Web Application Penetration Test

Stephen Broadbridge

CMP319: Web Application Penetration Testing

BSc Ethical Hacking Year 3

2023/24

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

Abstract

This report presents a thorough review of the ‘Hacklab Pizza’ web application. The focus of the penetration test was on the website itself and its connected databases, specifically those which interact with the functionality of the web application itself. The methodology used was based on the OWASP web application penetration testing methodology to ensure all fundamental components of the web application were tested.

The tools employed for this analysis were chosen because of the valuable insights they could grant without risking the integrity of the web application.

The findings of this report highlight significant security concerns within the web application, with vulnerabilities identified across various critical functions. The security measures in place appeared to be rudimentary and lacking in areas that are considered common knowledge. While specific recommendations have been made to address these vulnerabilities, the extent of the issues suggests an overhaul of the web application may be required.

Contents

1 Introduction 5

1.1 Background 5

1.2 Aims 5

2 Procedure and Results 6

2.1 Overview of Procedure 6

2.2 Information Gathering 7

2.2.1 Fingerprinting Web Server 7

2.2.3 Identifying Entry Points 8

2.2.4 Map Execution Paths Through Application 9

2.2.5 Reviewing Webserver Metafiles 10

2.2.6 Fingerprint Web Application 10

2.3 Configuration and Deployment Management Testing 11

2.3.1 Test Application Platform Configuration 11

2.3.2 Test File Extensions Handling for Sensitive Information 11

2.3.3 Test HTTP Strict Transport Security 12

2.4 Identity Management Testing 13

2.4.1 Test User Registration Process 13

2.5 Authentication Testing 14

2.5.1 Testing for Credentials Transported over an Encrypted Channel 14

2.5.2 Testing for Weak Lockout Mechanism 14

2.5.3 Testing for Default Credentials 15

2.5.4 Testing for Bypassing Authentication Schema 16

2.5.5 Testing for Weak Password Policy 17

2.6 Authorisation Testing 18

2.6.1 Testing Directory Traversal 18

2.6.2 Testing for Insecure Direct Object References 18

2.7 Session Management Testing 19

2.7.1 Testing for Session Management Schema 19

2.7.2 Testing Cookie Attributes 20

2.7.3 Testing Session Timeout 20

2.7.4 Testing for Session Hijacking 21

2.8 Input Validation Testing 22

2.8.1 Testing for Reflected Cross-Site Scripting 22
2.8.2 Testing for Stored Cross-Site Scripting 23
2.8.3 Testing for SQL Injection 24
2.9 Error Handling 25
2.9.1 Analysis of Error Codes 25
2.10 Cryptography 26
2.10.1 Testing for Weak Transport Layer Security 26
2.11 Business Logic Testing 27
2.11.1 Test Business Logic Data Validation 27
2.11.2 Test Upload of Unexpected File Types 28
2.12 Client-Side Testing 29
2.12.1 Testing for HTML Injection 29
2.13 Results 30
2.13.1 Information Gathering 30
2.13.2 Vulnerability Identification 30
2.13.3 Configuration and Deployment 31
2.13.4 Identity and Access Management 31
2.13.5 Authentication Testing 31
2.13.6 Authorisation Testing 32
2.13.7 Session Management 32
2.13.8 Input Validation Testing 32
3 Discussion 33
3.1 Overall Discussion 33
3.2 Future Work 34
3.3 References 35
4 Appendices 36
Appendix A - OWASP Zap – Mapping URL’s List 36
Appendix B - Dirb – Brute Force Attack Results 37
Appendix C – Schema.sql 38
Appendix D – Nikto Configuration Scan Results 45
Appendix E – Captured Information for Session Hijacking 48
Appendix D – Figures 48

1 Introduction

1.1 Background

In today's increasingly technical society, many transactions, communications, and businesses are often conducted online but, “with any new class of technology, web applications have brought with them a new range of security vulnerabilities” (Stuttard et al., 2011). It is within this context that ensuring the security of a web application is a critical component in the overall management of a website.

Alarmingly, “18% of websites are found to contain critical severity threats and 4.1 million websites contain malware at any time” (Palatty, 2023). These statistics highlight an important problem that businesses must address, which are the risks they face in the online domain. The repercussions of security breaches, exploits redirecting users to compromised websites or any other type of cyber attack could all permanently tarnish an organisation's reputation.

It is currently estimated that there are approximately 2200 cyber attacks every day meaning a cyber attack occurs every 39 seconds on average. “In the US, a data breach costs an average of $9.44 million and cybercrime is predicted to cost $8 trillion by 2023” (Palatty, 2023).

Given these security trends, the importance of security testing is paramount to an organisation staying safe and protected in the online domain. Penetration testing can help in this situation. A penetration test will be conducted on http://192.168.1.10/. This report aims to evaluate the security posture of the web application while demonstrating the feasibility of known and found exploits identified through testing. The scope of this penetration test is limited to the web application and server the web application is being hosted on.

The application being tested is “Hacklab Pizza”, which is an online pizza restaurant. Prior to the testing the following credentials were provided:

Username: hacklab@hacklab.com

Password: hacklab

1.2 Aims

The aims of this project are as follows:

- **Conduct a thorough penetration test:** Execute a penetration test aimed at evaluating the security mechanisms of “Hacklab Pizza’s” web application.
- **Uncover and exploit vulnerabilities:** Identify vulnerabilities within the web application and validate them through proof of concept exploits to establish their practical impact.
- **Report and detail methodology and findings:** Generate a report detailing the methodology employed throughout the testing process and the subsequent findings.

2 PROCEDURE AND RESULTS

2.1 OVERVIEW OF PROCEDURE

The OWASP methodology, which is listed below, is used in this study. Throughout the penetration test, each stage is divided into several sections.

1. Information Gathering
2. Configuration and Deployment Management Testing
3. Identify Management Testing
4. Authentication Testing
5. Authorisation Testing
6. Session Management Testing
7. Input Validation Testing
8. Error Handling
9. Cryptography
10. Business Logic Testing
11. Client-Side Testing.

These procedures guarantee that no testing component is skipped over and give an in-depth investigation of the testing that was completed. The following tools were employed during the penetration test: **Nmap, OWASP ZAP, Dirb, whatweb, Nikto, Curl, CyberChef, Mozilla Firefox**

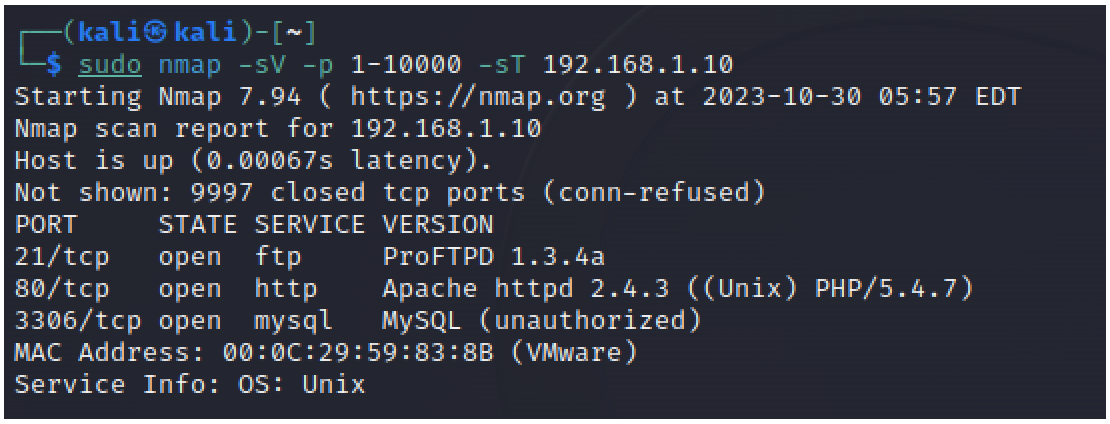
2.2 Information Gathering

The methodology begins by gathering information on the contents of the web application. This is a critical step in the process as it provides important data on the structure and components of the web application. This will streamline the process of identifying and cataloguing vulnerabilities found during the penetration test. Additionally, this information will identify which technologies exist within the web application which can save time on testing things that do not exist.

This foundational step aligns closely with the overall aims of this project, providing the necessary groundwork upon which a comprehensive penetration test can be effectively executed.

2.2.1 Fingerprinting Web Server

The OWASP methodology emphasises the importance of foot printing the webserver as the first stage of a web application penetration test. This step identifies the operating system, services and versions that are running. The tool used for this step was Nmap. A scan was performed on 192.168.1.10.



```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sV -p 1-10000 -sT 192.168.1.10
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-30 05:57 EDT
Nmap scan report for 192.168.1.10
Host is up (0.00067s latency).
Not shown: 9997 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
21/tcp   open  ftp     ProFTPD 1.3.4a
80/tcp   open  http    Apache httpd 2.4.3 ((Unix) PHP/5.4.7)
3306/tcp open  mysql   MySQL (unauthorized)
MAC Address: 00:0C:29:59:83:8B (VMware)
Service Info: OS: Unix
```

Figure 1 - Nmap Scan

This scan identified that ports 21, 80 and 3306 were open and the webserver was running on Apache/2.4.3 with a MySQL database.

2.2.3 Identifying Entry Points

To pinpoint potential weaknesses in the application a manual search of the website is undertaken to catalogue all available entry points. This will allow the discovery of areas of interest to be investigated further in the report. The entry points identified are as follows:

Account Creation and Authentication:

- Login – Email
- Login – Password
- Register – First Name
- Register – Last Name
- Register – Email
- Register – Password
- Register – Confirm Password
- Register – Security Answer

User Profile Management (Post-Authentication)

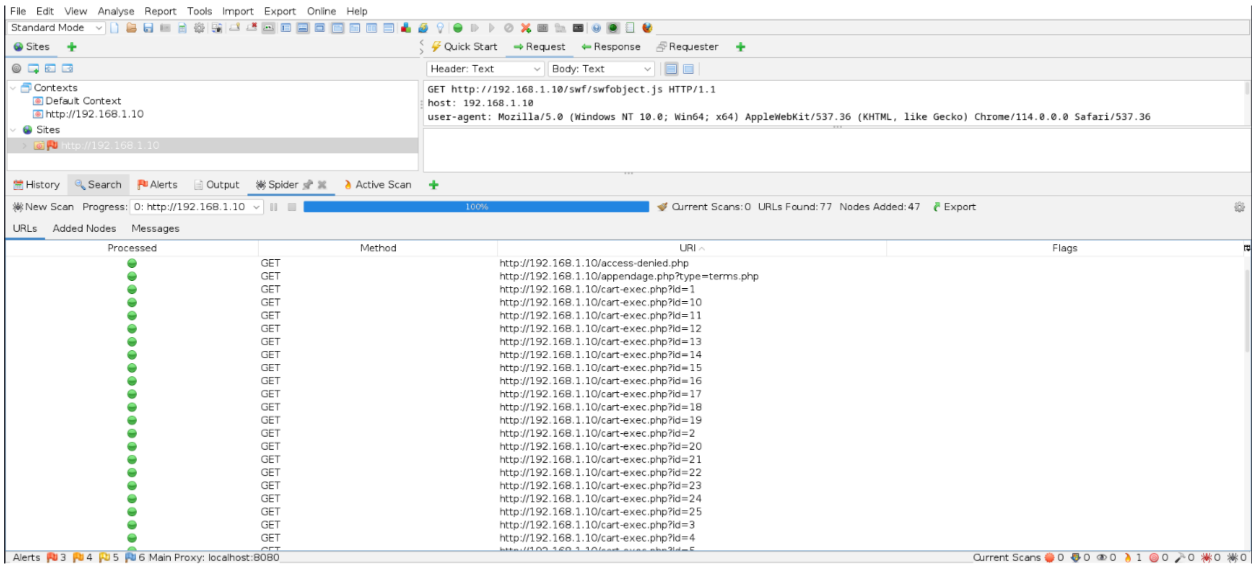
- Edit Profile – Old Password
- Edit Profile – New Password
- Edit Profile – Confirm New Password
- Edit Profile – First Name
- Edit Profile – Street Address
- Edit Profile – P.O Box No
- Edit Profile - City
- Edit Profile – Mobile No
- Edit Profile – Landline No

User Interaction:

- Rate Us – Comment
- Edit Profile – Change Password (Old Password)
- Edit Profile - Change Password (New Password)
- Edit Profile - Change Password (Confirm New Password)
- Edit Profile – Add Delivery/Billing Address (First Name)
- Edit Profile – Add Delivery/Billing Address (Last Name)
- Edit Profile – Add Delivery/Billing Address (P.O Box)
- Edit Profile – Add Delivery/Billing Address (City)
- Edit Profile – Add Delivery/Billing Address (Mobile No)
- Edit Profile – Add Delivery/Billing Address (Landline No)
- Edit Profile – Change Profile Picture

2.2.4 Map Execution Paths Through Application

To map the execution paths within the application the OWASP ZAP tool was implemented. OWASP Zap find all URLS associated within the http://192.168.1.10 domain via spidering through the application. Overall, OWASP Zap discovered 77 URL's. The complete list of URL's are in Appendix A.



```
GET http://192.168.1.10/swf/swfobject.js HTTP/1.1
host: 192.168.1.10
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.36
```

New Scan Progress: 0: http://192.168.1.10 100% Current Scans: 0 URLs Found: 77 Nodes Added: 47

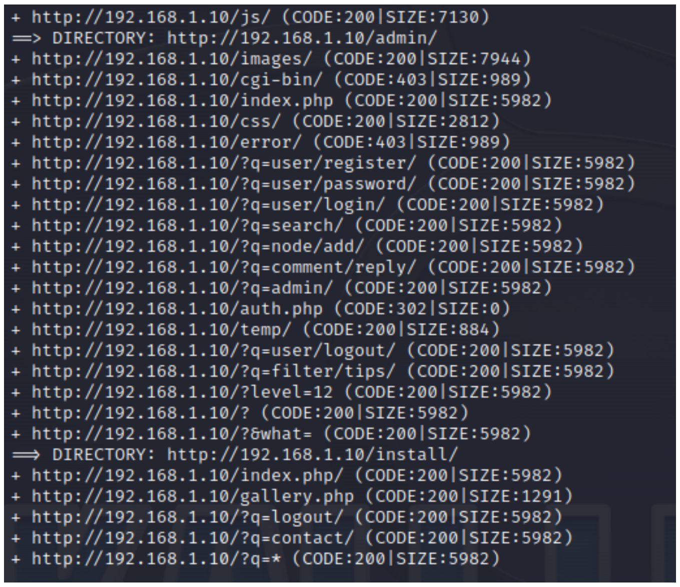
Processed	Method	URI	Flags
●	GET	http://192.168.1.10/access-denied.php	
●	GET	http://192.168.1.10/appendage.php?type=terms.php	
●	GET	http://192.168.1.10/cart-exec.php?id=1	
●	GET	http://192.168.1.10/cart-exec.php?id=10	
●	GET	http://192.168.1.10/cart-exec.php?id=11	
●	GET	http://192.168.1.10/cart-exec.php?id=12	
●	GET	http://192.168.1.10/cart-exec.php?id=13	
●	GET	http://192.168.1.10/cart-exec.php?id=14	
●	GET	http://192.168.1.10/cart-exec.php?id=15	
●	GET	http://192.168.1.10/cart-exec.php?id=16	
●	GET	http://192.168.1.10/cart-exec.php?id=17	
●	GET	http://192.168.1.10/cart-exec.php?id=18	
●	GET	http://192.168.1.10/cart-exec.php?id=19	
●	GET	http://192.168.1.10/cart-exec.php?id=2	
●	GET	http://192.168.1.10/cart-exec.php?id=20	
●	GET	http://192.168.1.10/cart-exec.php?id=21	
●	GET	http://192.168.1.10/cart-exec.php?id=22	
●	GET	http://192.168.1.10/cart-exec.php?id=23	
●	GET	http://192.168.1.10/cart-exec.php?id=24	
●	GET	http://192.168.1.10/cart-exec.php?id=25	
●	GET	http://192.168.1.10/cart-exec.php?id=3	
●	GET	http://192.168.1.10/cart-exec.php?id=4	

Alerts 3 4 5 6 Main Proxy: localhost:8080

Figure 2 - OWASP Zap Mapping

Upon completing the OWASP Zap scan, the next step involved utilising the Dirb tool to perform a brute force attack. The aim was to discover any potential hidden files, folders and URLs that may not be visible through regular browsing. This is vital for mapping out all execution paths through the web application.

The screenshot below provides a shortened view of the Dirb scan results. A full review of the information can be found in Appendix B.

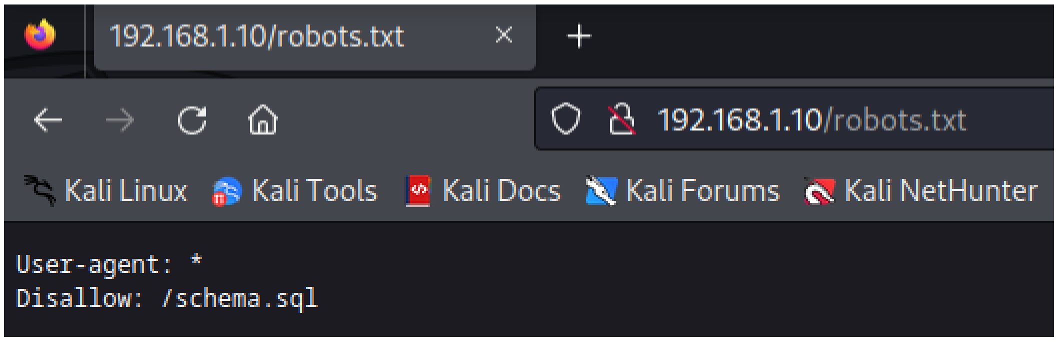


```
+ http://192.168.1.10/js/ (CODE:200|SIZE:7130)
==> DIRECTORY: http://192.168.1.10/admin/
+ http://192.168.1.10/images/ (CODE:200|SIZE:7944)
+ http://192.168.1.10/cgi-bin/ (CODE:403|SIZE:989)
+ http://192.168.1.10/index.php (CODE:200|SIZE:5982)
+ http://192.168.1.10/css/ (CODE:200|SIZE:2812)
+ http://192.168.1.10/error/ (CODE:403|SIZE:989)
+ http://192.168.1.10/?q=user/register/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=user/password/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=user/login/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=search/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=node/add/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=comment/reply/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=admin/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/auth.php (CODE:302|SIZE:0)
+ http://192.168.1.10/temp/ (CODE:200|SIZE:884)
+ http://192.168.1.10/?q=user/logout/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=filter/tips/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?level=12 (CODE:200|SIZE:5982)
+ http://192.168.1.10/? (CODE:200|SIZE:5982)
+ http://192.168.1.10/?&what= (CODE:200|SIZE:5982)
==> DIRECTORY: http://192.168.1.10/install/
+ http://192.168.1.10/index.php/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/gallery.php (CODE:200|SIZE:1291)
+ http://192.168.1.10/?q=logout/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=contact/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=* (CODE:200|SIZE:5982)
```

Figure 3 - Dirb Brute Force Attack

2.2.5 Reviewing Webserver Metafiles

After the enumeration of all the URLs within 192.168.1.10 web application the focus was shifted to examining the servers metafiles. This phase was started by navigating to the URL 192.168.1.10/robots.txt to scrutinise and unintentional data exposure. The robots.txt file indicated an attempt to hide the schema.sql file from webcrawlers.



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

Figure 4 - 192.168.1.10/robots.txt

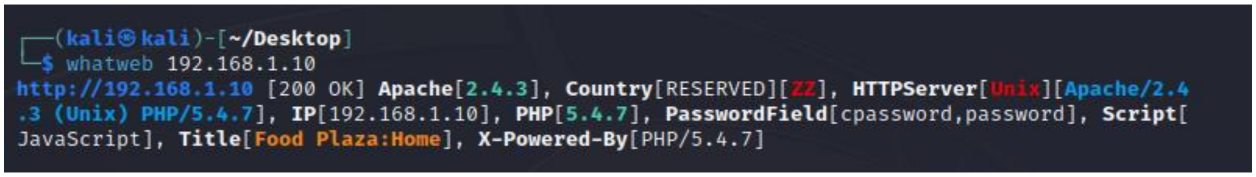
Exploring this discovery further, navigating to 192.168.1.10/schema.sql resulted in the download of the schema.sql file. This document servers as the blueprint for the SQL database, outlining its structure and relationships. The fact that this file is accessible publicly despite its intent to conceal it presents a significant security risk. Having access the database schema could enable an attacker to conduct more targeted SQL injection attacks. This discovery is categories as a significant security misconfiguration and should be rectified immediately by restricting public access to the schema.sql file.

