Web Application Testing

A Report Detailing the results of a web application pen test.

Jack Sime

CMP319: Ethical Hacking 2

BSc Ethical Hacking Year 3

2021/22

CMP319 – Coursework 1: You should include Introduction, Procedure and Results, References Part 1 and Appendices part 1.

CMP319 – Coursework 2: You should include Abstract, Discussion, References Part 2 and Appendixes part 2

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

.

Abstract

This report details the process of conducting a web application test for Hacklab Security Solutions and the overall results that the testing returned. The web application was tested thoroughly and aimed to find any vulnerabilities present within the site ranging from low level up to critical level issues. Resolutions and preventative actions relating to the vulnerabilities found were detailed to allow for implementation.

The methodology used to plan and undertake the test was taken from the Web Application Hackers Handbook and covered the complete website and all of its functionality. Source code analysis was conducted after the physical testing had been conducted to both confirm vulnerabilities spotted earlier and to uncover any potential issues that may have been missed in the physical testing. Following the methodology, the website was mapped and footprinted to provide information useable within testing each of the targeted sections that followed. After the physical test and the source code analysis, countermeasures were suggested along with their benefits, and an ending summarization of the current state of the website was provided.

A large number of vulnerabilities were found within the web application, including critical vulnerabilities that utilized mistakes within the website construction to obtain information that shouldn't have been obtainable. Information leaked from the website applied to both the employees of Hacklab Security Solutions and the customers that they served and included sensitive information such as addresses and contact details. Other vulnerabilities included employee-only directories being accessible to any visitor to the site and issues within the logic of the shopping aspect of the site allowing customers to manually set prices of items including negative numbers allowing for them to be returned money from the company after their purchase.

The Website, with all of the vulnerabilities, is insecure and isn't safe to be currently in use by Hacklab Security Solutions. The issues found could pose serious risks to customer information and would pose a large risk to the company.

.

Contents

.

.

.

1 INTRODUCTION

1.1 BACKGROUND

Many businesses today are looking to further increase their customer reach, and many are opting to utilize the internet. Whether it is just offering websites to view products and get in touch or allowing products to be directly bought from the site, many businesses in today's climate are choosing to go online with some part of their business. Due to the current pandemic, more shopping is being done online and so more cyber-attacks are taking place impacting more people. Interpol found that in a 4-month period there was 48,000 malicious URL discovered by one of their private sector partners (INTERPOL, 2021). With more traffic being directed to shopping websites an increase of up to 7% was seen for all sales that were online in the UK from 2019 to 2020 (Global e-commerce jumps to $26.7 trillion, COVID-19 boosts online sales | UNCTAD, 2021). With this comes negative exposure as hackers and other cybercriminals attempt to steal information and de-fraud the businesses running the websites. An Interpol report looks at the distribution of cybercrime during the pandemic and found that 22% was related to the usage of malicious domains whereas 59% was related to general fraud and scamming (Figure 1).

