in case of a brute force attack attempt.

4.5 LOW SEVERITY VULNERABILITIES

4.5.1 Low - Exposed Directory Listings

Vulnerability Class: Security Misconfiguration

CVSS 3.1 Base Score: 3.7 (Low)

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

Reference: Section 3.3.1

4.5.1.1 Description

Gobuster revealed that sensitive web directories are publicly available including the phpMyAdmin page. The scan also found content directories were available which may contain sensitive information such as database backup files.

4.5.1.2 Impact

This can allow an attacker to attempt a brute-forcing attack to gain access to the database.

4.5.1.3 Remediation

The listed directories and phpMyAdmin page should be made available to localhost only. This would prevent users from having access to these sensitive folders.

4.5.2 Low - Information Disclosure Through robots.txt and phpinfo.php

Vulnerability Class: Information Disclosure

CVSS 3.1 Base Score: 2.7 (Low)

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

Reference: Section 3.3.1

4.5.2.1 Description

The robots.txt file reveals the existence of the phpinfo.php page which contains sensitive website configuration information.

4.5.2.2 Impact

The disclosure of details such as:

- MySQL version and configuration
- File system paths and directory structures
- The disabled_functions parameter being unset
- Server operating system

Make information gathering for an adversary significantly easier, and aids in creating crafted payloads if other vulnerabilities are found.

4.5.2.3 Remediation

phpinfo.php should be removed from the robots.txt page and should be either removed or made accessible exclusively locally. This would make the configuration details of the website more secure.

4.5.3 Low - Outdated PHP Version

Vulnerability Class: Vulnerable and Outdated Components

CVSS 3.1 Base Score: 3.1 (Low)

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

Reference: Section 3.2.1

4.5.3.1 Description

Wappalyzer and the existing robots.txt information disclosure vulnerability reveal that the website is running an outdated version of PHP which has been deprecated since 2015.

4.5.3.2 Impact

The use of PHP 5.4.7 creates a serious threat as this version, released in 2012, has been deprecated since 2015. PHP 5.4 contains 9 confirmed vulnerabilities (ExploitDB, 2025) which may be applicable here. Some of these vulnerabilities may contain proof of concepts which could be used to target the website and make it easy to do so.

4.5.3.3 Remediation

This could be addressed by updating PHP to the most recent version and continuing to receive the most up to date security patches.

5 DISCUSSION

5.1 GENERAL DISCUSSION

5.1.1 Methodology Discussion

Initially a home-made methodology geared towards Capture the Flag machines was considered. This methodology is closer to the Penetration Testing Execution Standard which follows 6 main phases (Pre-engagement is not relevant in this case). This CTF methodology is based on the mantra "test what you see" with remote code execution as the final aim, however this approach is not methodical enough for a penetration test, where the end goal is assurance through a comprehensive test.

The use of OWASP web security testing guide as a base was quite effective as it is extremely comprehensive, however much of the original methodology does not apply here. For example, sections such as subdomain reconnaissance are not relevant as it is known that the website does not utilize subdomains. Similarly, sections on certain technologies such as testing JWT (Json Web Tokens) were excluded as the website cannot be tested for services it does not use. The sections tested used recommended tools from OWASP in addition to popular tools used by other penetration testers in the industry to follow up on potential vulnerabilities.

Overall, the report was successful in using an appropriate methodology with proper tools. Throughout proof of concepts are provided in the form of screenshots and code snippets to demonstrate evidence and prove the results are repeatable.

5.1.2 Vulnerability Assessment Discussion

A possible improvement could be to use CVSS V4.0, which is the most up to date version, however, this would introduce additional complexity which could affect the readability of the report and is not necessary for a penetration test of this scale. Furthermore, CVSS V3.1 is still used throughout the industry, for example, NIST still shows CVSS scores using V3.1 by default for vulnerabilities. For all these reasons, V3.1 was deemed acceptable for the report.

The use of CVSS V3.1 made sorting the vulnerabilities found easy and readable, especially for less technical readers. It was also possible to make an objective observation on which issues hold priority over others by using the scoring system.

The vulnerabilities generally stemmed from similar root causes. Namely poor user input filtering and configuration failures:

Poor user input filtering allowed for the most destructive vulnerabilities, namely remote code execution (Section 3.2.1), SQL injection across 4 input forms (Section 3.2.3) and XSS in five locations (Section 3.3.2). Collectively, this root cause makes up 11 out of 16 (68%) total vulnerabilities and 2 out of 3 critical vulnerabilities which compromise user passwords, emails and personal details alongside business data. The same apostrophe payload works across customer.php, process.php, contact.php and changepassword.php pages. This suggests that vulnerable code is reused throughout the website, without consideration for prepared statements or the use of htmlspecialcharacters() to tackle the XSS and SQL

injection problems. With proper input validation as detailed in sections 3.2.1, 3.2.3 and 3.3.2, 11 vulnerabilities could be mitigated which makes this the most important root cause to address.

The website configuration decisions throughout the deployment of the website created 7 medium severity vulnerabilities which represent 43% of total findings. These include weak session management (section 3.4.3), insecure cookie attributes with HttpOnly and Secure flags set to false along with the absence of HTTPs which allows for MitM attacks. Most critically, the admin panel performs authentication checks after loading the page (Section 3.2.2) which allows for viewing and executing administrator functions with the use of a web proxy. This pattern shows that default configurations were accepted without security consideration. For example, HTTPs implementation is free with certificates available from services such as "Let's Encrypt" and cookie attributes remain at insecure defaults. Implementing server-side authentication, HTTPs, more secure cookie attributes alongside rate-limiting would mitigate 7 existing vulnerabilities and prevent denial-of-service and session hijacking attacks.