For a full comprehensive review, the schema.sql file contents can be found in Appendix C.

2.2.6 Fingerprint Web Application

To further understand the underlying technologies of the web application, fingerprinting was conducted using the whatweb tool. The results from this scan shown below display critical details about the web application architecture. Notably, the application runs on Apache 2.4.3 and PHP 5.4.7, both of which are out of date and contain numerous security vulnerabilities.

Additionally, the scan results reveal the use of JavaScript. This opens the possibilities for additional testing focused on client-side Cross-Site Scripting (XSS).



```
┌──(kali㉿kali)-[~/Desktop]
└─$ whatweb 192.168.1.10
http://192.168.1.10 [200 OK] Apache[2.4.3], Country[RESERVED][ZZ], HTTPServer[Unix][Apache/2.4
.3 (Unix) PHP/5.4.7], IP[192.168.1.10], PHP[5.4.7], PasswordField[cpassword,password], Script[
JavaScript], Title[Food Plaza:Home], X-Powered-By[PHP/5.4.7]
```

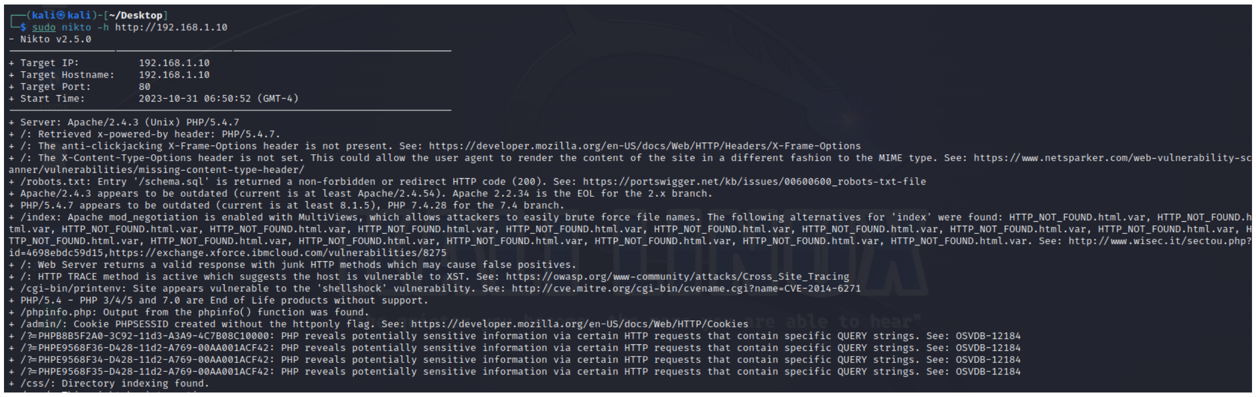
Figure 5 - whatweb results

2.3 Configuration and Deployment Management Testing

In the configuration and deployment management testing phase the focus is shifted into examining the web applications architectural elements and how they are configured. This is an important step and aims to build on the information gathering stage. The aim is to identify any critical misconfigurations that could potentially compromise the web applications security.

2.3.1 Test Application Platform Configuration

To understand if the web application had any known vulnerabilities, the Nikto tool was utilised. The Nikto scan revealed several critical security concerns, some of which are displayed in the screenshot below.



```
┌──(kali㉿kali)-[~/Desktop]
└─$ sudo nikto -h http://192.168.1.10
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          192.168.1.10
+ Target Hostname:    192.168.1.10
+ Target Port:        80
+ Start Time:         2023-10-31 06:50:52 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.3 (Unix) PHP/5.4.7
+ /: Retrieved x-powered-by header: PHP/5.4.7.
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-sc
anner/vulnerabilities/missing-content-type-header/
+ /robots.txt: Entry '/schema.sql' is returned a non-forbidden or redirect HTTP code (200). See: https://portswigger.net/kb/issues/00600600_robots-txt-file
+ Apache/2.4.3 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ PHP/5.4.7 appears to be outdated (current is at least 8.1.5), PHP 7.4.28 for the 7.4 branch.
+ /index: Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. The following alternatives for 'index' were found: HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.h
tml.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, H
TTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var. See: http://www.wisec.it/sectou.php?
id=4698ebdc59d15,https://exchange.xforce.ibmcloud.com/vulnerabilities/8275
+ /: Web Server returns a valid response with junk HTTP methods which may cause false positives.
+ /: HTTP TRACE method is active which suggests the host is vulnerable to XST. See: https://owasp.org/www-community/attacks/Cross_Site_Tracing
+ /cgi-bin/printenv: Site appears vulnerable to the 'shellshock' vulnerability. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271
+ PHP/5.4 - PHP 3/4/5 and 7.0 are End of Life products without support.
+ /phpinfo.php: Output from the phpinfo() function was found.
+ /admin/: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /css/: Directory indexing found.
```

Figure 6 - Nikto Scan

As previously discovered the server is running outdated versions of Apache (2.4.3) and PHP (5.4.7) and additionally shows that there is an absence of security headers which is a concern as it weakens the web applications defences. A full review of the results can be found in Appendix D.

2.3.2 Test File Extensions Handling for Sensitive Information

Throughout the penetration test so far, multiple tools have been utilised to scan and enumerate the web applications file structure. Nikto, OWASP Zap and Dirb were used to identify publicly accessible files, directories, and URL's. These can be seen in Appendix A, B, D.

Noteworthy findings include:

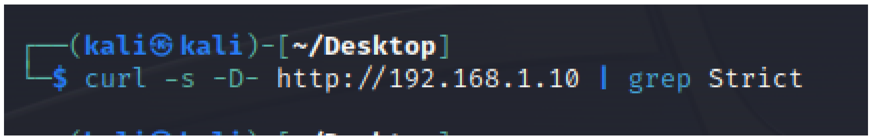
- **Downloadable schema.sql file:** The '/robots.txt' file contains an entry for a '/schema.sql' which is accessible and downloadable.
- **Directory Indexing:** Directories such as '/css/', '/install', '/stylesheets ' and '/temp' have directory indexing enabled, exposing potentially sensitive files.
- **Phpinfo() Output:** The '/phpinfo.php' endpoint runs the 'phpinfo()' function revealing sensitive information.

2.3.3 Test HTTP Strict Transport Security

The web application contains a HTTP Strict Transport Security (HSTS) header. This is a security mechanism that forces webservers to communicate with via secure HTTPS connections only. To test if a HSTS header is present the curl tool was utilised along with grep using the following command:

curl -s -D- http://192.168.1.10 | grep Strict

The command returned no result, highlighting the absence of the HSTS header on 192.168.1.10 as shown below.



```
┌──(kali㉿kali)-[~/Desktop]
└─$ curl -s -D- http://192.168.1.10 | grep Strict
```

Figure 7 - HSTS testing

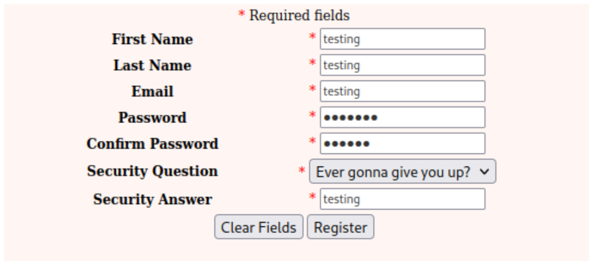
This absence is considered a security as it leaves the web application vulnerable to protocol downgrade attacks where an attacker could intercept and alter the communication between the user’s browser and the webserver.

2.4 IDENTITY MANAGEMENT TESTING

2.4.1 Test User Registration Process

The registration process for the web application contains no checks to see if a created account is legitimate. The email entry does not require any validation such as an "@" symbol to signify a legitimate email or require any password requirements. In addition, the password and confirm password field do not require the same password as shown below.

In this phase of the penetration test an evaluation was conducted to assess the web applications user registration process. The test was undertaken by completing the web applications user registration form. It was observed that the registration form lacks adequate validation checks for email addresses. There is no requirement for the '@' symbol, a fundamental component of any email address. A screenshot displaying this discrepancy is shown below.



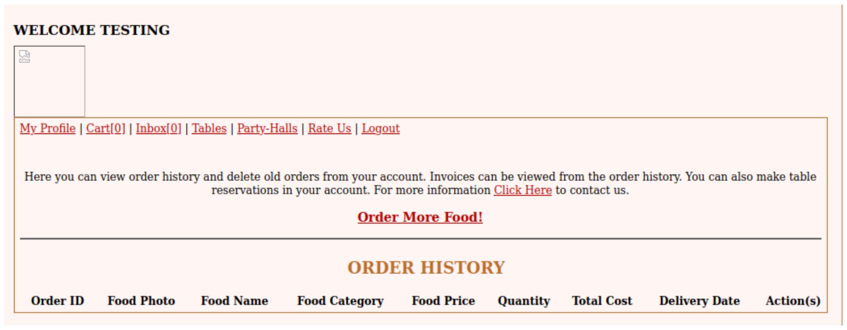
* Required fields

First Name	* testing
Last Name	* testing
Email	* testing
Password	* •••••••
Confirm Password	* ••••••
Security Question	* Ever gonna give you up?
Security Answer	* testing

Figure 8 - User Registration Process Test

Secondly, the form fails to impose any password complexity rules, making it vulnerable to brute-force and dictionary attacks.

Lastly, the form also has no correlation checks between 'Password' and 'Confirm Password' fields during the registration process. To demonstrate this, test entries "testing" for 'Password' and "testin" for 'Confirm Password' were used. Despite the discrepancy between the two entries the account creation was successful. A screenshot of the successful account creation is shown below.



WELCOME TESTING

My Profile | Cart[0] | Inbox[0] | Tables | Party-Halls | Rate Us | Logout

Here you can view order history and delete old orders from your account. Invoices can be viewed from the order history. You can also make table reservations in your account. For more information Click Here to contact us.

ORDER HISTORY

Order ID	Food Photo	Food Name	Food Category	Food Price	Quantity	Total Cost	Delivery Date	Action(s)

Figure 9 - Created User Login

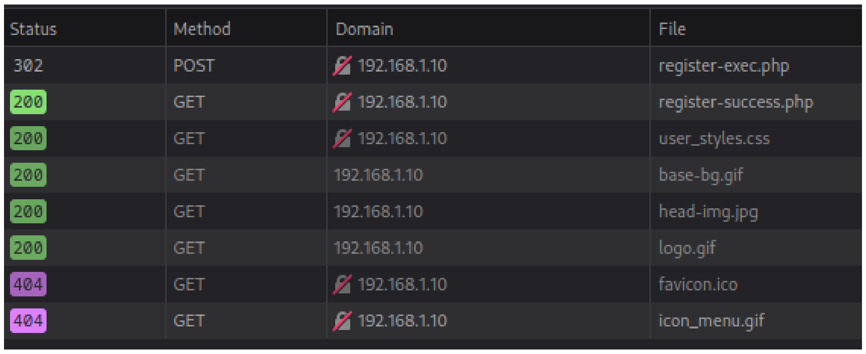
After successfully registering, the user will then have complete access to the user section of the web application. In addition, the same email cannot be used to make multiple accounts. Below details a screenshot of the successfully created account logged in.

2.5 AUTHENTICATION TESTING

This phase of the penetration test aims to scrutinise the authentication mechanisms used by the web application to authenticate its users. This phase aims to identify vulnerabilities that could allow unauthorised access or compromise user accounts.

2.5.1 Testing for Credentials Transported over an Encrypted Channel

The objective of this test is to ensure that the web application handles user credentials in a secure manner, specifically during transmission. Using the developer tools within the Firefox web browser an observation of network requests during the user registration process was conducted. The screenshot below displays the network activity when registering an account.



Status	Method	Domain	File
302	POST	192.168.1.10	register-exec.php
200	GET	192.168.1.10	register-success.php
200	GET	192.168.1.10	user_styles.css
200	GET	192.168.1.10	base-bg.gif
200	GET	192.168.1.10	head-img.jpg
200	GET	192.168.1.10	logo.gif
404	GET	192.168.1.10	favicon.ico
404	GET	192.168.1.10	icon_menu.gif

Figure 10 - HTTP User Credentials

An immediate concern is the POST request which will typically carry user credentials is being sent to 'register-exec.php' using HTTP, which is an unencrypted protocol. This is a critical security risk as information such as usernames and passwords can be intercepted by attackers using Man-in-the-Middle attacks. It is recommended that the web application shifts to HTTPS to ensure the encryption of data between the user and the web application.

2.5.2 Testing for Weak Lockout Mechanism

The web application lockout mechanism was examined for the user login section. The examination revealed that there was no lockout mechanism implemented meaning the web application is vulnerable to brute force attacks.

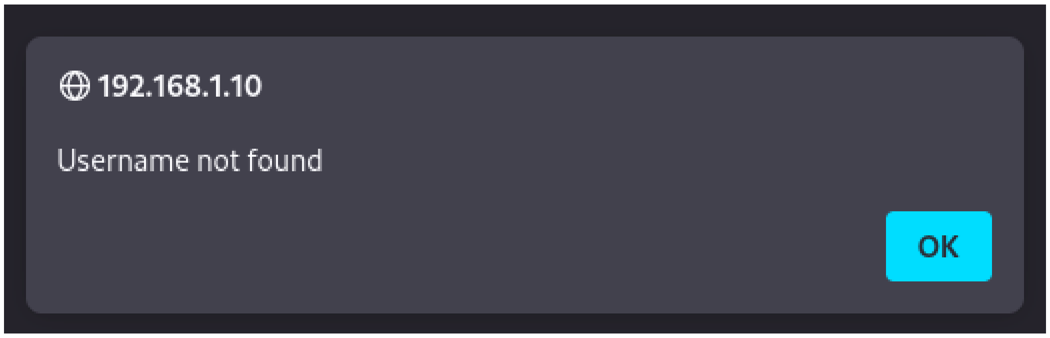
This vulnerability was identified by repeated attempts to log in with incorrect credentials without encountering any restrictions or consequences.

2.5.3 Testing for Default Credentials

The admin login page is accessible by navigating to http://192.168.1.10/admin/login-form.php. Entries were manually entered into the form. The following logins were tested:

Username	**Password**
admin	admin
admin	password
admin	12345
test	test

It should be noted that when the username of "test" was entered a pop up displaying the message "Username not found" appeared signifying that previous attempts using the username of "admin" was legitimate username. This can be seen in the figure below.



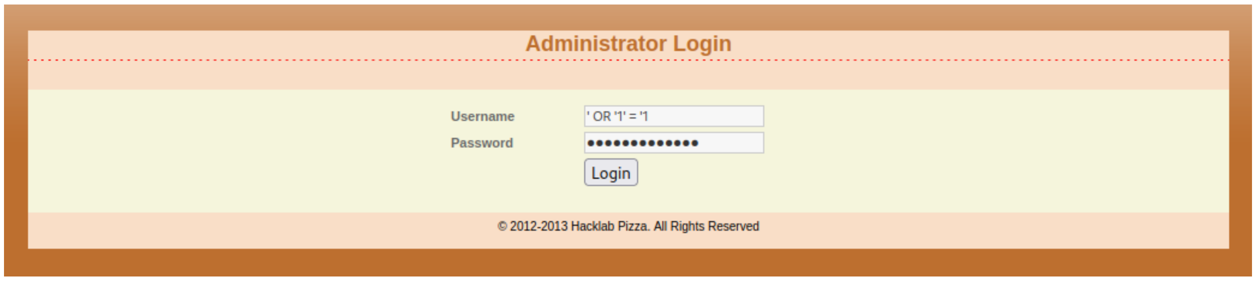
192.168.1.10

Username not found

Figure 11 - Username not found.

2.5.4 Testing for Bypassing Authentication Schema

To test if the authentication on the web application can be bypassed, an SQL injection was tested on the admin login page on http://192.168.1.10/admin/login-form.php. The query "' OR '1' = '1" was inputted in both username and password sections of the form. As seen in the screenshot below.



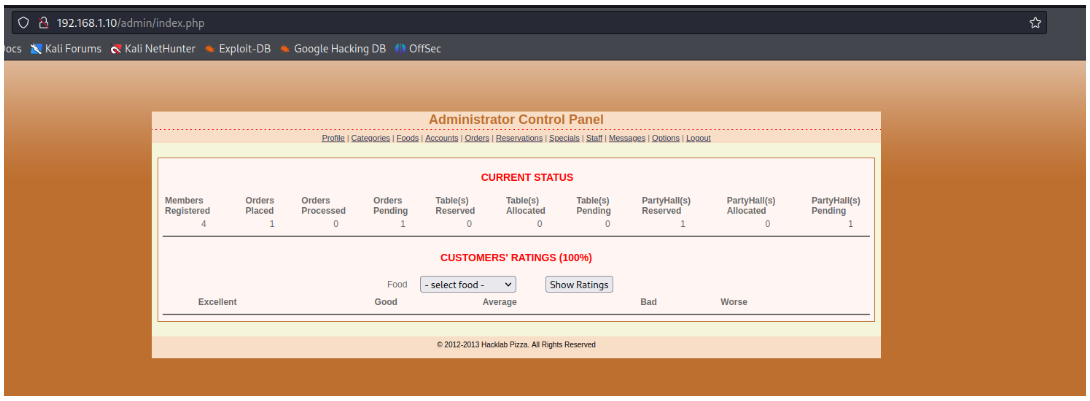
Administrator Login

Username	' OR '1' = '1
Password	●●●●●●●●●●●●●

© 2012-2013 Hacklab Pizza. All Rights Reserved

Figure 12 - Admin Login SQL Injection

This SQL injection attack was a success granting access to the Administrator Control Panel shown in the screenshot below. This discovery highlights a critical vulnerability that the web application does not validate user inputs correctly allowing the execution of SQL code. This also indicates that entries are directly passed to the SQL server without any parameterisation.



192.168.1.10/admin/index.php

ocs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec

Administrator Control Panel

Profile | Categories | Foods | Accounts | Orders | Reservations | Specials | Staff | Messages | Options | Logout

CURRENT STATUS

Members Registered	Orders Placed	Orders Processed	Orders Pending	Table(s) Reserved	Table(s) Allocated	Table(s) Pending	PartyHall(s) Reserved	PartyHall(s) Allocated	PartyHall(s) Pending
4	1	0	1	0	0	0	1	0	1

CUSTOMERS' RATINGS (100%)

Food: - select food -

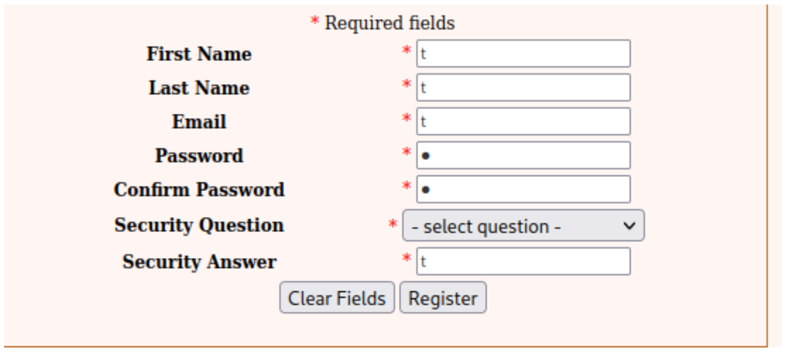
Excellent	Good	Average	Bad	Worse

© 2012-2013 Hacklab Pizza. All Rights Reserved

Figure 13 - Admin Control Panel Access

2.5.5 Testing for Weak Password Policy

As highlighted in section 2.4.1 it was discovered that the user registration form had a lack validation checks when creating an account. To validate this further another account was created using the minimalistic input of "t" in all fields, including email and password as shown below.