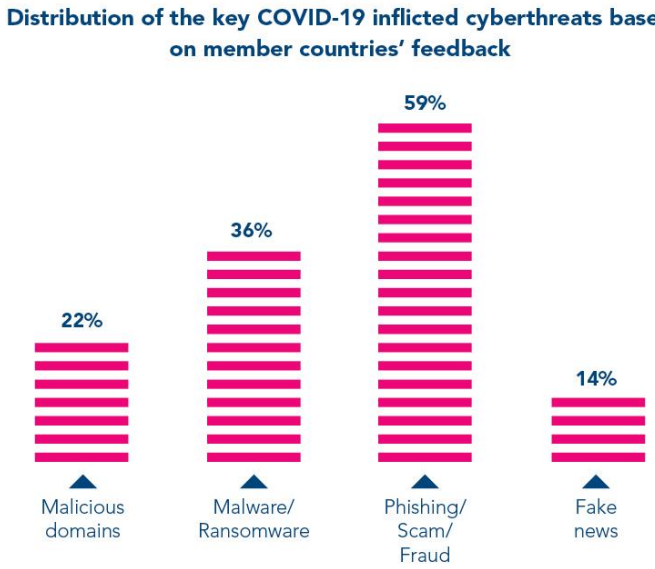


Figure 1: Interpol Report Findings

With websites being easily accessible to anyone if they aren't secured correctly then there can be major consequences. Depending on the website and its initial function, the website could contain lots of sensitive details on all its customers from something as simple as their name to potential bank details and addresses. All this information must be secure to ensure that customers trust the business they are shopping at and so that the business in question doesn't receive fines for any potential data breaches that may occur. During the pandemic, the cost of a data breach rose to around £16,000 but ranges can

be as low as a few hundred pounds to upwards of several hundred thousand pounds (Patterson, 2021). Many attacks come from several different avenues as seen above but with a business's website being easily accessible it will be one of the first points of contact for obtaining information and so should be secure to limit the potential of the attacks.

The owner of Hacklab Security Solutions has requested that a penetration test be carried out on their website to ensure that everything is secure and that there are no faults within the site that may allow information to be leaked or allow hackers access to sensitive data. A report would be produced after the test detailing the procedure of the test along with results and findings. Recommendations on how to fix any areas identified would also be provided inside the report.

1.2 AIMS

This report aims to identify, test, and evaluate the security the security of the Hacklab Security Solutions website. This overall aim comprises of a number of sub aims:

- Identify security vulnerabilities within the website
- Attempt to exploit the vulnerabilities found
- Present results of exploitation
- Discuss the implications that the found vulnerabilities could have on the website
- Provide suitable suggestions to fix the vulnerabilities found

2 PROCEDURE AND RESULTS

2.1 OVERVIEW OF PROCEDURE

For the Testing process, the chosen methodology to use was from the Web Application Hackers Handbook (Stuttard and Pinto, 2011). This methodology was used by the tester as simulates the real-life steps an attacker may take against a web application and covers the majority of the main ways a hacker may attempt to infiltrate such as application. The chosen methodology provided the tester with a good level of confidence that all of the potential areas of attack had been investigated with the resources that were available to the tester.

The methodology used contained 13 sections each detailing a specific avenue of attack that could be taken, these headings are listed below:

1. Map the Applications Content
2. Analyse the Application
3. Test Client-side Controls
4. Test the Authentication Mechanism
5. Test the Session Management Mechanism
6. Test Access Controls
7. Test for Input-based Vulnerabilities
8. Test for Function-specific Input Vulnerabilities
9. Test for Logic Flaws
10. Test for Shared Hosting Vulnerabilities
11. Test for Application Server Applications
12. Miscellaneous Checks
13. Follow Up any Information Leakage

Although the methodology has 13 sections to test, some of the listed sections weren't explored by the tester as they didn't apply to the scope of the web application in question. Sections 8,10 and 11 (Function-specific Input Vulnerabilities, Shared Hosting Vulnerabilities, and Application Server Applications) were the sections that the tester had excluded, section 12 (Miscellaneous Checks) was also omitted from the report as the tester found nothing that related to the section during the testing. The end report followed sections 1-7,9,13.

The tester made use of several different tools throughout the testing with a wide range of functionality. A virtual machine with Kali Linux installed was utilized along with many of the tools that come installed on the package (Kali Linux Tools, 2021). A select few applications and websites were used in conjunction with the kali tools for the proportion of the testing. These were the web browser OWASP Mantra and the websites CyberChef (CyberChef (gchq.github.io)) and Crackstation (CrackStation - Online Password Hash Cracking - MD5, SHA1, Linux, Rainbow Tables, etc.).

The website CyberChef, produced by GCHQ, is used for a wide range of encoding, decoding, arithmetic, and logic functions as well as many other operations on the input given to it (Figure 2).