5.1.3 Remediation Discussion

The remediation offered in section 3 offers practical and implementable solutions for all the 16 identified vulnerabilities. Each finding contains specific guidance on how to address the problem, such as section 3.2.3.3 which specifically suggests the use of prepared statements. Similarly, section 3.2.2.3 proposes the use of the htmlspecialcharacters function to secure user input. This shows that remediation provided is specific, furthermore, references to documentation are provided where necessary, which was the aim of the report.

Additionally, section 4.1.2 goes into more detail about the root causes of the vulnerabilities so that issues can be addressed on a high level. Overall, the report is successful in providing guidance on how to resolve security issues and follow better security practices going forward.

5.2 FUTURE WORK

This penetration test shows the existing, found security vulnerabilities with mitigation instructions on how to rectify this. The results show that since remote code execution is possible, the business network may be compromised and should also be tested. This will ensure that a potential adversary has not left a back door which could be persistent even after mitigations have been made.

Mitigation should firstly focus on the various user input filtering vulnerabilities which are most critical. Following this, the configuration and deployment management issues should be addressed next as these can also be severe. Only then should the low priority vulnerabilities be addressed.

To conclude, Rickstore should follow the following steps, which are listed in order of importance:

- Network level penetration test with a complete topological map and investigation to determine if a network level attack has occurred
- Addressing user input filtering vulnerabilities
- Addressing configuration and deployment management vulnerabilities
- Addressing low priority vulnerabilities as listed in results
- Employee cyber awareness training to mitigate threat of human engineering
- Additional website penetration test to ensure changes were made properly

6 REFERENCES

Damele, B. and Stampar, M. (2025) sqlmap [Computer program] Automatic SQL injection and database takeover tool'. Available at: https://github.com/sqlmapproject/sqlmap (Accessed: 4 December 2025).

Department for Science, Innovation & Technology (2025) *Cyber security breaches survey 2025*, *GOV.UK*. Available at: https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2025 (Accessed: 4 December 2025).

ExploitDB (2025) *OffSec's Exploit Database Archive*. Available at: https://www.exploit-db.com/search?q=PHP+5.4 (Accessed: 4 December 2025).

FIRST (2019) *Common Vulnerability Scoring System v3.1: Specification Document. Revision 1. Forum of Incident Response and Security Teams. Available at: https://www.first.org/cvss/v3-1/cvss-v31-specification_r1.pdf*

Lyon, G. (2023) Nmap: The Network Mapper. Version 7.49 [Computer program]. Available at: https://nmap.org/ (Accessed: 4 December 2025).

National Institute of Standards and Technology (2025) *NIST Special Publication 800-63B*. Available at: https://pages.nist.gov/800-63-4/sp800-63b.html (Accessed: 4 December 2025).

Office for National Statistics (2025) *Retail sales, Great Britain - Office for National Statistics*, *Retail sales, Great Britain: September 2025*. Available at: https://www.ons.gov.uk/businessindustryandtrade/retailindustry/bulletins/retailsales/september2025 (Accessed: 4 December 2025).

OWASP Foundation (2021) *Web Security Testing Guide. Version 4.2*. Available at: https://owasp.org/www-project-web-security-testing-guide/stable/ (Accessed: 4 December 2025).

OWASP Foundation (2025) OWASP Zed Attack Proxy (ZAP). Version 2.15.1, *ZAP*. [Computer program] Available at: https://www.zaproxy.org/ (Accessed: 4 December 2025).

PortSwigger Ltd (2025) Burp Suite [Computer program]. Available at: https://portswigger.net/burp (Accessed: 4 December 2025).

Reeves, O.J. (2025) Gobuster [Computer program]. Available at: https://github.com/OJ/gobuster (Accessed: 4 December 2025).

Stenberg, D. (2025) curl [Computer program] Available at: https://curl.se/ (Accessed: 4 December 2025).

The PHP Documentation Group (2025) *PHP: Prepared Statements - Manual*. Available at: https://www.php.net/manual/en/mysqli.quickstart.prepared-statements.php (Accessed: 4 December 2025).

Wappalyzer (2025) *Find out what websites are built with - Wappalyzer*. Available at: https://www.wappalyzer.com/ (Accessed: 4 December 2025).

Wireshark Foundation (2025) Wireshark [Computer program] Available at: https://www.wireshark.org/ (Accessed: 4 December 2025).

APPENDIX

APPENDIX A – ZAP SCAN RESULT

6.1 SUMMARY OF ALERTS

Risk Level	Number of Alerts
High	5
Medium	8
Low	9
Informational	0

6.2 ALERTS

Name	Risk Level	Number of Instances
Cross Site Scripting (Reflected)	High	12
Path Traversal	High	1
SQL Injection	High	1
SQL Injection - MySQL	High	39
Vulnerable JS Library	High	1
Absence of Anti-CSRF Tokens	Medium	1
Application Error Disclosure	Medium	45
Content Security Policy (CSP) Header Not Set	Medium	105

Directory Browsing	Medium	49
Hidden File Found	Medium	1
Missing Anti-clickjacking Header	Medium	77
Vulnerable JS Library	Medium	5
XSLT Injection	Medium	4
Cookie No HttpOnly Flag	Low	7
Cookie without SameSite Attribute	Low	7
Cross-Domain JavaScript Source File Inclusion	Low	3
Private IP Disclosure	Low	2
Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low	107
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	351
Timestamp Disclosure - Unix	Low	3
X-Content-Type-Options Header Missing	Low	261
ZAP is Out of Date	Low	2