* Required fields

First Name	* t
Last Name	* t
Email	* t
Password	* •
Confirm Password	* •
Security Question	* - select question -
Security Answer	* t

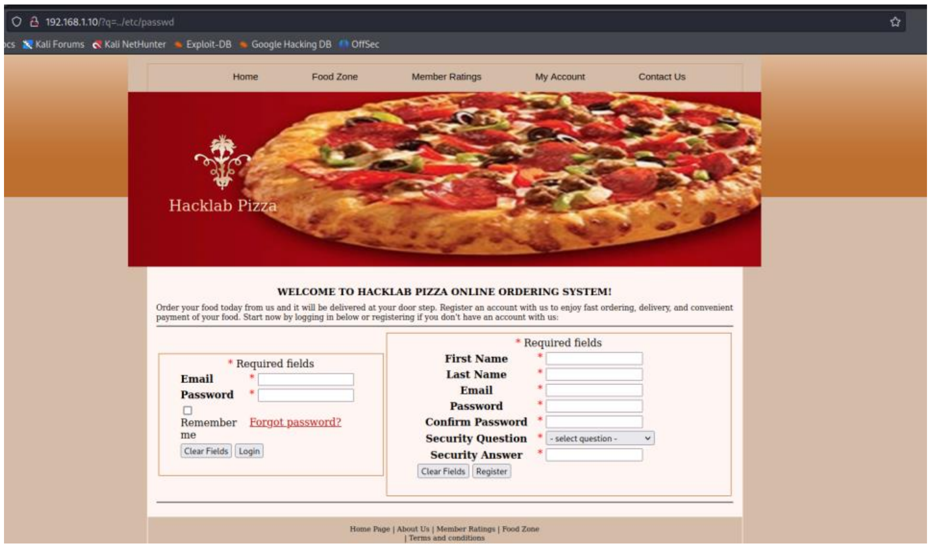
Figure 14 - Weak Password Policy

This successfully created the account and indicates that there is no password policy in place whatsoever. This oversight could allow attackers to perform brute force or dictionary attacks. It is recommended that a robust password policy is implemented.

2.6 AUTHORISATION TESTING

2.6.1 Testing Directory Traversal

During this phase, an attempt was made to exploit potential directory traversal vulnerabilities on 192.168.1.10. The test involved manipulating the query parameter in the URL by injecting '../etc/passwd' to see if it was possible to traverse directories and retrieve sensitive files from the server. The specific URL used for the test was 'http://192.168.1.10/?q=../etc/passswd'. However upon accessing this URL, the web application response remained unchanged and continued displaying the home page, as shown in the screenshot below.



WELCOME TO HACKLAB PIZZA ONLINE ORDERING SYSTEM!

Order your food today from us and it will be delivered at your door step. Register an account with us to enjoy fast ordering, delivery, and convenient payment of your food. Start now by logging in below or registering if you don't have an account with us:

* Required fields
Email *
Password *
Remember me
Forgot password?

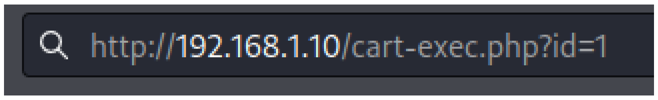
* Required fields
First Name *
Last Name *
Email *
Password *
Confirm Password *
Security Question *
Security Answer *

Figure 15 - 192.168.1.10/?q=../etc/passwd

This behaviour suggest that the web application is sanitising the input effectively to prevent Directory Traversal attacks.

2.6.2 Testing for Insecure Direct Object References

A test for Insecure Direct Object References (IDOR) was conducted on the web application, specifically targeting the shopping cart management feature accessed via '/cart-exec.php'. This test involved manipulated the 'id' parameter within the URL, which is a common vector for IDOR attacks.



http://192.168.1.10/cart-exec.php?id=1

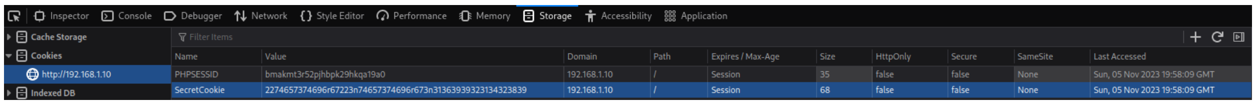
Figure 16 - IDOR Attack Vector

The test was conducted under authenticated conditions and allowed for the addition of cart items to be placed within the logged in users own cart. It was identified that there were no cross-user capabilities present within the web application.

2.7 Session Management Testing

2.7.1 Testing for Session Management Schema

The session management schema was tested to assess the security of a user's session cookies. The Mozilla Firefox web browser's developer tools were used to capture the users login credentials during the authentication process. This consisted of two separate cookies labelled as "PHPSESSID" and "Secret Cookie". These can be seen in the screenshot below.



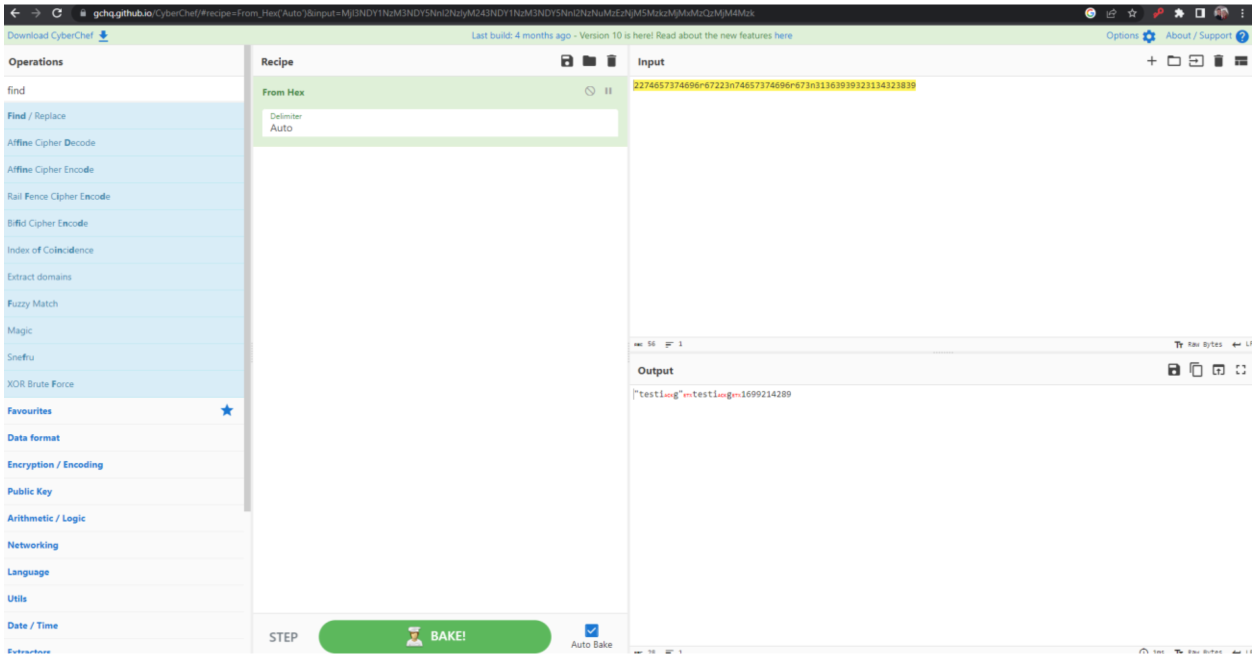
Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
PHPSESSID	bmakmt3r52pjhbpk29hkqa19a0	192.168.1.10	/	Session	35	false	false	None	Sun, 05 Nov 2023 19:58:09 GMT
SecretCookie	2274657374696r67223n74657374696r673n31363939323134323839	192.168.1.10	/	Session	68	false	false	None	Sun, 05 Nov 2023 19:58:09 GMT

Figure 17 - Login Cookies

The details for these cookies were:

- **PHPSESSID:** bmakmt3r52pjhbpk29hkqa19a0
- **SecretCookie:** 2274657374696r67223n74657374696r673n31363939323134323839

The 'PHPSESSID' cookie is a session token, while the 'SecretCookie' value is a hex-encoded string and could potentially hold login credentials of the user. Using the CyberChef tool the 'SecretCookie' revealed a plaintext string which can be seen in the screenshot below.



Recipe: From Hex — Delimiter: Auto

Input:
```
2274657374696r67223n74657374696r673n31363939323134323839
```

Output:
```
"testiACKg"ETXtestiACKgETX1699214289
```

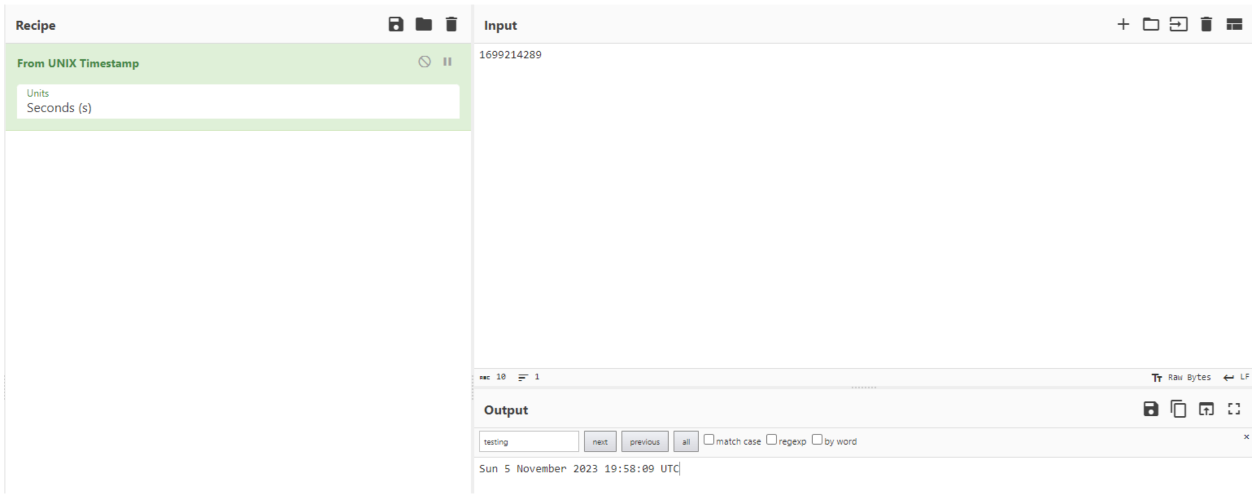
Figure 18 - CyberChef Result

The CyberChef result shows that the 'SecretCookie' value, once decoded yielded a plaintext string which closely resembled the word 'testing' followed by a numerical sequence. This near readable format of the 'SecretCookie' data raises concerns over the handling of sensitive user data.

2.7.2 Testing Cookie Attributes

Using the information gathered in section 2.6.1 about the value held within 'SecretCookie' it displayed 3 separate pieces of information.

The first value contains the email of the user, and the second value is the password. In this case 'testing' and 'testing'. The numerical sequence following this information is a UNIX timestamp. This timestamp displays the time that the cookie was created. This can be seen in the screenshot below.



```
1699214289
```

```
Sun 5 November 2023 19:58:09 UTC
```

Figure 19 - UNIX Timestamp

2.7.3 Testing Session Timeout

A test was conducted to evaluate if the website had a session timeout feature and if so to test its effectiveness. This is essential for preventing unauthorised access to inactive sessions and ensuring sensitive information is not exposed to the browser cache.

This test was performed by utilising the developer tools within Mozilla Firefox to observe the 'PHPSESSID' after a prolonged period of inactivity by the user. The session was displayed no session timeout behaviour and the session remained active even after a substantial amount of time had expired.

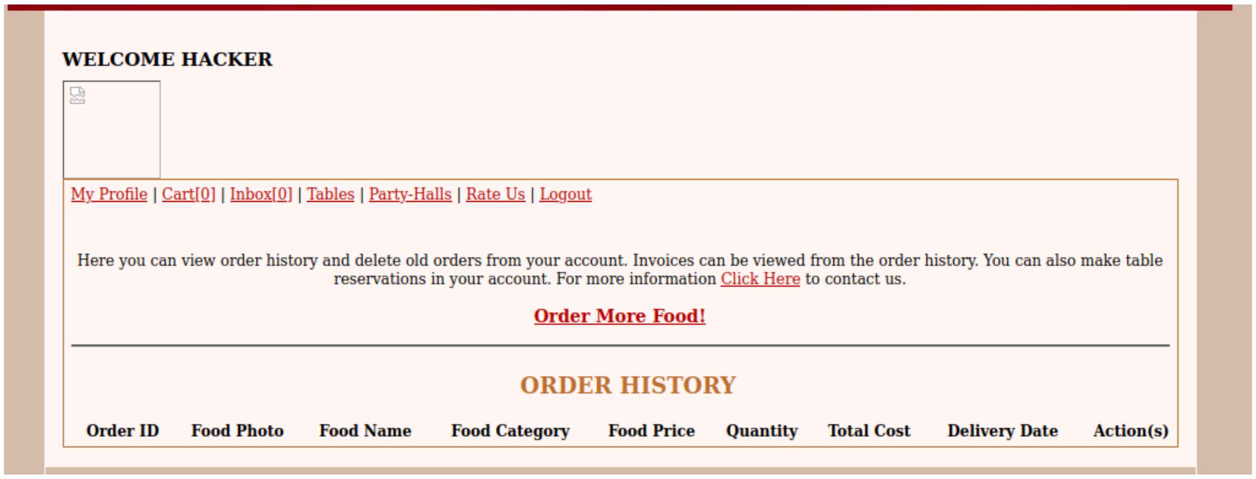
This is a security concern as attackers could exploit an idle session, potentially leading to unauthorised access to users accounts compromising sensitive information.

Ideally, sessions should be configured to a 15-30 minute inactivity period.

2.7.4 Testing for Session Hijacking

The potential for session hijacking was testing by examining the exploitation of the session management schema for http://192.168.1.10. The initial step involved using OWASP ZAP to intercept and capture the cookie values for 'PHPSESSID' and 'SecretCookie' during the login process for the user 'testing'. It should be noted that these cookies did not have the Secure attribute rendering them vulnerable for interception over non-HTTPS connections.

After this, a separate browser session was opened where another account with the alias 'hacker' was logged in. This can be seen in the screenshot below.



WELCOME HACKER

My Profile | Cart[0] | Inbox[0] | Tables | Party-Halls | Rate Us | Logout

Here you can view order history and delete old orders from your account. Invoices can be viewed from the order history. You can also make table reservations in your account. For more information Click Here to contact us.

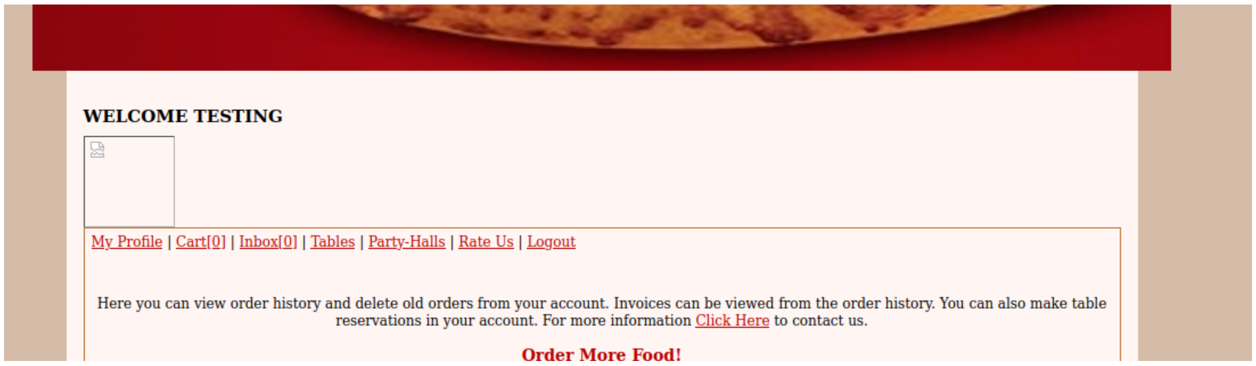
Order More Food!

ORDER HISTORY

Order ID	Food Photo	Food Name	Food Category	Food Price	Quantity	Total Cost	Delivery Date	Action(s)

Figure 20 - Hacker Account Logged In

Utilising the developer tools within Mozilla Firefox, the cookie values obtained from the testing account were manually injected into the session handling the 'hacker' account. Upon refreshing the browser, the session had transitioned to the 'testing' account confirming that it is possible to hijack sessions my obtaining the insecure cookie values. This is shown in the screenshot below.



WELCOME TESTING

My Profile | Cart[0] | Inbox[0] | Tables | Party-Halls | Rate Us | Logout

Here you can view order history and delete old orders from your account. Invoices can be viewed from the order history. You can also make table reservations in your account. For more information Click Here to contact us.

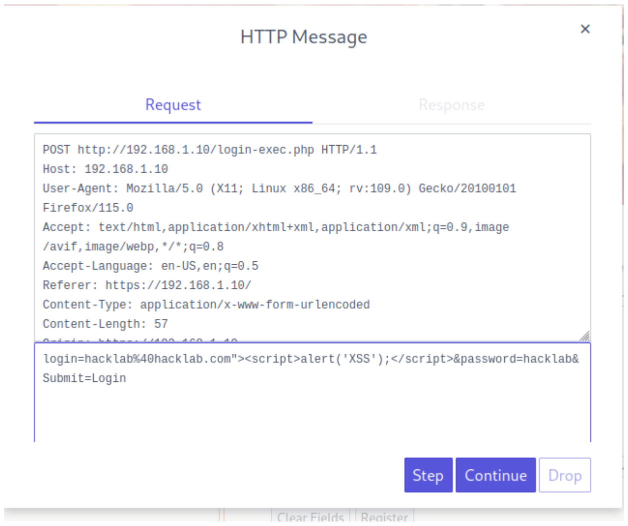
Order More Food!

Figure 21 - Session Hijack Success

2.8 Input Validation Testing

2.8.1 Testing for Reflected Cross-Site Scripting

During this phase an attempt was made to identify if the application was vulnerable to Reflected Cross-Site Scripting. This was tested using manual injection techniques. An XSS payload was crafted as '><script>alert('XSS');', which was inputted into the login forms input field, aimed at triggering a JavaScript alert upon successful reflection to the server. This attempt can be seen in the screenshot below.

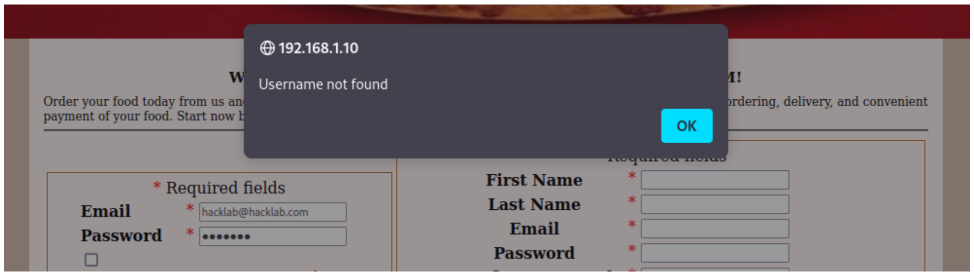


```
POST http://192.168.1.10/login-exec.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image
/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: https://192.168.1.10/
Content-Type: application/x-www-form-urlencoded
Content-Length: 57
```

```
login=hacklab%40hacklab.com"><script>alert('XSS');</script>&password=hacklab&
Submit=Login
```

Figure 22 - Reflected XSS Payload

Upon submission, the applications response did not reflect the XSS payload and instead, returned a standard error massage displaying 'Username not found'. This evidence can seen in the screenshot below.


192.168.1.10

Username not found

OK
Order your food today from us an
payment of your food. Start now b
rdering, delivery, and convenient
* Required fields
Email * hacklab@hacklab.com
Password * •••••••
First Name *
Last Name *
Email *
Password *

Figure 23 - XSS Username not found

The absence of the payload in the response and lack of any script execution indicates that this specific entry point within the web application is not vulnerable to a reflected XSS attack.

2.8.2 Testing for Stored Cross-Site Scripting

During this phase of the penetration test, stored Cross-Site Scripting was tested by interacting with the User Interaction section of the web application. Within http://192.168.1.10/ratings.php users can fill out a form which will leave a comment on http://192.168.1.10/member-ratings.php

A payload script was created as '<script>alert(document.cookie);</script>' which was inputted within the 'Rate Us' form with the aim to display the 'PHPSESSID' cookie within an alert box. This can be seen within the image below.