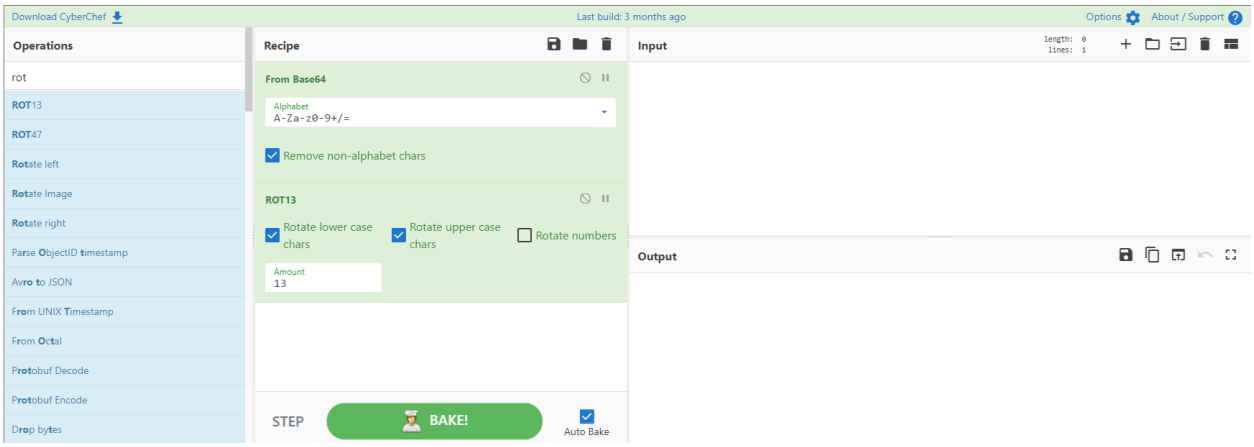


Figure 2: CyberChef Interface

The website Crackstation is used to crack many different password hashes including MD5 and SHA1 and uses lookup tables with over 15 billion entries (Figure 3).