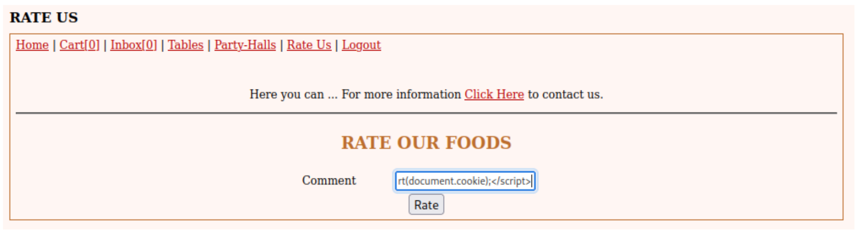
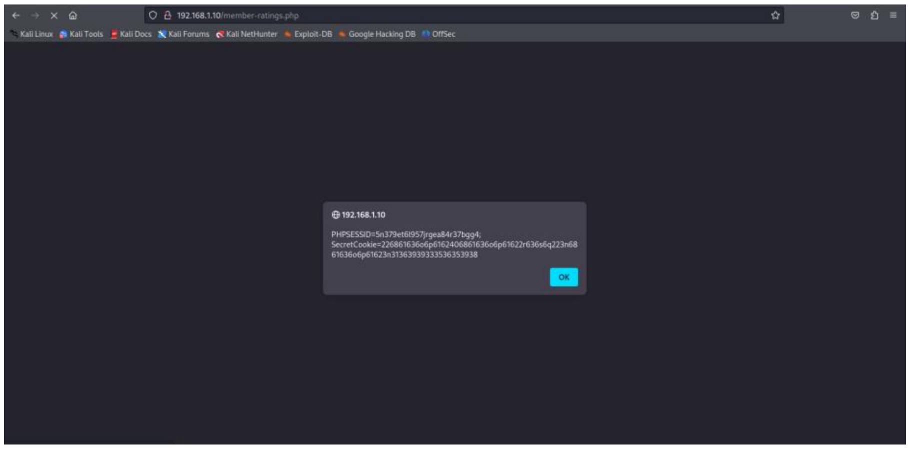

RATE US
Home | Cart[0] | Inbox[0] | Tables | Party-Halls | Rate Us | Logout
Here you can ... For more information Click Here to contact us.
RATE OUR FOODS
Comment
rt(document.cookie);</script>
Rate

Figure 24 - Rate Us XSS Payload

After the payload was submitted, revisiting http://192.168.1.10/member-ratings.php resulted in the execution of the injected script as evidence of an alert box displaying the 'PHPSESSID' cookie value. This confirmed that the payload was successfully reflected and stored within the web applications 'Rate Us' database. This is shown within the screenshot below.



```
PHPSESSID=5n379et6l957jrgea84r37bgg4;
SecretCookie=226861636o6p6162406861636o6p61622r636s6q223n68
61636o6p61623n3136393933353636353938
```

Figure 25 - Reflected XSS Success

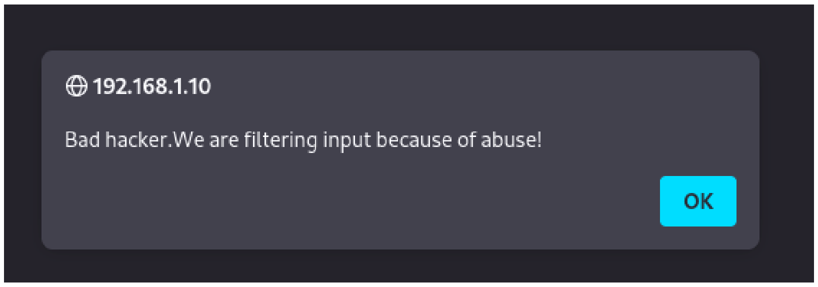
This evidence displays that the web application is vulnerable to Stored Cross-Site Scripting. This form of Cross-Site Scripting is especially dangerous as it involves the storage of malicious scripts within the web applications database which can be accessed by any user visiting http://192.168.1.10/member-ratings.php.

The detection of Stored Cross-Site Scripting means that the website has a critical need to implement input validation and sanitation processes on the server-side. It is recommended that all user inputs be treated as untrusted and be validated correctly.

2.8.3 Testing for SQL Injection

During this phase of the penetration test,the schema.sql file which was previously enumerated was utilised to craft a tailored SQL query to extract sensitive user information from the web applications database. The tailored query payload was '' UNION SELECT NULL, login, passwd FROM members -- -'. This payload was designed to extract usernames and passwords from the 'members' table.

This query was then entered into the user login page for Email and Password. The applications responded by displaying a pop-up alert stating 'Bad hacker. We are filtering input because of abuse!' This result indicates that there is a presence of input validation within the login form. This is shown in the screenshot below.



192.168.1.10

Bad hacker.We are filtering input because of abuse!

Figure 26 - SQL Injection Bad Hacker Response

Additional tests utilising the same SQL query payload were performed within the admin login page within http://192.168.1.10/admin/login-form.php. This endpoint responded identically to the user login page suggesting the same security posture for input validation across the web applications user login pages which yielded the exact same result.

It should be noted that previous testing within section 2.5.4 Testing Authentication Schema, the admin login page was susceptible to a simpler form of SQL injection, specifically the '' OR '1' = '1' statement, allowed for unauthorised access to the web applications administrator control panel. It appears that while more complex SQL payloads are being detected and blocked by web application firewalls, the simpler payload bypasses the input validation.

To protect against this type of attack it is recommended that the input validation encompasses simpler payloads in addition to targeted ones to prevent unauthorised access to the administrator control panel.

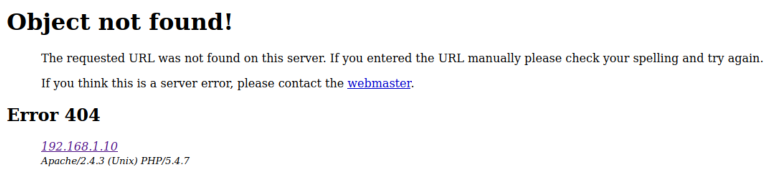
2.9 Error Handling

2.9.1 Analysis of Error Codes

Testing of error codes was conducted to identify how the web application responded to incorrect information being inputted by the user. To begin with, the incorrect URL's were inputted, these URL's are:

- http://192.168.1.10/contact.php
- http://192.168.1.10/products.php

Both URL's returned the same error code, this can be seen in the screenshot below.



Object not found!

The requested URL was not found on this server. If you entered the URL manually please check your spelling and try again.

If you think this is a server error, please contact the webmaster.

Error 404

192.168.1.10
Apache/2.4.3 (Unix) PHP/5.4.7

Figure 27 - Object Not Found Error

In addition to checking incorrect URL's, tests were performed on the user login pages to determine how the web application would respond to incorrect user login credentials for both Email and Password. The following login credentials were inputted into the user login form:

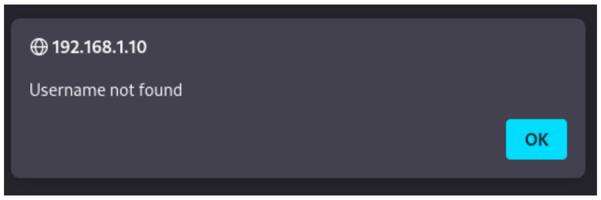
Attempt 1 – Incorrect Email

- **Email:** hacklab@haklab.com
- **Password**: hacklab

Attempt 2 – Incorrect Password

- **Email:** hacklab@hacklab.com
- **Password**: hacklabb

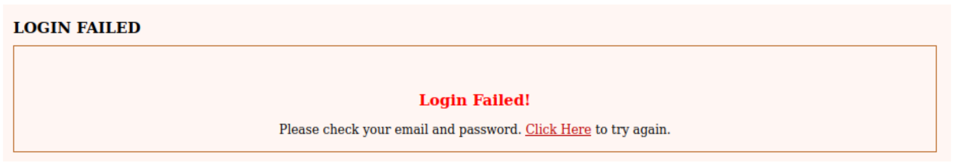
The results of these tests are shown in the screenshots below.



192.168.1.10

Username not found

Figure 28 - Incorrect Email Error



LOGIN FAILED

Login Failed!

Please check your email and password. Click Here to try again.

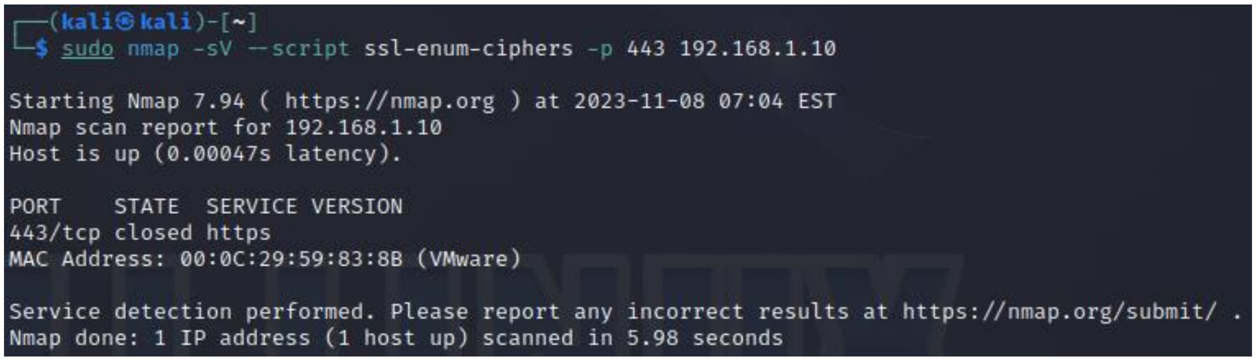
Figure 29 - Incorrect Password Error

2.10 Cryptography

2.10.1 Testing for Weak Transport Layer Security

In section 2.2.1 within the Information Gathering phase of this penetration test it was identified that 192.168.1.10 does not run on HTTPS, which is a significant vulnerability. This means that any data transmitted, including passwords, session tokens and personal information can be intercepted by man-in-the-middle type attacks.

To examine the SSL/TLS configurations further another Nmap scan was performed using the 'ssl-enum-ciphers' script targeting port 443, the standard port for HTTPS transactions.



```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sV --script ssl-enum-ciphers -p 443 192.168.1.10

Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-08 07:04 EST
Nmap scan report for 192.168.1.10
Host is up (0.00047s latency).

PORT    STATE  SERVICE VERSION
443/tcp closed https
MAC Address: 00:0C:29:59:83:8B (VMware)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 5.98 seconds
```

Figure 30 - Nmap ssl-enum-ciphers

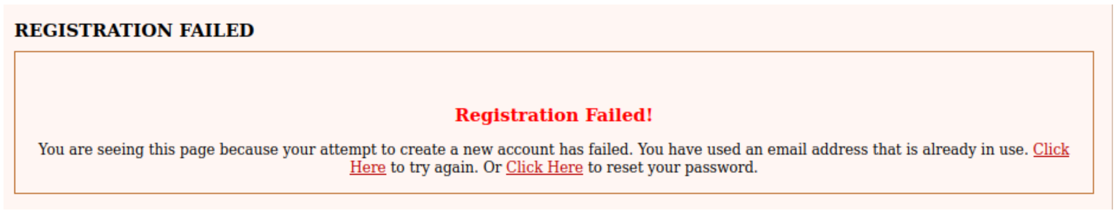
The results of this scan can be seen in the above screenshot. These results confirm that the port 443 is close, reinforcing the initial discovery that the web application lacks any secure transport layer protocols.

2.11 Business Logic Testing

2.11.1 Test Business Logic Data Validation

The Business Logic Data Validation testing will build upon earlier findings outlined in previous stages of the penetration test. A review of the user registration form revealed an absence of any validation mechanisms. It was possible to register an account with an invalid email address. In addition, the registration form lacked any password policy leaving the web application vulnerable to brute force attacks. It should also be noted that the registration form did not verify the match between 'Password' and 'Confirm Password'.

Further investigation into the web applications registration form was performed by testing against duplicate account creation using pre-existing email addresses. An attempt was made to re-register the test account given for the penetration test 'hacklab@hacklab.com' which was met with failure, triggering an error message indicating the form checks against existing entries within the web applications database. This can be seen in the screenshot below.



REGISTRATION FAILED

Registration Failed!

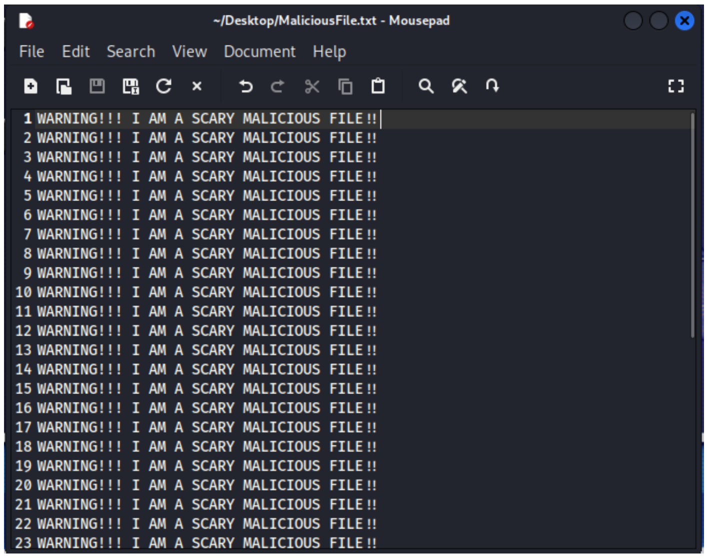
You are seeing this page because your attempt to create a new account has failed. You have used an email address that is already in use. Click Here to try again. Or Click Here to reset your password.

Figure 31 - Registration Failed

Other aspects of the websites input validation were tested and are detailed within the Input Validation Testing section of this report. A significant vulnerability was identified in the 'Rate Us' feature which allows logged-in users can leave comments. This feature was susceptible to stored Cross-Site Scripting indicating a lack of sufficient input validation throughout the web application.

2.11.2 Test Upload of Unexpected File Types

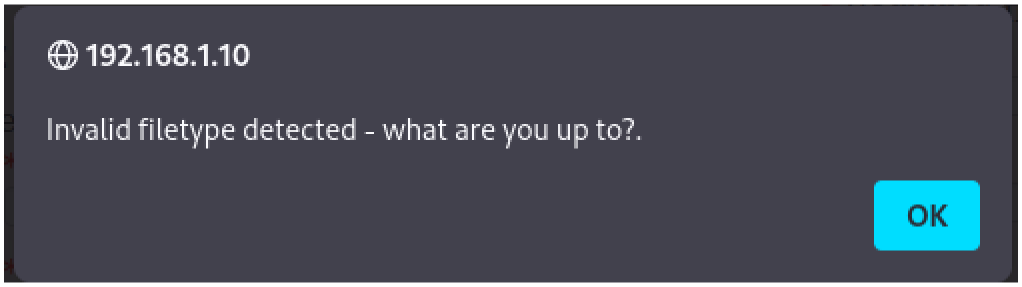
Within the Edit Profile section, logged-in users have the capability to modify their profile picture by uploading a file. This functionality will be tested by making an attempt to upload a non-image file. In this case a text file named 'MaliciousFile.txt' with the contents of plain text reading 'WARNING!!! I AM A SCARY MALICIOUS FILE!!' multiple times. This can be seen in the screenshot below.


~/Desktop/MaliciousFile.txt - Mousepad
File Edit Search View Document Help

```
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
```

Figure 32 - MaliciousFile.txt

This file is then selected to be uploaded as the new profile picture for the test account on 'hacklab@hacklab.com'. The submitted file fails to upload, and an alert prompt is displayed with the message 'Invalid filetype detected – what are you up to?'. This displays a clear indication of file type validation when users upload. Evidence of this prompt is shown below.


192.168.1.10

Invalid filetype detected - what are you up to?.

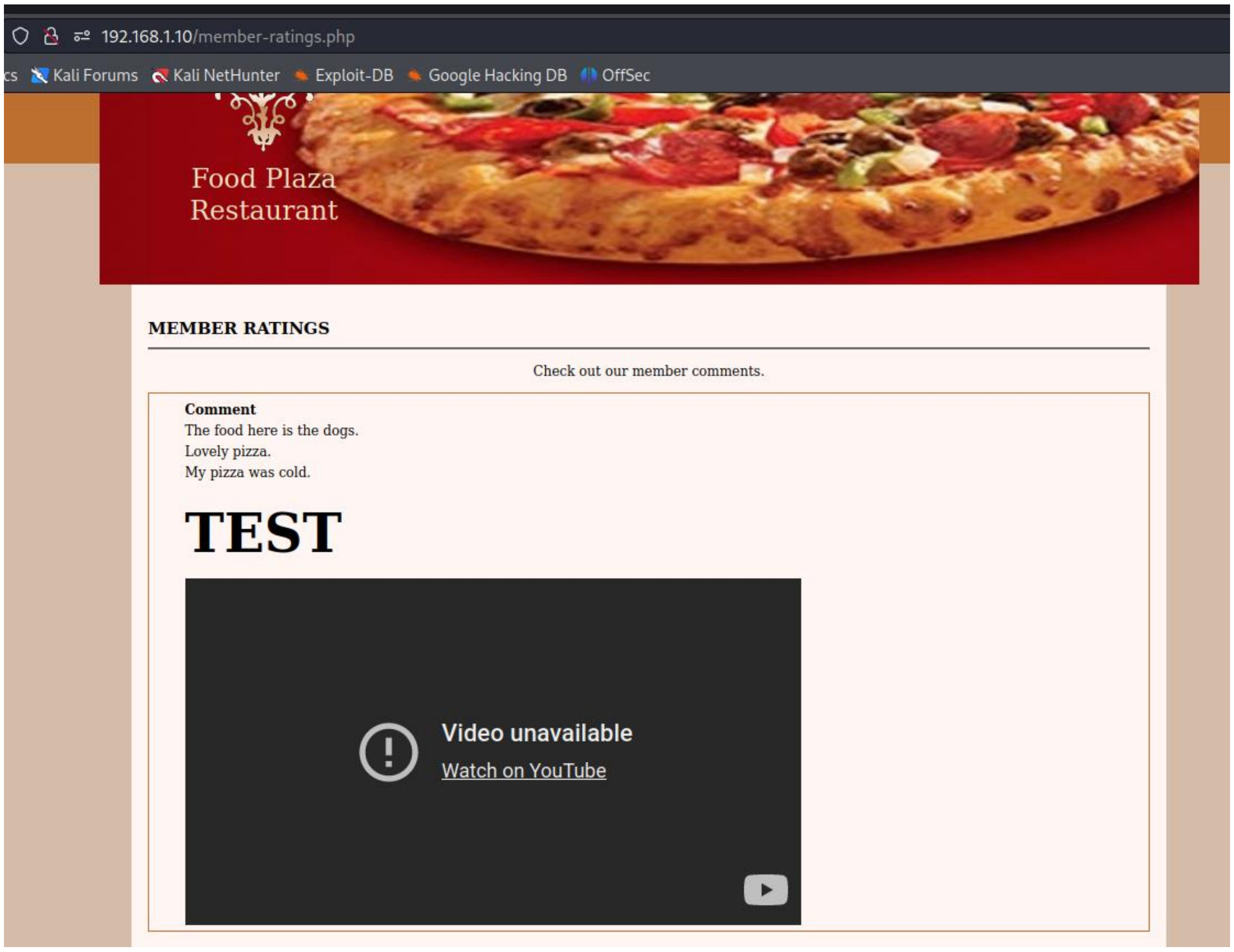
Figure 33 - Invalid Filetype

2.12 Client-Side Testing

2.12.1 Testing for HTML Injection

Earlier in the penetration test, the ‘Rate Us’ feature available to logged-in users was tested for vulnerabilities. These tests identified the feature was susceptable to stored Cross-Site Scripting indicating a potential risk for other injection based exploits. To validate this, a HTML injection test was conducted on the ‘Rate Us’ feature comment form. A simple payload, ‘<h1 style="font-size: 48px; color: black;">Test</h1>’ was submitted which resulted in the display of the word ‘Test’ in a large font on the page http://192.168.1.10/member-ratings.php.

To elaborate on this vulnerability, an attempt was made to embed an external video within the same Members Rating page. The injected code was ‘‘<iframe width="560" height="315" src="https://www.youtube.com/embed/dQw4w9WgXcQ" frameborder="0" allowfullscreen></iframe>’. The outcome was partially successful; the iframe was embedded and while the video content itself was marked as ‘unavailable’ the option for users to navigate to YouTube for viewing remained. This result further demonstrates the application's vulnerability to HTML injection, posing a risk of content spoofing and unwarranted redirections.



MEMBER RATINGS

Check out our member comments.

Comment
The food here is the dogs.
Lovely pizza.
My pizza was cold.

TEST

Video unavailable
Watch on YouTube

Figure 34 - HTML Injection

2.13 Results

This results section presents a detailed account of the vulnerabilities identified within the 'Hacklab Pizza' web application. Each finding is categorised by severity to provide clarity and the urgency along with the potential impact of the issue.

- **Critical findings** represent a significant security risk that could lead to severe consequences such as a data breach or system compromise and require immediate attention.
- **Moderate issues** are less severe but could still affect the applications security if exploited.

2.13.1 Information Gathering

Critical: Open ports and an outdated server were identified using Nmap and WhatWeb revealing significant vulnerabilities (Ports 21, 80, 3306 and Apache 2.4.3, PHP 5.4.7).

Impact: An attacker could exploit the open ports to gain unauthorised access or conduct service specific attacks potentially leading to data breaches or service disruptions.

Recommendation: Prioritise the update of server components the latest available releases and employ a firewall to secure any unnecessary open ports.

2.13.2 Vulnerability Identification

2.13.2.1 Authentication Flaws

Critical: The absence of any email validation and weak password policies were discovered during the User Registration Process (section 2.4.1).

Impact: An attacker could exploit weak password policies by conducting dictionary or brute-force attacks to gain unauthorised access.

Recommendation: Introduce strict validation rules for email format and enforce a robust password policy with complexity requirements and account lockouts after multiple failed attempts.

2.13.2.2 Session Management

Moderate: Insecure cookie handling without 'Secure' flags was identified during the Session Management Schema Testing (section 2.7.1).

Impact: Without 'Secure' flags, session cookies could be intercepted over non-HTTPS connections, allowing attackers to potentially hijack user sessions and access sensitive information.

Recommendation: Assign 'Secure' attributes to cookies and transition to HTTPS to protect session data.

2.13.2.3 Input Validation

Critical: Reflected and stored Cross-Site Scripting (XSS) vulnerabilities were detected throughout Input Validation Testing (sections 2.8.1 and 2.8.2).

Impact: XSS vulnerabilities could allow an attacker to inject malicious scripts, leading to the theft of session cookies, defacement of the website or distribution of malware to users.

Recommendation: Implement suitable sanitisation of user inputs, employ Content Security Policy headers, and encode output data to mitigate XSS risks.

2.13.2.4 SQL Injection
Critical: SQL injection flaws were identified during the SQL Injection Testing (section 2.8.3).

Impact: SQL injections could allow attackers to manipulate database queries leading to unauthorised data disclosure or data loss.

Recommendation: Utilise parameterised queriers and prepared statements for database interactions.

2.13.2.5 HTML Injection
Critical: HTML injection flaws were identified during the Client-Side Testing (section 2.12.1).

Impact: Successful HTML injections could allow attackers to permanently modify the elements on a webpage. This exposes the organisation to various attacks including content spoofing and redirect users to malicious websites. This can compromise user trust, damaging the organisations reputation

Recommendation: Enforce proper input validation and sanitise user inputs to remove HTML content.

2.13.3 Configuration and Deployment
Moderate: Configuration vulnerabilities such as outdated software and missing security headers were identified during the Configuration and Deployment Management testing (section 2.3)

Impact: Poor configuration and outdated components could lead to a system compromise and data breaches severely damaging the organisations reputation.

Recommendation: Apply software updates and configure security headers.

2.13.4 Identity and Access Management
Critical: The lack of any account validation mechanisms and insufficient brute-force protection was observed during the Identity Management Testing (section 2.4).

Impact: Attackers could exploit the lack of any validation mechanism or brute-force protection to gain unauthorised access to user accounts and sensitive data.

Recommendation: Introduce multi-factor authentication, verify accounts through secure methods and set up a policy for account lockouts.

2.13.5 Authentication Testing
Critical: Authentication mechanism bypass vulnerabilities were discovered during Authentication Testing (section 2.5).

Impact: Bypassing authentication mechanisms allows attackers to impersonate administrators, alter system settings and access confidential information which could result in regulatory fines and reputational damage.

Recommendation: Strengthen authentication controls with multi-factor authentication and ongoing monitoring of login activities.

2.13.6 Authorisation Testing

Critical: The web application failed to prevent directory traversal and IDOR vulnerabilities were identified in the Authentication Testing (section 2.6).

Impact: A lack of authorisation checks could allow an attacker to access or modify restricted data, leading to legal implications.

Recommendation: Implement strict role-based access control measures and validate direct object references against authenticated user sessions.

2.13.7 Session Management

Critical: A lack of timeout features was identified during Session Management Testing (section 2.7).

Impact: The absence of any timeouts could allow attackers to maintain permenant access to sessions, even after users believe they have logged out. This could lead to an account takeover.

Recommendation: Enforce session timeouts and rotate session identifiers post-authentication.

2.13.8 Input Validation Testing

Critical: Authentication mechanism bypass vulnerabilities were discovered during Authentication Testing (section 2.5).

Impact: Attackers could exploit a lack of input validation to carry out sophisticated XSS or SQL injection attacks which could compromise the entire user database, disrupt service, and damage the organisations reputation.

Recommendation: Strengthen authentication controls with multi-factor authentication and ongoing monitoring of login activities.

3 DISCUSSION

3.1 OVERALL DISCUSSION

When reflecting upon the outcomes of the web application penetration test which was conducted on 'Hacklab Pizza', it is evident that the projects aim has been successfully achieved. A thorough penetration test was executed using the OWASP web applications penetration testing methodology as a foundation, ensuring the coverage of the OWASP Top 10 vulnerabilities. This approach facilitated the identification of multiple security concerns ranging from server misconfigurations to authentication flaws, which aligns with the projects second aim of identifying and exploiting vulnerabilities.

One notable example of these vulnerabilities was demonstrated through the 'Rate Us' feature. Initially this feature was identified to be susceptible to stored Cross-Site Scripting which was an indicator of other potential injection-based exploits. Future probing with HTML injection revealed the web application was also vulnerable to content spoofing, allowing for an attacker to permanently alter the content on the 'Member Ratings' page.

This penetration tests value extends beyond identifying vulnerabilities. It serves another purpose of pinpointing weaknesses within 'Hacklab Pizza' that require immediate attention and educates the organisation about the implications these vulnerabilities have. As demonstrated in this report, security vulnerabilities can be found within a single flawed line of code among a collection of thousands, making scanning and specialised tools integral to any penetration test. This test therefore acts as a critical component in a larger security strategy.

This report, while providing some immediate solutions, should be viewed as a snapshot at the constantly evolving landscape of cyber security. This means continuous monitoring, regular updates and a proactive security stance are a must for any organisation.

The volume and severity of the vulnerabilities identified within 'Hacklab Pizza' present a significate challenge. Given the gravity of these vulnerabilities, rebuilding the entire web application from the ground up may be a more effective solution than patching each individual vulnerability. While this approach may a have significant financial cost and cause disruption it is a necessary consideration given the extent of the fundamental flows detected.

In conclusion, this penetration test not only underscores the importance of proactive security measures but also highlights the need for a continuous approach to securing web applications.

3.2 Future Work

The penetration test conducted on the web application provided significant insights into several security vulnerabilities, with SQL injection and stored cross-site scripting (XSS) among the most critical. With any future work, the aim would be to delve deeper into these vulnerabilities to understand the full extent of their impact.

For SQL injection, there is a need to explore advanced exploitation techniques that could reveal more sensitive data or even allow complete control over the database server. This would involve crafting more complex SQL injection payloads, possibly automating the attack to identify all the vulnerable points within the application systematically.

In terms of stored XSS, further tests should be conducted to see if it's possible to inject persistent malicious scripts that could target other users of the web application. The aim would be to determine if these scripts can be utilised to gain elevated privileges within the application or to compromise users' data. This could include creating payloads that bypass the client-side filters or using event handlers and advanced JavaScript to execute code in unexpected ways.

Moreover, the implementation of these tests must be done with caution to prevent any real damage to the web application and its users. Ideally these tests would be set up a controlled environment that mirrors the live environment for safe exploitation. Any findings from these advanced tests would be crucial for developing a more robust security framework for the web application.

In conclusion, while the initial testing has laid the groundwork for understanding the web application's security posture, future work would be beneficial by exploring these types of attacks to their fullest potential.

3.3 References

Stuttard, D. and Pinto, M. (2011) 'Web Application Security', in The web application hacker's Handbook: Finding and exploiting security flaws. Indianapolis: Wiley.

OWASP Web Application Security Testing Methodology (no date) GitHub. Available at: https://github.com/OWASP/wstg/tree/master/document/4-Web_Application_Security_Testing (Accessed: 08 November 2023).

Owasp Top Ten (no date) OWASP Top Ten | OWASP Foundation. Available at: https://owasp.org/www-project-top-ten/ (Accessed: 08 November 2023).

Palatty, N.J. (2023) 160 cybersecurity statistics 2023 [updated], Astra Security Blog. Available at: https:// https://www.getastra.com/blog/security-audit/cyber-security-statistics/ (Accessed: 11 November 2023).

4 APPENDICES

APPENDIX A - OWASP ZAP – MAPPING URL’S LIST

```
http://192.168.1.10
http://192.168.1.10/robots.txt
http://192.168.1.10/sitemap.xml
http://192.168.1.10/schema.sql
http://192.168.1.10/index.php
http://192.168.1.10/
http://192.168.1.10/foodzone.php
http://192.168.1.10/member-ratings.php
http://192.168.1.10/member-index.php
http://192.168.1.10/contactus.php
http://192.168.1.10/aboutus.php
http://192.168.1.10/access-denied.php
http://192.168.1.10/appendage.php?type=terms.php
http://192.168.1.10/stylesheets/user_styles.css
http://192.168.1.10/swf/swfobject.js
http://192.168.1.10/images/pizza-inn-map4-mombasa-road.png
http://192.168.1.10/login-register.php
http://192.168.1.10/validation/user.js
http://192.168.1.10/images/img001.png
http://192.168.1.10/cart-exec.php?id=1
http://192.168.1.10/images/img002.png
http://192.168.1.10/cart-exec.php?id=2
http://192.168.1.10/images/img003.png
http://192.168.1.10/cart-exec.php?id=3
http://192.168.1.10/images/img004.png
http://192.168.1.10/cart-exec.php?id=4
http://192.168.1.10/images/img005.png
http://192.168.1.10/cart-exec.php?id=5
http://192.168.1.10/images/img006.png
http://192.168.1.10/cart-exec.php?id=6
http://192.168.1.10/images/img007.png
http://192.168.1.10/cart-exec.php?id=7
http://192.168.1.10/images/img008.png
http://192.168.1.10/cart-exec.php?id=8
http://192.168.1.10/images/img009.png
http://192.168.1.10/cart-exec.php?id=9
http://192.168.1.10/images/img010.png
http://192.168.1.10/cart-exec.php?id=10
http://192.168.1.10/images/img011.png
http://192.168.1.10/cart-exec.php?id=11
http://192.168.1.10/images/img012.png
http://192.168.1.10/cart-exec.php?id=12
http://192.168.1.10/images/img013.png
http://192.168.1.10/login-exec.php
http://192.168.1.10/cart-exec.php?id=13
http://192.168.1.10/images/img014.png
```

```
http://192.168.1.10/cart-exec.php?id=14
http://192.168.1.10/images/img015.png
http://192.168.1.10/cart-exec.php?id=15
http://192.168.1.10/images/img016.png
http://192.168.1.10/cart-exec.php?id=16
http://192.168.1.10/images/img017.png
http://192.168.1.10/cart-exec.php?id=17
http://192.168.1.10/images/img018.png
http://192.168.1.10/register-exec.php
http://192.168.1.10/cart-exec.php?id=18
http://192.168.1.10/images/img019.png
http://192.168.1.10/cart-exec.php?id=19
http://192.168.1.10/images/img020.png
http://192.168.1.10/cart-exec.php?id=20
http://192.168.1.10/images/img021.png
http://192.168.1.10/cart-exec.php?id=21
http://192.168.1.10/images/img022.png
http://192.168.1.10/cart-exec.php?id=22
http://192.168.1.10/images/img023.png
http://192.168.1.10/cart-exec.php?id=23
http://192.168.1.10/images/img024.png
http://192.168.1.10/cart-exec.php?id=24
http://192.168.1.10/images/img025.png
http://192.168.1.10/cart-exec.php?id=25
http://192.168.1.10/foodzone.php
http://192.168.1.10/register-success.php
```

APPENDIX B - DIRB – BRUTE FORCE ATTACK RESULTS

```
---- Scanning URL: http://192.168.1.10/ ----
+ http://192.168.1.10/admin/ (CODE:302|SIZE:0)
+ http://192.168.1.10/install/ (CODE:200|SIZE:893)
+ http://192.168.1.10/js/ (CODE:200|SIZE:7130)
==> DIRECTORY: http://192.168.1.10/admin/
+ http://192.168.1.10/images/ (CODE:200|SIZE:7944)
+ http://192.168.1.10/cgi-bin/ (CODE:403|SIZE:989)
+ http://192.168.1.10/index.php (CODE:200|SIZE:5982)
+ http://192.168.1.10/css/ (CODE:200|SIZE:2812)
+ http://192.168.1.10/error/ (CODE:403|SIZE:989)
+ http://192.168.1.10/?q=user/register/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=user/password/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=user/login/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=search/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=node/add/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=comment/reply/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=admin/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/auth.php (CODE:302|SIZE:0)
+ http://192.168.1.10/temp/ (CODE:200|SIZE:884)
+ http://192.168.1.10/?q=user/logout/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=filter/tips/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?level=12 (CODE:200|SIZE:5982)
+ http://192.168.1.10/? (CODE:200|SIZE:5982)
+ http://192.168.1.10/?&what= (CODE:200|SIZE:5982)
==> DIRECTORY: http://192.168.1.10/install/
+ http://192.168.1.10/index.php/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/gallery.php (CODE:200|SIZE:1291)
+ http://192.168.1.10/?q=logout/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=contact/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=* (CODE:200|SIZE:5982)
+ http://192.168.1.10/?msk (CODE:200|SIZE:5982)
+ http://192.168.1.10/?favorites (CODE:200|SIZE:5982)
+ http://192.168.1.10/?* (CODE:200|SIZE:5982)
+ http://192.168.1.10/robots.txt (CODE:200|SIZE:36)

---- Entering directory: http://192.168.1.10/admin/ ----
+ http://192.168.1.10/admin/index.php (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/?q=user/register/ (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/?q=user/password/ (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/?q=user/login/ (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/?q=search/ (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/?q=node/add/ (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/?q=comment/reply/ (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/?q=admin/ (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/auth.php (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/?q=user/logout/ (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/?q=filter/tips/ (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/?level=12 (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/? (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/?&what= (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/profile.php (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/index.php/ (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/?q=logout/ (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/?q=contact/ (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/?q=* (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/?msk (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/?favorites (CODE:302|SIZE:0)
+ http://192.168.1.10/admin/?* (CODE:302|SIZE:0)
```

APPENDIX C – SCHEMA.SQL

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

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
```

```
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `billing_details`
--

DROP TABLE IF EXISTS `billing_details`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `billing_details` (
  `billing_id` int(10) NOT NULL AUTO_INCREMENT,
  `member_id` int(15) NOT NULL,
  `Street_Address` varchar(100) NOT NULL,
  `P_O_Box_No` varchar(15) NOT NULL,
  `City` text NOT NULL,
  `Mobile_No` varchar(15) NOT NULL,
  `Landline_No` varchar(15) DEFAULT NULL,
  PRIMARY KEY (`billing_id`)
) ENGINE=MyISAM AUTO_INCREMENT=9 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `cart_details`
--

DROP TABLE IF EXISTS `cart_details`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `cart_details` (
  `cart_id` int(15) NOT NULL AUTO_INCREMENT,
  `member_id` int(15) NOT NULL,
  `food_id` int(15) NOT NULL,
  `quantity_id` int(15) NOT NULL,
  `total` float NOT NULL,
  `flag` int(1) NOT NULL,
  PRIMARY KEY (`cart_id`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `categories`
--

DROP TABLE IF EXISTS `categories`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
```

```
CREATE TABLE `categories` (
  `category_id` int(15) NOT NULL AUTO_INCREMENT,
  `category_name` varchar(45) NOT NULL,
  PRIMARY KEY (`category_id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `currencies`
--

DROP TABLE IF EXISTS `currencies`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `currencies` (
  `currency_id` int(5) NOT NULL AUTO_INCREMENT,
  `currency_symbol` varchar(15) NOT NULL,
  `flag` int(1) NOT NULL,
  PRIMARY KEY (`currency_id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `food_details`
--

DROP TABLE IF EXISTS `food_details`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `food_details` (
  `food_id` int(15) NOT NULL AUTO_INCREMENT,
  `food_name` varchar(45) NOT NULL,
  `food_description` text NOT NULL,
  `food_price` float NOT NULL,
  `food_photo` varchar(45) NOT NULL,
  `food_category` int(15) NOT NULL,
  PRIMARY KEY (`food_id`)
) ENGINE=InnoDB AUTO_INCREMENT=26 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `members`
--

DROP TABLE IF EXISTS `members`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `members` (
  `member_id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `firstname` varchar(100) DEFAULT NULL,
  `lastname` varchar(100) DEFAULT NULL,
  `login` varchar(100) NOT NULL DEFAULT '',
  `passwd` varchar(32) NOT NULL DEFAULT '',
  `question_id` int(5) NOT NULL,
  `answer` varchar(45) NOT NULL,
  `thumbnail` varchar(100) NOT NULL,
  PRIMARY KEY (`member_id`)
) ENGINE=MyISAM AUTO_INCREMENT=18 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `messages`
--

DROP TABLE IF EXISTS `messages`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `messages` (
  `message_id` int(15) NOT NULL AUTO_INCREMENT,
  `message_from` varchar(25) NOT NULL,
  `message_date` date NOT NULL,
  `message_time` time NOT NULL,
  `message_subject` text NOT NULL,
  `message_text` text NOT NULL,
  PRIMARY KEY (`message_id`)
) ENGINE=MyISAM AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `orders_details`
--

DROP TABLE IF EXISTS `orders_details`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `orders_details` (
  `order_id` int(10) NOT NULL AUTO_INCREMENT,
  `member_id` int(10) NOT NULL,
  `billing_id` int(10) NOT NULL,
  `cart_id` int(15) NOT NULL,
  `delivery_date` date NOT NULL,
  `StaffID` int(15) NOT NULL,
  `flag` int(1) NOT NULL,
  `time_stamp` time NOT NULL,
  PRIMARY KEY (`order_id`)
) ENGINE=MyISAM AUTO_INCREMENT=22 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `partyhalls`
--

DROP TABLE IF EXISTS `partyhalls`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
```

```
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `partyhalls` (
  `partyhall_id` int(5) NOT NULL AUTO_INCREMENT,
  `partyhall_name` varchar(45) NOT NULL,
  PRIMARY KEY (`partyhall_id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `pizza_admin`
--

DROP TABLE IF EXISTS `pizza_admin`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `pizza_admin` (
  `Admin_ID` int(45) NOT NULL AUTO_INCREMENT,
  `Username` varchar(45) NOT NULL,
  `Password` varchar(45) NOT NULL,
  PRIMARY KEY (`Admin_ID`)
) ENGINE=MyISAM AUTO_INCREMENT=3 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `polls_details`
--

DROP TABLE IF EXISTS `polls_details`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `polls_details` (
  `poll_id` int(15) NOT NULL AUTO_INCREMENT,
  `member_id` int(15) NOT NULL,
  `food_id` int(15) NOT NULL,
  `rate_id` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`poll_id`)
) ENGINE=MyISAM AUTO_INCREMENT=28 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `quantities`
--

DROP TABLE IF EXISTS `quantities`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `quantities` (
  `quantity_id` int(5) NOT NULL AUTO_INCREMENT,
  `quantity_value` int(5) NOT NULL,
  PRIMARY KEY (`quantity_id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;
```

```
--
-- Table structure for table `questions`
--

DROP TABLE IF EXISTS `questions`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `questions` (
  `question_id` int(5) NOT NULL AUTO_INCREMENT,
  `question_text` text NOT NULL,
  PRIMARY KEY (`question_id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `ratings`
--

DROP TABLE IF EXISTS `ratings`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `ratings` (
  `rate_id` int(5) NOT NULL AUTO_INCREMENT,
  `rate_name` varchar(15) NOT NULL,
  PRIMARY KEY (`rate_id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `reservations_details`
--

DROP TABLE IF EXISTS `reservations_details`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `reservations_details` (
  `ReservationID` int(15) NOT NULL AUTO_INCREMENT,
  `member_id` int(15) NOT NULL,
  `table_id` int(5) NOT NULL,
  `partyhall_id` int(5) NOT NULL,
  `Reserve_Date` date NOT NULL,
  `Reserve_Time` time NOT NULL,
  `StaffID` int(15) NOT NULL,
  `flag` int(1) NOT NULL,
  `table_flag` int(1) NOT NULL,
  `partyhall_flag` int(1) NOT NULL,
  PRIMARY KEY (`ReservationID`)
) ENGINE=MyISAM AUTO_INCREMENT=8 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
```