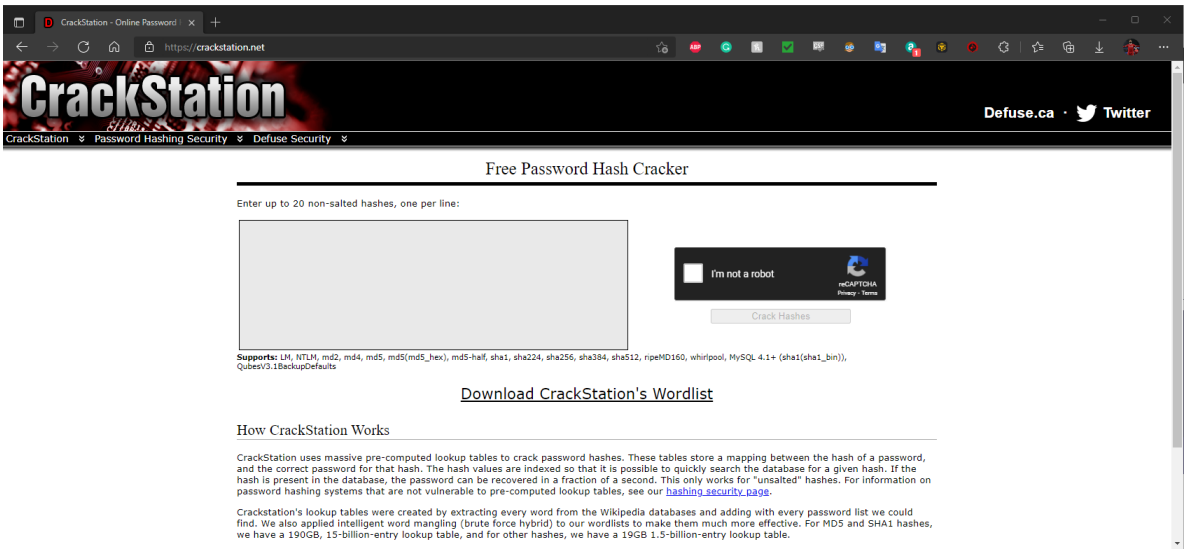


Figure 3: Crackstation Interface

2.2 MAP THE APPLICATION'S CONTENT

2.2.1 EXPLORE VISIBLE CONTENT

To start with, OWASP Zap was set up and Mantra was configured to run through the Zap proxy to allow the target website to be investigated and mapped as it was initially explored like a regular user of the website would (Figure 4). All the webpages available by using the website normally were visited along with the usually available functions of the website such as logging in to an account with the details given, browsing through the products available to purchase, and registering another user account. After logging in using the given account, features and pages that were only available while logged in were also visited such as adding items to the cart, changing passwords of the account, and visiting the profile page. A site map was generated by Zap as the website was navigated along with showing any requests and responses from any of the webpages as they pass through the proxy.

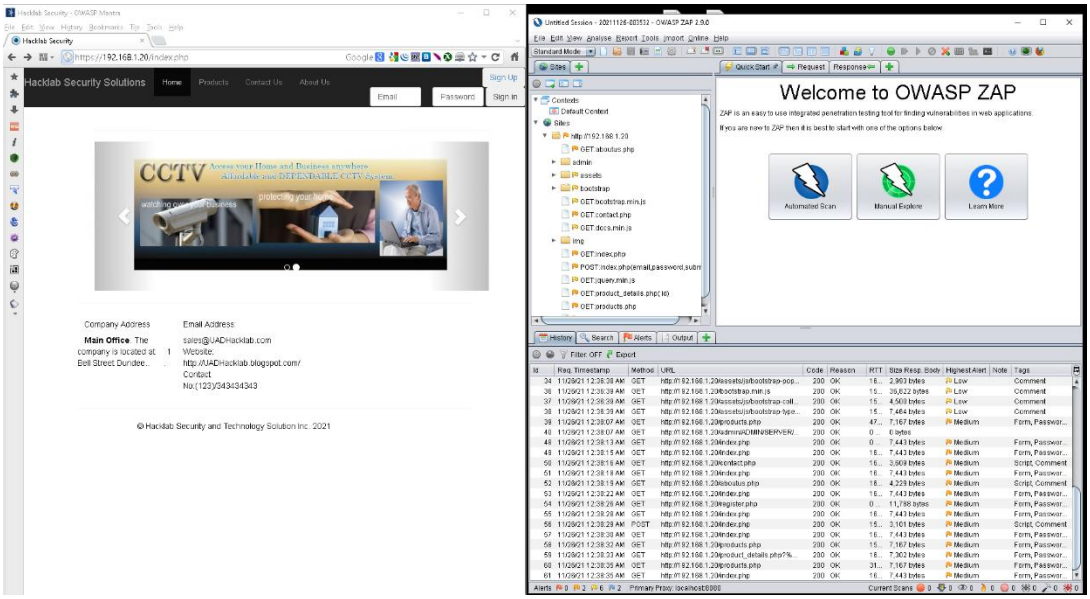


Figure 4:ZAP and Mantra Configured

Both the login page and the registration page were accessed during the initial investigation and both pages looked to function properly when used. The use of Java and regex was discovered on the registration page to check the entered password and impose a limit of between 7-14 characters for the field and to ensure that it didn't contain any symbols or special characters. It was also used to validate the age with the date of birth that was supplied to ensure the registering user was over the age of 16, this doesn't match the alert that is shown when trying to register while under 16 as the alert states the user must be over 18 (Figure 5).

```
<script type='text/javascript'>
function validation(){
    //var CheckPassword = /^[A-Za-z]\w{7,14}$/; - numbers and characters and uppercase
    //var CheckPassword = /^[a-z]\w{7,14}$/; -
        var letterexp = /^[a-zA-Z]+$/;
    var quanti = 32;
    var CheckPassword = /^\w{7,14}$/;
    if(document.getElementById('password').value.match(CheckPassword)){
    }else{
        alert('Password must have minimum and maximum of 7 to 14 characters');
        document.getElementById('password').value = '';
        document.getElementById('password').focus();
    }

  var date1 = new Date();
        var  dob= document.getElementById("dob").value;
        var date2=new Date(dob);
            var y1 = date1.getFullYear(); //getting current year
            var y2 = date2.getFullYear(); //getting dob year
            var ages = y1 - y2;           //calculating age
        if(+ages<=16){
            alert("Age below 18 is not allowed to register");
             document.getElementById('dob').value='';
        }
}
</script>
```