```
-- Table structure for table `specials`
--

DROP TABLE IF EXISTS `specials`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `specials` (
  `special_id` int(15) NOT NULL AUTO_INCREMENT,
  `special_name` varchar(25) NOT NULL,
  `special_description` text NOT NULL,
  `special_price` float NOT NULL,
  `special_start_date` date NOT NULL,
  `special_end_date` date NOT NULL,
  `special_photo` varchar(45) NOT NULL,
  PRIMARY KEY (`special_id`)
) ENGINE=MyISAM AUTO_INCREMENT=8 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `staff`
--

DROP TABLE IF EXISTS `staff`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `staff` (
  `StaffID` int(15) NOT NULL AUTO_INCREMENT,
  `firstname` varchar(25) NOT NULL,
  `lastname` varchar(25) NOT NULL,
  `Street_Address` text NOT NULL,
  `Mobile_Tel` varchar(20) NOT NULL,
  PRIMARY KEY (`StaffID`)
) ENGINE=MyISAM AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `tables`
--

DROP TABLE IF EXISTS `tables`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `tables` (
  `table_id` int(5) NOT NULL AUTO_INCREMENT,
  `table_name` varchar(45) NOT NULL,
  PRIMARY KEY (`table_id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Table structure for table `timezones`
--
```

```
DROP TABLE IF EXISTS `timezones`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `timezones` (
  `timezone_id` int(5) NOT NULL AUTO_INCREMENT,
  `timezone_reference` varchar(45) NOT NULL,
  `flag` int(1) NOT NULL,
  PRIMARY KEY (`timezone_id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

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

DROP TABLE IF EXISTS `users`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `users` (
  `id` smallint(6) NOT NULL AUTO_INCREMENT,
  `username` varchar(30) NOT NULL,
  `password` varchar(32) NOT NULL,
  `email` varchar(100) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=3 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

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

-- Dump completed on 2023-09-21 10:26:51
```

APPENDIX D – NIKTO CONFIGURATION SCAN RESULTS

```
- Nikto v2.5.0

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

+ Target IP:          192.168.1.10

+ Target Hostname:    192.168.1.10

+ Target Port:        80

+ Start Time:         2023-10-31 06:50:52 (GMT-4)
```

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

+ Server: Apache/2.4.3 (Unix) PHP/5.4.7

+ /: Retrieved x-powered-by header: PHP/5.4.7.

+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-
US/docs/Web/HTTP/Headers/X-Frame-Options

+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content
of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-
scanner/vulnerabilities/missing-content-type-header/

+ /robots.txt: Entry '/schema.sql' is returned a non-forbidden or redirect HTTP code (200). See:
https://portswigger.net/kb/issues/00600600_robots-txt-file

+ Apache/2.4.3 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for
the 2.x branch.

+ PHP/5.4.7 appears to be outdated (current is at least 8.1.5), PHP 7.4.28 for the 7.4 branch.

+ /index: Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute
force file names. The following alternatives for 'index' were found: HTTP_NOT_FOUND.html.var,
HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
HTTP_NOT_FOUND.html.var. See:
http://www.wisec.it/sectou.php?id=4698ebdc59d15,https://exchange.xforce.ibmcloud.com/vulnerabilit
ies/8275

+ /: Web Server returns a valid response with junk HTTP methods which may cause false positives.

+ /: HTTP TRACE method is active which suggests the host is vulnerable to XST. See:
https://owasp.org/www-community/attacks/Cross_Site_Tracing

+ /cgi-bin/printenv: Site appears vulnerable to the 'shellshock' vulnerability. See:
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271

+ PHP/5.4 - PHP 3/4/5 and 7.0 are End of Life products without support.

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

+ /admin/: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla.org/en-
US/docs/Web/HTTP/Cookies

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

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

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

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

+ /css/: Directory indexing found.

+ /css/: This might be interesting.

+ /install/: Directory indexing found.

+ /install/: This might be interesting.

+ /stylesheets/: Directory indexing found.

+ /stylesheets/: This might be interesting.

+ /temp/: Directory indexing found.

+ /temp/: This might be interesting.

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

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

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

+ /icons/: Directory indexing found.

+ /images/: Directory indexing found.

+ /docs/: Directory indexing found.

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

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

+ 9868 requests: 0 error(s) and 33 item(s) reported on remote host

+ End Time:          2023-10-31 06:51:21 (GMT-4) (29 seconds)

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

+ 1 host(s) tested
```

APPENDIX E – CAPTURED INFORMATION FOR SESSION HIJACKING

HTTP/1.1 302 Found

Date: Mon, 06 Nov 2023 11:17:53 GMT

Server: Apache/2.4.3 (Unix) PHP/5.4.7

X-Powered-By: PHP/5.4.7

Set-Cookie: PHPSESSID=t6486f07n3m4dqe4cajoee52f2; path=/

Expires: Thu, 19 Nov 1981 08:52:00 GMT

Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0

Pragma: no-cache

Set-Cookie: SecretCookie=2274657374696r67223n74657374696r673n31363939323639343733

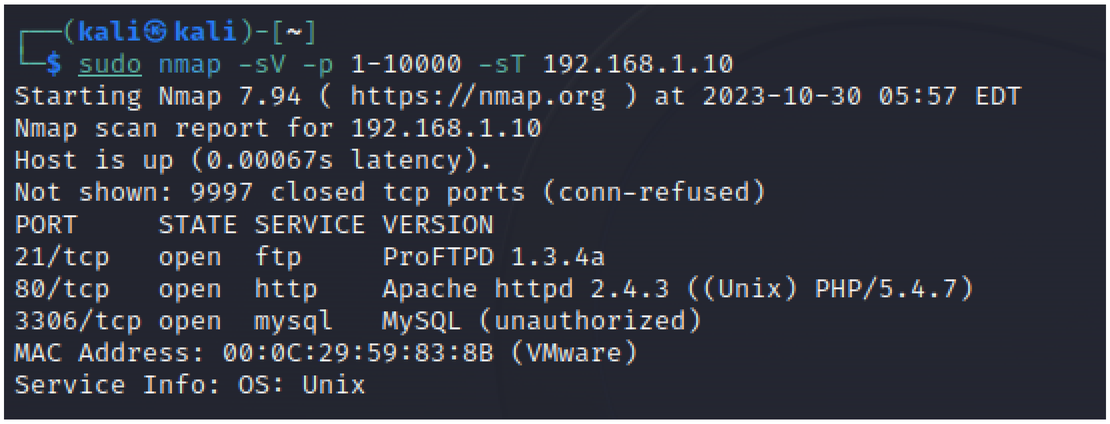
Set-Cookie: PHPSESSID=dauikhoiv0nlv2ubu2a17hbbr6; path=/

location: member-index.php

Content-Length: 0

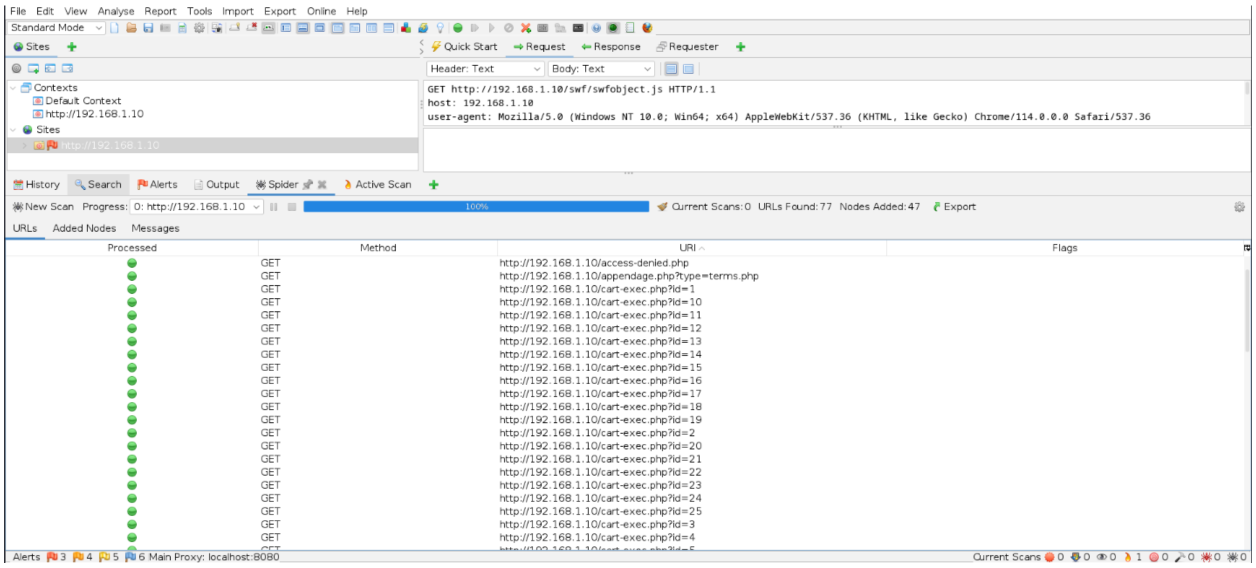
Keep-Alive: timeout=5, max=100

APPENDIX D – FIGURES



```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sV -p 1-10000 -sT 192.168.1.10
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-30 05:57 EDT
Nmap scan report for 192.168.1.10
Host is up (0.00067s latency).
Not shown: 9997 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
21/tcp   open  ftp     ProFTPD 1.3.4a
80/tcp   open  http    Apache httpd 2.4.3 ((Unix) PHP/5.4.7)
3306/tcp open  mysql   MySQL (unauthorized)
MAC Address: 00:0C:29:59:83:8B (VMware)
Service Info: OS: Unix
```

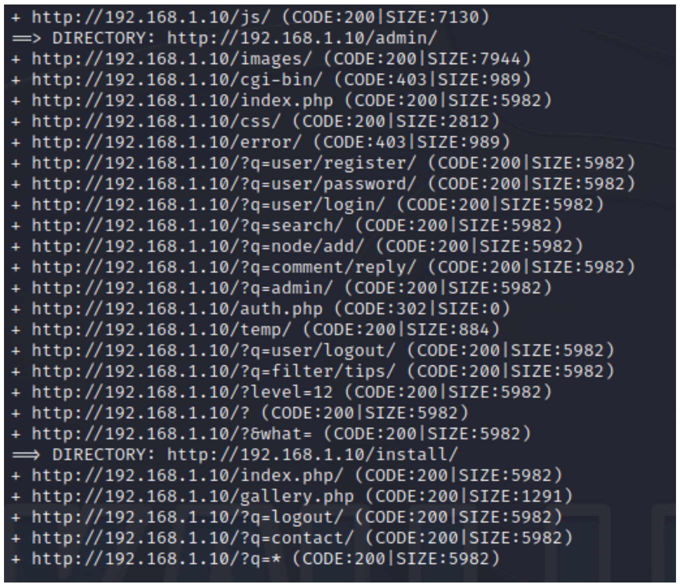
Figure 135 - Nmap Scan



```
GET http://192.168.1.10/swf/swfobject.js HTTP/1.1
host: 192.168.1.10
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.36
```

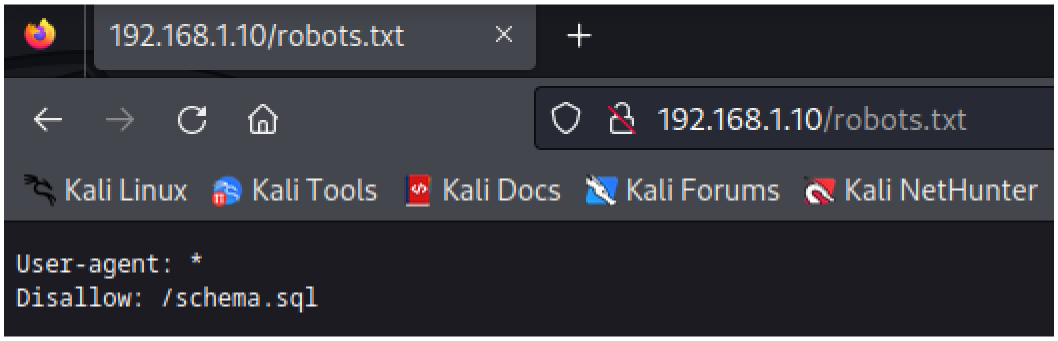
Processed	Method	URI	Flags
●	GET	http://192.168.1.10/access-denied.php	
●	GET	http://192.168.1.10/appendage.php?type=terms.php	
●	GET	http://192.168.1.10/cart-exec.php?id=1	
●	GET	http://192.168.1.10/cart-exec.php?id=10	
●	GET	http://192.168.1.10/cart-exec.php?id=11	
●	GET	http://192.168.1.10/cart-exec.php?id=12	
●	GET	http://192.168.1.10/cart-exec.php?id=13	
●	GET	http://192.168.1.10/cart-exec.php?id=14	
●	GET	http://192.168.1.10/cart-exec.php?id=15	
●	GET	http://192.168.1.10/cart-exec.php?id=16	
●	GET	http://192.168.1.10/cart-exec.php?id=17	
●	GET	http://192.168.1.10/cart-exec.php?id=18	
●	GET	http://192.168.1.10/cart-exec.php?id=19	
●	GET	http://192.168.1.10/cart-exec.php?id=2	
●	GET	http://192.168.1.10/cart-exec.php?id=20	
●	GET	http://192.168.1.10/cart-exec.php?id=21	
●	GET	http://192.168.1.10/cart-exec.php?id=22	
●	GET	http://192.168.1.10/cart-exec.php?id=23	
●	GET	http://192.168.1.10/cart-exec.php?id=24	
●	GET	http://192.168.1.10/cart-exec.php?id=25	
●	GET	http://192.168.1.10/cart-exec.php?id=3	
●	GET	http://192.168.1.10/cart-exec.php?id=4	

Figure 36 - OWASP Zap Mapping



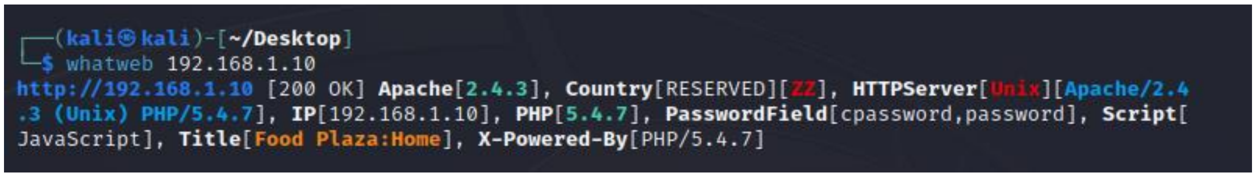
```
+ http://192.168.1.10/js/ (CODE:200|SIZE:7130)
==> DIRECTORY: http://192.168.1.10/admin/
+ http://192.168.1.10/images/ (CODE:200|SIZE:7944)
+ http://192.168.1.10/cgi-bin/ (CODE:403|SIZE:989)
+ http://192.168.1.10/index.php (CODE:200|SIZE:5982)
+ http://192.168.1.10/css/ (CODE:200|SIZE:2812)
+ http://192.168.1.10/error/ (CODE:403|SIZE:989)
+ http://192.168.1.10/?q=user/register/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=user/password/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=user/login/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=search/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=node/add/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=comment/reply/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=admin/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/auth.php (CODE:302|SIZE:0)
+ http://192.168.1.10/temp/ (CODE:200|SIZE:884)
+ http://192.168.1.10/?q=user/logout/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=filter/tips/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?level=12 (CODE:200|SIZE:5982)
+ http://192.168.1.10/? (CODE:200|SIZE:5982)
+ http://192.168.1.10/?&what= (CODE:200|SIZE:5982)
==> DIRECTORY: http://192.168.1.10/install/
+ http://192.168.1.10/index.php/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/gallery.php (CODE:200|SIZE:1291)
+ http://192.168.1.10/?q=logout/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=contact/ (CODE:200|SIZE:5982)
+ http://192.168.1.10/?q=* (CODE:200|SIZE:5982)
```

Figure 37 - Dirb Brute Force Attack



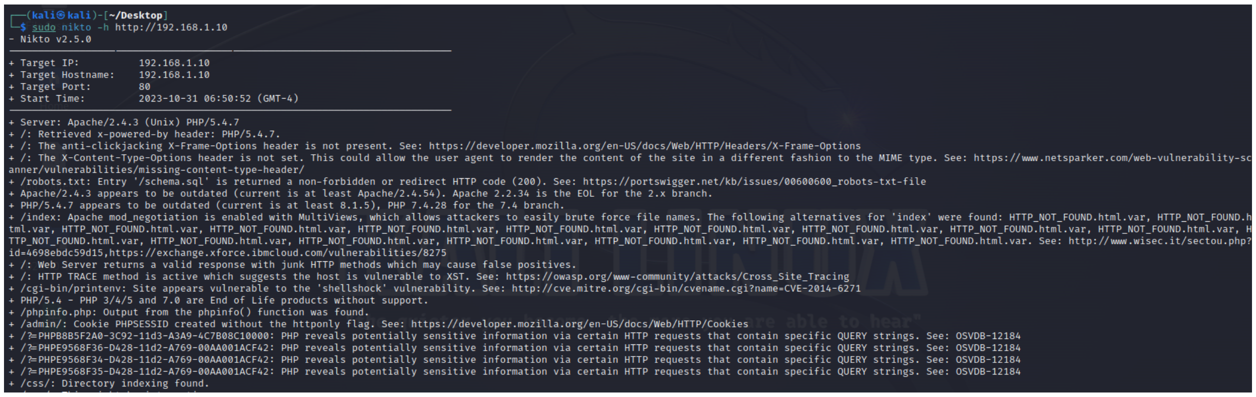
```
User-agent: *
Disallow: /schema.sql
```

Figure 38 - 192.168.1.10/robots.txt



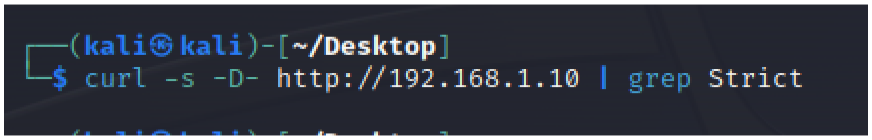
```
┌──(kali㉿kali)-[~/Desktop]
└─$ whatweb 192.168.1.10
http://192.168.1.10 [200 OK] Apache[2.4.3], Country[RESERVED][ZZ], HTTPServer[Unix][Apache/2.4
.3 (Unix) PHP/5.4.7], IP[192.168.1.10], PHP[5.4.7], PasswordField[cpassword,password], Script[
JavaScript], Title[Food Plaza:Home], X-Powered-By[PHP/5.4.7]
```

Figure 39 - whatweb results



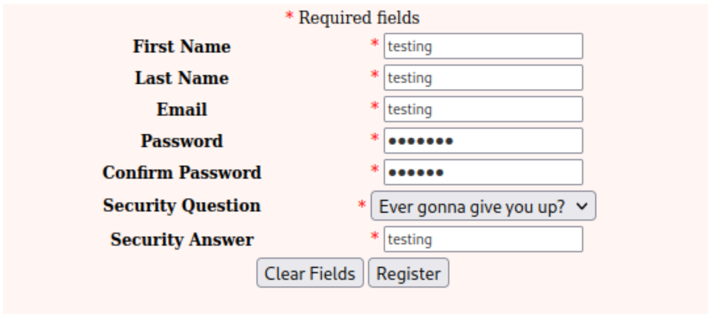
```
┌──(kali㉿kali)-[~/Desktop]
└─$ sudo nikto -h http://192.168.1.10
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          192.168.1.10
+ Target Hostname:    192.168.1.10
+ Target Port:        80
+ Start Time:         2023-10-31 06:50:52 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.3 (Unix) PHP/5.4.7
+ /: Retrieved x-powered-by header: PHP/5.4.7.
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-sc
anner/vulnerabilities/missing-content-type-header/
+ /robots.txt: Entry '/schema.sql' is returned a non-forbidden or redirect HTTP code (200). See: https://portswigger.net/kb/issues/00600600_robots-txt-file
+ Apache/2.4.3 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ PHP/5.4.7 appears to be outdated (current is at least 8.1.5), PHP 7.4.28 for the 7.4 branch.
+ /index: Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. The following alternatives for 'index' were found: HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.h
tml.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, H
TTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var. See: http://www.wisec.it/sectou.php?
id=4698ebdc59d15,https://exchange.xforce.ibmcloud.com/vulnerabilities/8275
+ /: Web Server returns a valid response with junk HTTP methods which may cause false positives.
+ /: HTTP TRACE method is active which suggests the host is vulnerable to XST. See: https://owasp.org/www-community/attacks/Cross_Site_Tracing
+ /cgi-bin/printenv: Site appears vulnerable to the 'shellshock' vulnerability. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271
+ PHP/5.4 - PHP 3/4/5 and 7.0 are End of Life products without support.
+ /phpinfo.php: Output from the phpinfo() function was found.
+ /admin/: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
+ /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specific QUERY strings. See: OSVDB-12184
+ /css/: Directory indexing found.
```