Figure 5: Java Script for password and age

Two other regex lines are present in the source of the page and have the functionality of checking the password input for length and content but both have been commented out and so they aren't executed. The email section of the form did not contain the same monitoring JavaScript and allowed an invalid email address to be entered that did not contain the "@" symbol. It was seen that PHP error reporting was enabled as after an unsuccessful login, using a correct username but an incorrect password, PHP errors were seen on the webpage in plain view, this gave away information on directories being used and files being accessed during the process of logging in (Figure 6).

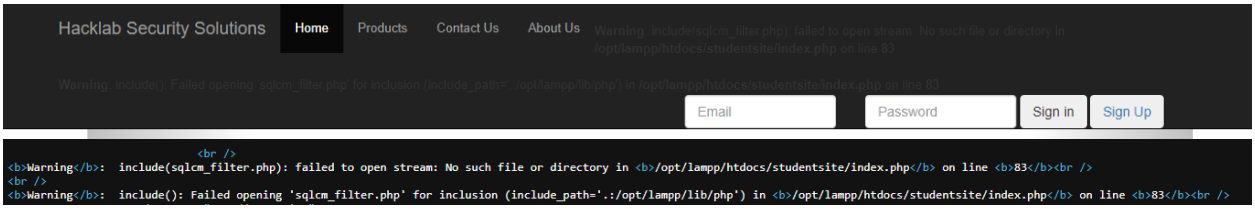


Figure 6: PHP Errors on Screen and in Source Code

An automated Zap spider was then performed to uncover any webpages that may have been missed during the manual scan and then an automated scan was performed to test all the discovered pages for vulnerabilities that may be present. The scan was set up to ensure that it stayed within the scope of the investigation and to ensure that no web pages that could have disrupted the session were attacked. A list of the URLs found can be seen in Appendix A.

2.2.2 DISCOVER HIDDEN CONTENT

Following on from the scans, the website was then tested with a webpage that it would be unable to display, the address "192.168.1.20/errortest". This resulted in the website displaying a 404-error message which shows the website wasn't able to find the requested webpage (Figure 7). The error

page also displayed information detailing the operating system that was being used and the software that was in use.

Figure 7: 404 Error page containing OS information

Robots.txt is used to tell search engine crawlers which URL's web crawlers can access on your site (Robots.txt Introduction and Guide | Google Search Central, 2021). After examining the robots.txt for 192.168.1.20, there was only an info.php file listed under the disallow section (Figure 8). Due to this file being excluded from web crawlers view, it implied that there was something to hide within the file and that it should be investigated further to discover its contents.

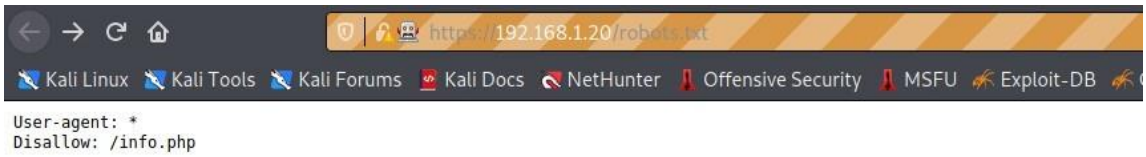


Figure 8: Robots.txt

Info.php was then accessed and within it was a large amount of sensitive information related to the target website. Within the file there was information on the operating system version and the loaded modules along with other sensitive info (Figure 9).