Figure 40 - Nikto Scan



```
┌──(kali㉿kali)-[~/Desktop]
└─$ curl -s -D- http://192.168.1.10 | grep Strict
```

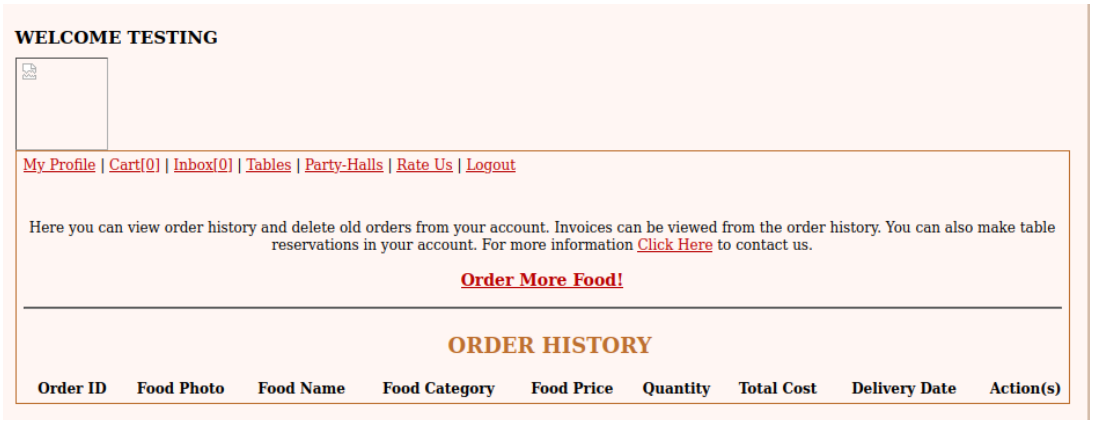
Figure 41 - HSTS testing



* Required fields

First Name	* testing
Last Name	* testing
Email	* testing
Password	* •••••••
Confirm Password	* ••••••
Security Question	* Ever gonna give you up?
Security Answer	* testing

Figure 42 - User Registration Process Test



WELCOME TESTING

My Profile | Cart[0] | Inbox[0] | Tables | Party-Halls | Rate Us | Logout

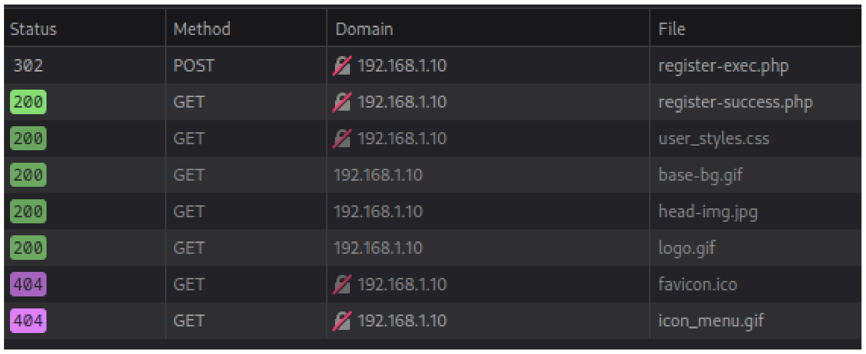
Here you can view order history and delete old orders from your account. Invoices can be viewed from the order history. You can also make table reservations in your account. For more information Click Here to contact us.

Order More Food!

ORDER HISTORY

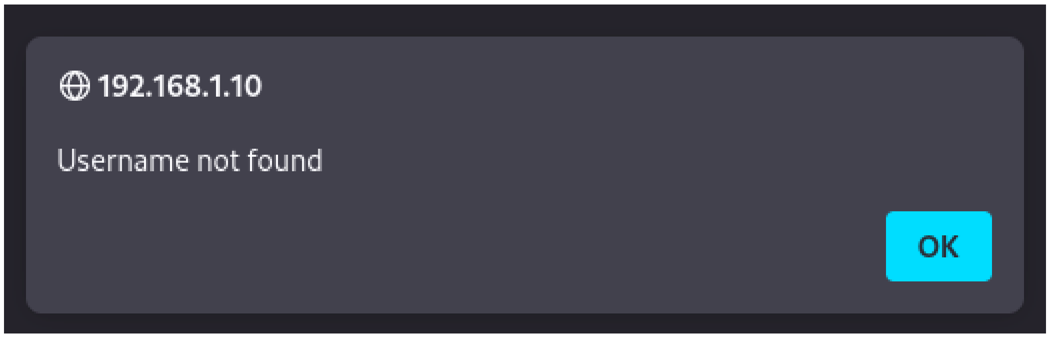
Order ID	Food Photo	Food Name	Food Category	Food Price	Quantity	Total Cost	Delivery Date	Action(s)

Figure 43 - Created User Login



Status	Method	Domain	File
302	POST	192.168.1.10	register-exec.php
200	GET	192.168.1.10	register-success.php
200	GET	192.168.1.10	user_styles.css
200	GET	192.168.1.10	base-bg.gif
200	GET	192.168.1.10	head-img.jpg
200	GET	192.168.1.10	logo.gif
404	GET	192.168.1.10	favicon.ico
404	GET	192.168.1.10	icon_menu.gif

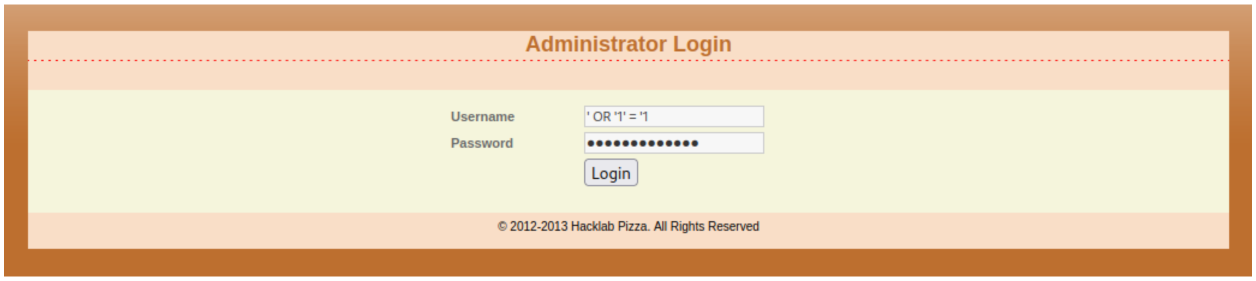
Figure 44 - HTTP User Credentials



192.168.1.10

Username not found

Figure 45 - Username not found.



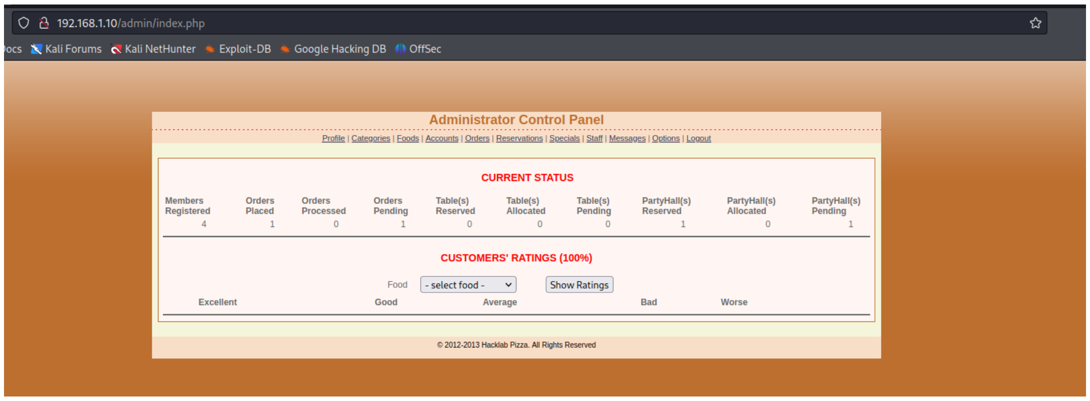
Administrator Login

Username: ' OR '1' = '1

Password: ••••••••••••••

© 2012-2013 Hacklab Pizza. All Rights Reserved

Figure 46 - Admin Login SQL Injection



192.168.1.10/admin/index.php

Administrator Control Panel

Profile | Categories | Foods | Accounts | Orders | Reservations | Specials | Staff | Messages | Options | Logout

CURRENT STATUS

Members Registered	Orders Placed	Orders Processed	Orders Pending	Table(s) Reserved	Table(s) Allocated	Table(s) Pending	PartyHall(s) Reserved	PartyHall(s) Allocated	PartyHall(s) Pending
4	1	0	1	0	0	0	1	0	1

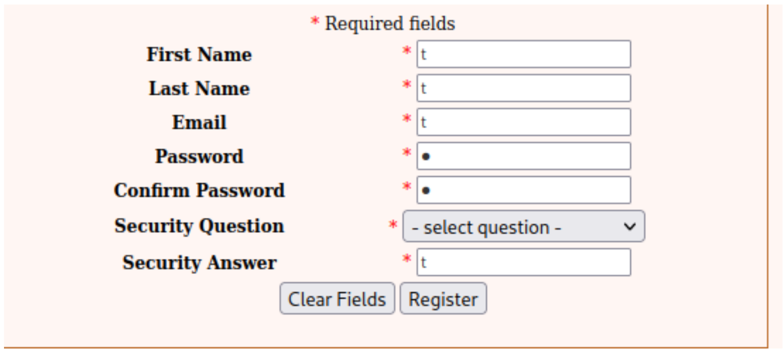
CUSTOMERS' RATINGS (100%)

Food: - select food -

Excellent	Good	Average	Bad	Worse

© 2012-2013 Hacklab Pizza. All Rights Reserved

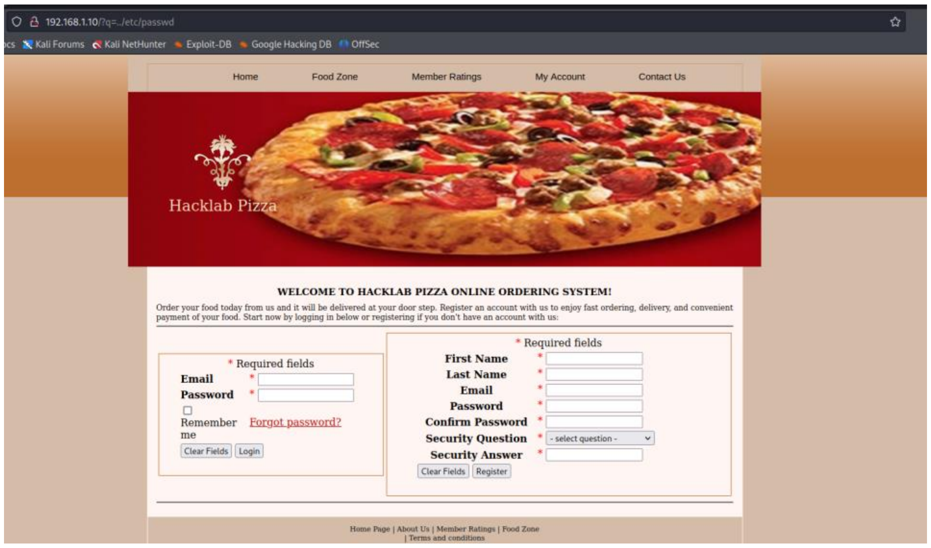
Figure 47 - Admin Control Panel Access



* Required fields

First Name *	t
Last Name *	t
Email *	t
Password *	•
Confirm Password *	•
Security Question *	- select question -
Security Answer *	t

Figure 48 - Weak Password Policy



192.168.1.10/?q=../etc/passwd

Home | Food Zone | Member Ratings | My Account | Contact Us

Hacklab Pizza

WELCOME TO HACKLAB PIZZA ONLINE ORDERING SYSTEM!

Order your food today from us and it will be delivered at your door step. Register an account with us to enjoy fast ordering, delivery, and convenient payment of your food. Start now by logging in below or registering if you don't have an account with us:

* Required fields

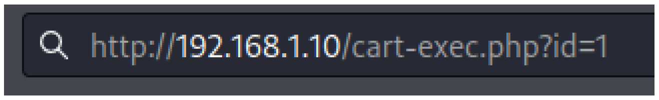
Email *
Password *
Remember me
Forgot password?

* Required fields

First Name *
Last Name *
Email *
Password *
Confirm Password *
Security Question * - select question -
Security Answer *

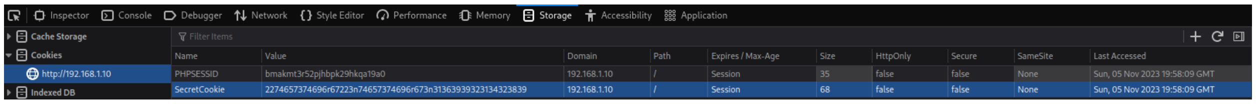
Home Page | About Us | Member Ratings | Food Zone | Terms and conditions

Figure 49 - 192.168.1.10/?q=../etc/passwd



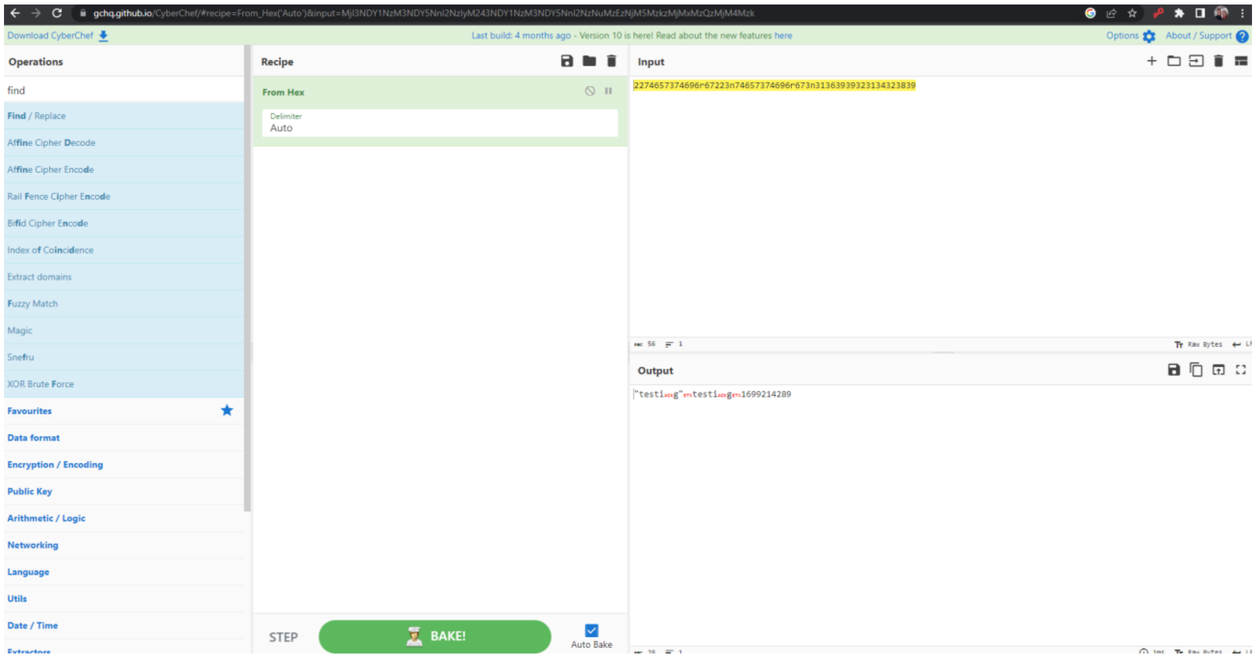
http://192.168.1.10/cart-exec.php?id=1

Figure 50 - IDOR Attack Vector



Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
PHPSESSID	bmakmt3r52pjhbpk29hkqa19a0	192.168.1.10	/	Session	35	false	false	None	Sun, 05 Nov 2023 19:58:09 GMT
SecretCookie	2274657374696r67223n74657374696r673n31363939323134323839	192.168.1.10	/	Session	68	false	false	None	Sun, 05 Nov 2023 19:58:09 GMT

Figure 51 - Login Cookies



```
gchq.github.io/CyberChef/#recipe=From_Hex('Auto')&input=MjI3NDY1NzM3NDY5NnI2NzIyM243NDY1NzM3NDY5NnI2NzNuMzEzNjM5MzkzMjMxMzQzMjM4Mzk
```

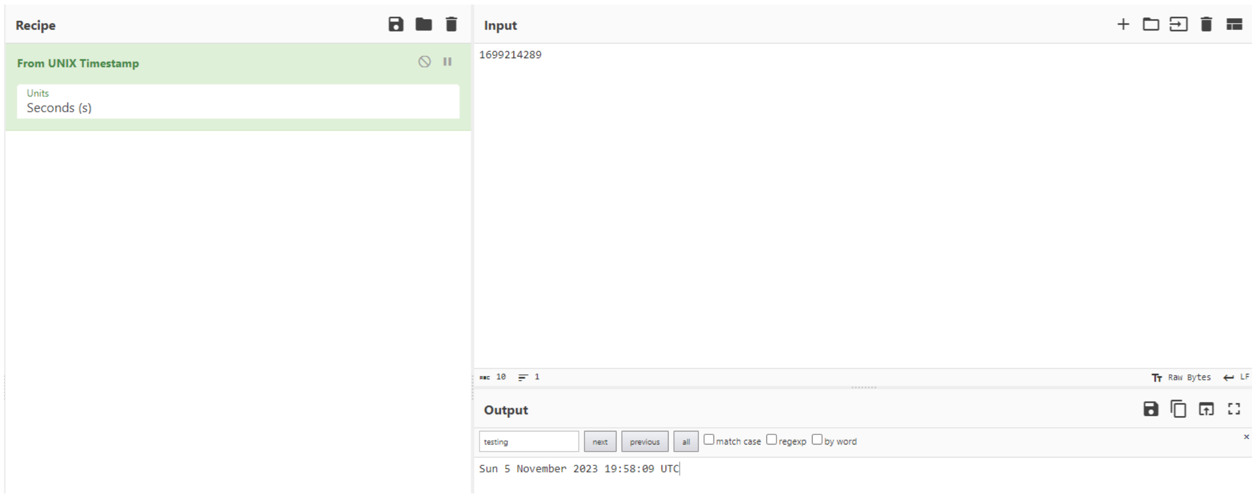
From Hex
Delimiter
Auto

```
2274657374696r67223n74657374696r673n31363939323134323839
```

56 1

```
"testi␆g"␃testi␆g␃1699214289
```

Figure 52 - CyberChef Result



From UNIX Timestamp
Units
Seconds (s)

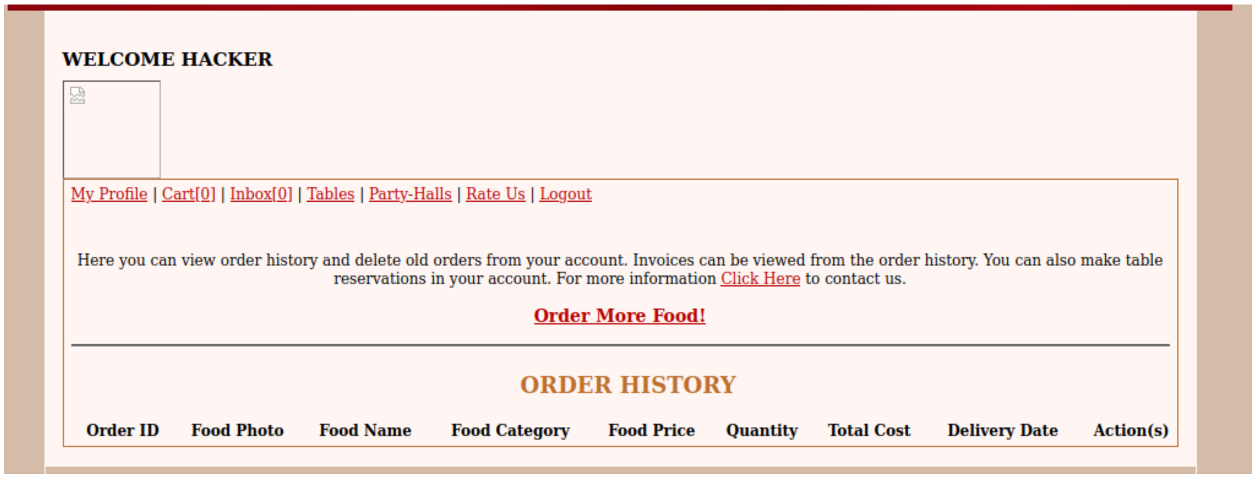
```
1699214289
```

10 1

testing

```
Sun 5 November 2023 19:58:09 UTC
```

Figure 53 - UNIX Timestamp



WELCOME HACKER

My Profile | Cart[0] | Inbox[0] | Tables | Party-Halls | Rate Us | Logout

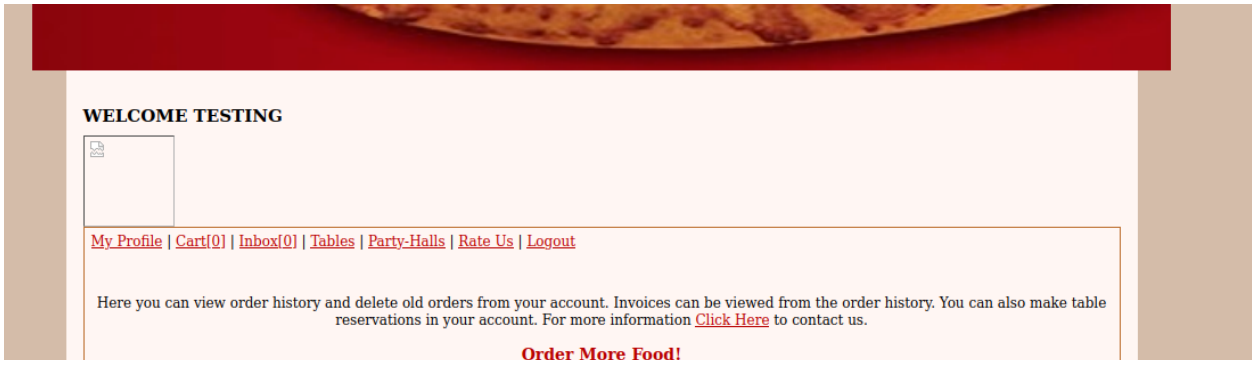
Here you can view order history and delete old orders from your account. Invoices can be viewed from the order history. You can also make table reservations in your account. For more information Click Here to contact us.

Order More Food!

ORDER HISTORY

Order ID	Food Photo	Food Name	Food Category	Food Price	Quantity	Total Cost	Delivery Date	Action(s)

Figure 54 - Hacker Account Logged In



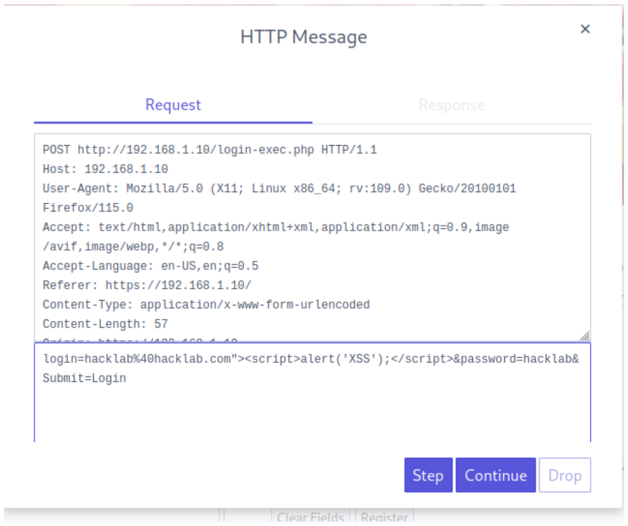
WELCOME TESTING

My Profile | Cart[0] | Inbox[0] | Tables | Party-Halls | Rate Us | Logout

Here you can view order history and delete old orders from your account. Invoices can be viewed from the order history. You can also make table reservations in your account. For more information Click Here to contact us.

Order More Food!

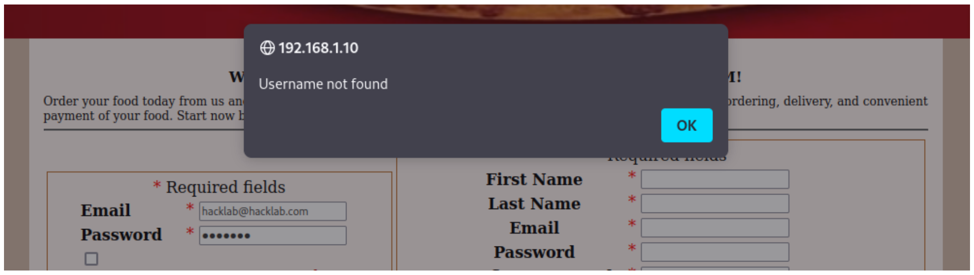
Figure 55 - Session Hijack Success



```
POST http://192.168.1.10/login-exec.php HTTP/1.1
Host: 192.168.1.10
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101
Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image
/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: https://192.168.1.10/
Content-Type: application/x-www-form-urlencoded
Content-Length: 57
```

```
login=hacklab%40hacklab.com"><script>alert('XSS');</script>&password=hacklab&
Submit=Login
```

Figure 56 - Reflected XSS Payload

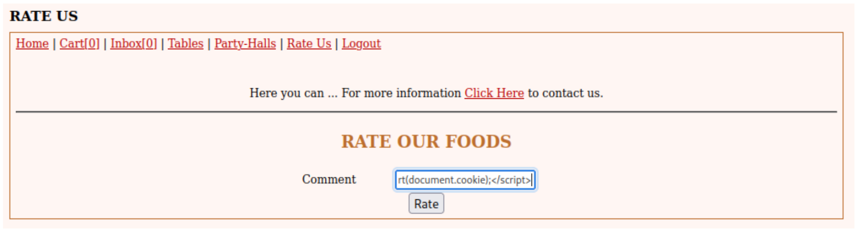


192.168.1.10

Username not found

OK
Order your food today from us and ... payment of your food. Start now b...
...ordering, delivery, and convenient
* Required fields
Email
hacklab@hacklab.com
Password
First Name
Last Name
Email
Password

Figure 57 - XSS Username not found



RATE US

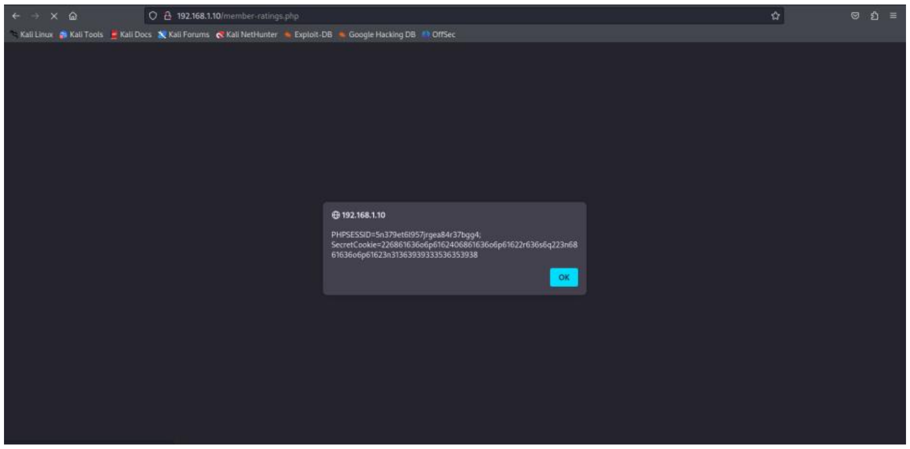
Home | Cart[0] | Inbox[0] | Tables | Party-Halls | Rate Us | Logout

Here you can ... For more information Click Here to contact us.

RATE OUR FOODS

Comment rt(document.cookie);</script>

Figure 58 - Rate Us XSS Payload

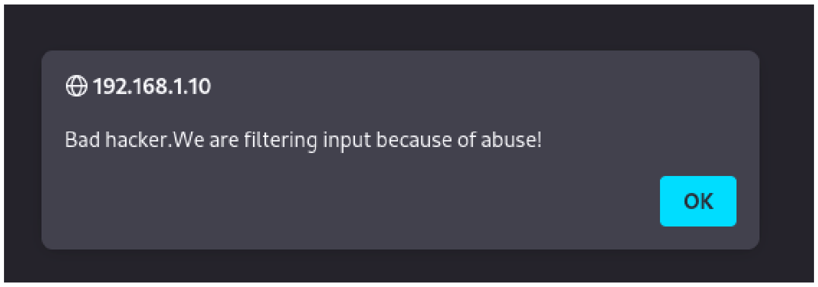


192.168.1.10/member-ratings.php

192.168.1.10

```
PHPSESSID=5n379et6l957jrgea84r37bgg4;
SecretCookie=226861636o6p6162406861636o6p61622r636s6q223n68
61636o6p61623n3136393933353636353938
```

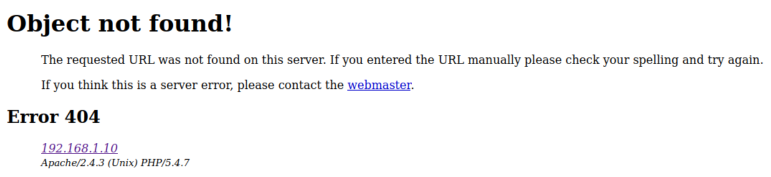
Figure 59 - Reflected XSS Success



192.168.1.10

Bad hacker.We are filtering input because of abuse!

Figure 60 - SQL Injection Bad Hacker Response


Object not found!

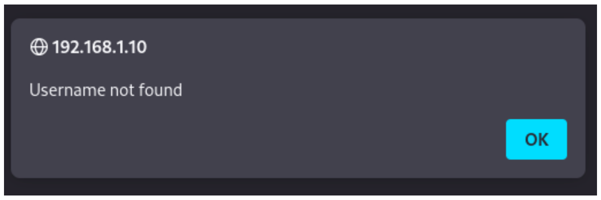
The requested URL was not found on this server. If you entered the URL manually please check your spelling and try again.

If you think this is a server error, please contact the webmaster.

Error 404

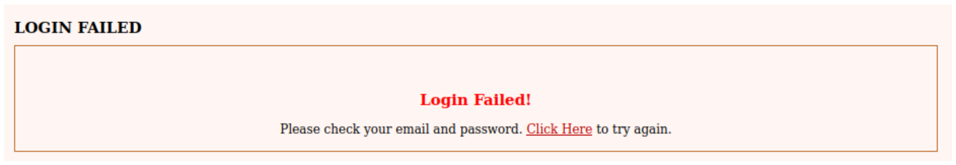
192.168.1.10
Apache/2.4.3 (Unix) PHP/5.4.7

Figure 61 - Object Not Found Error


192.168.1.10

Username not found

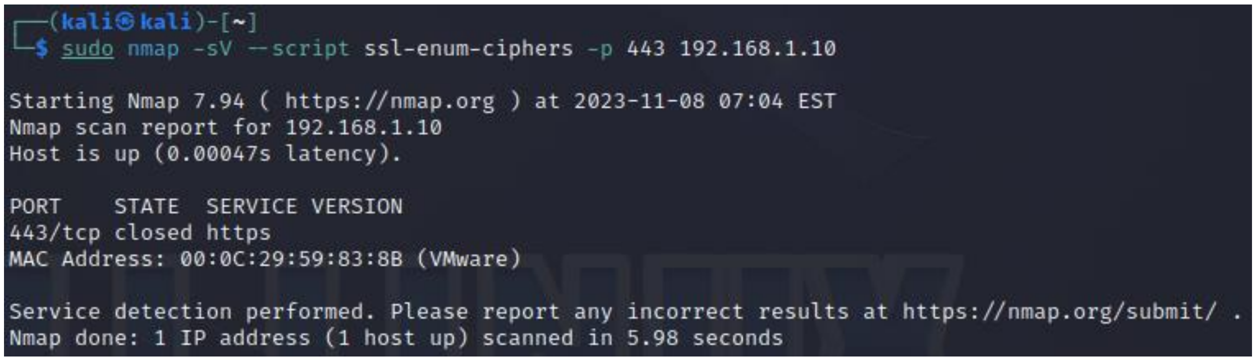
Figure 62 - Incorrect Email Error


LOGIN FAILED

Login Failed!

Please check your email and password. Click Here to try again.

Figure 63 - Incorrect Password Error

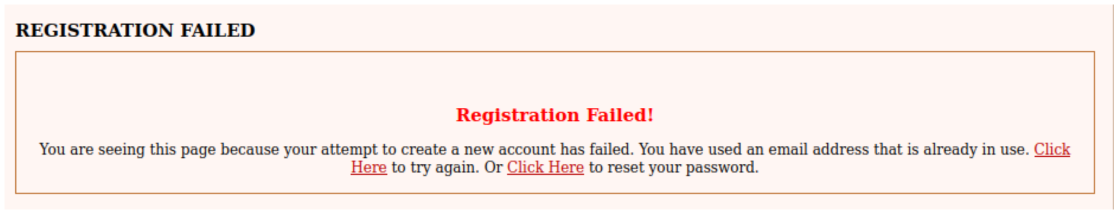

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sV --script ssl-enum-ciphers -p 443 192.168.1.10

Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-08 07:04 EST
Nmap scan report for 192.168.1.10
Host is up (0.00047s latency).

PORT    STATE  SERVICE VERSION
443/tcp closed https
MAC Address: 00:0C:29:59:83:8B (VMware)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 5.98 seconds
```

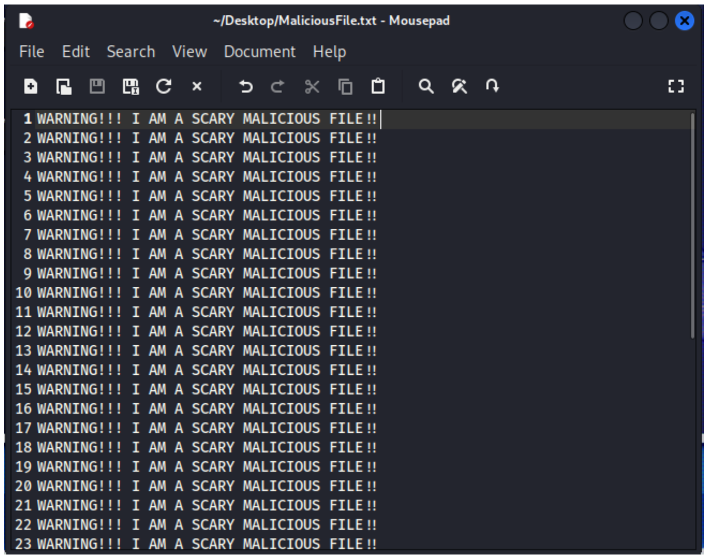
Figure 64 - Nmap ssl-enum-ciphers


REGISTRATION FAILED

Registration Failed!

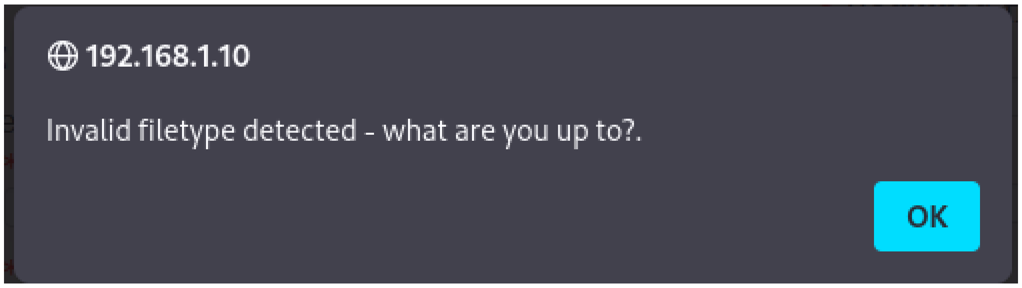
You are seeing this page because your attempt to create a new account has failed. You have used an email address that is already in use. Click Here to try again. Or Click Here to reset your password.

Figure 65 - Registration Failed


~/Desktop/MaliciousFile.txt - Mousepad
File Edit Search View Document Help

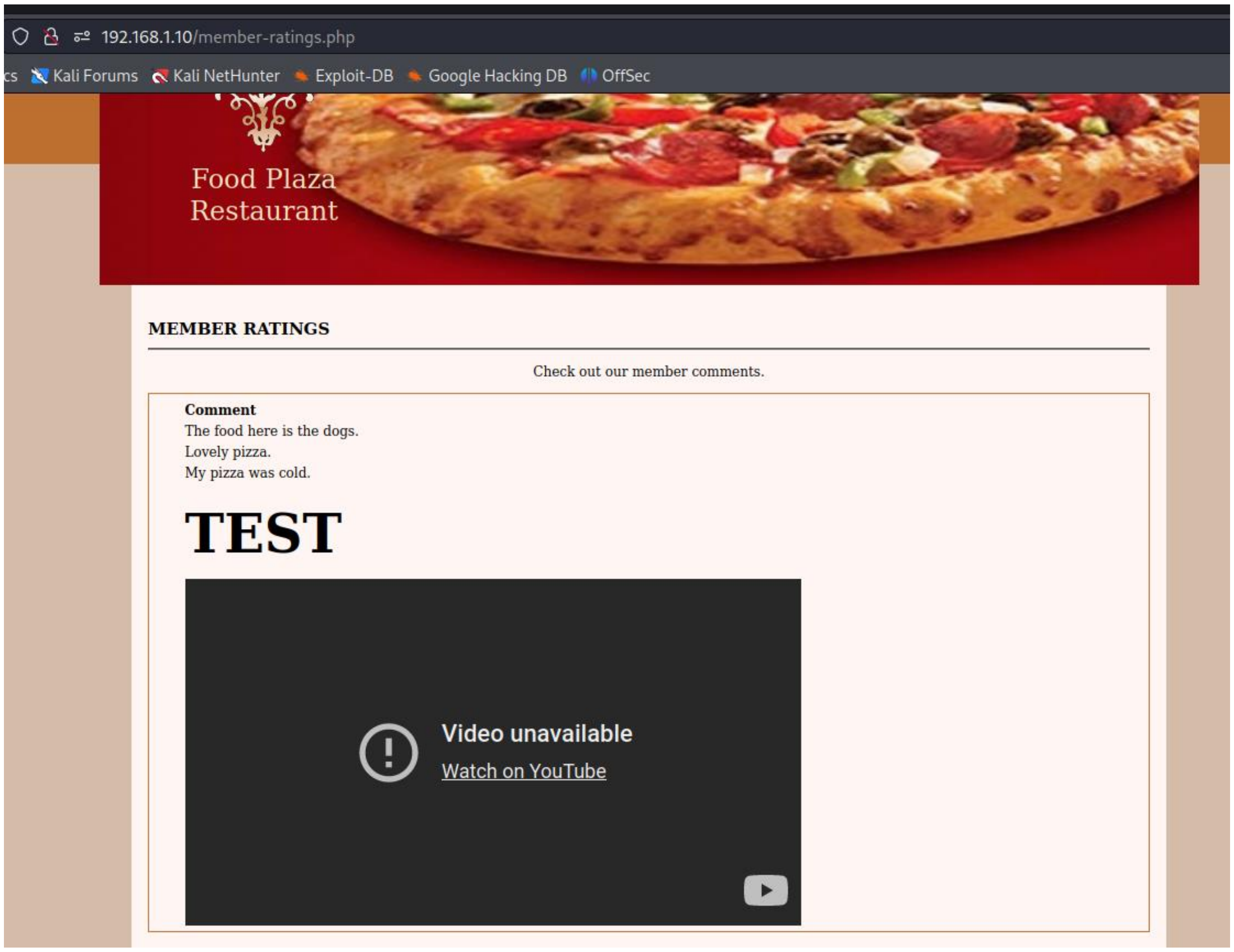
```
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
WARNING!!! I AM A SCARY MALICIOUS FILE!!
```

Figure 66 - MaliciousFile.txt


192.168.1.10

Invalid filetype detected - what are you up to?.

Figure 67 - Invalid Filetype



MEMBER RATINGS

Check out our member comments.

Comment
The food here is the dogs.
Lovely pizza.
My pizza was cold.

TEST

Figure 68 - HTML Injection