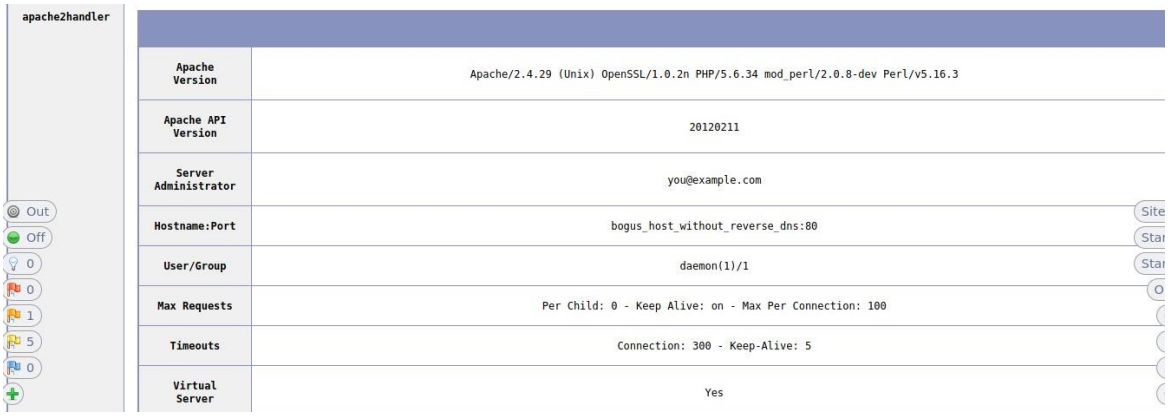


Figure 9: Info.php

The applications Dirb and Dirbuster were both used to uncover any hidden directories that may not have been discovered yet. Dirbuster was used first and the wordlists “directory-list2.3-small.txt” and “directory-list2.3-medium.txt” included with the software were used to brute force any directories that match the lists used (Figure 10). The small.txt returned the best results and displayed the most information out of the two and returned a large number of directories and files along with their response codes, the full list can be seen in Appendix B.

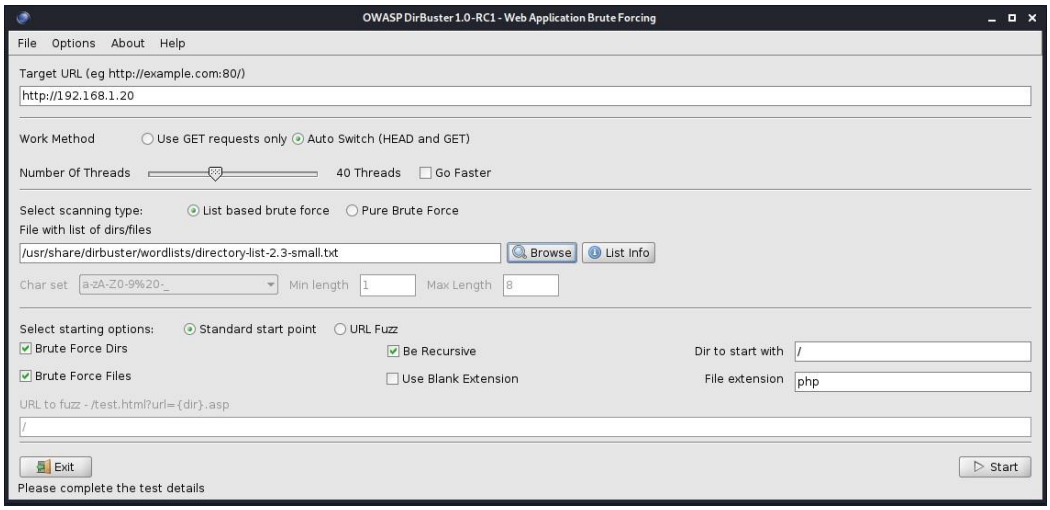


Figure 10: Dirbuster UI

Dirb was then used, unlike DirBuster, it is run from a terminal instead of a GUI (Figure 11). Two different wordlists were used again that were included with the application, common.txt and big.txt were both used against the target site and common.txt returned the most useful results. The output files containing all the directories found from common.txt will be available in Appendix C.

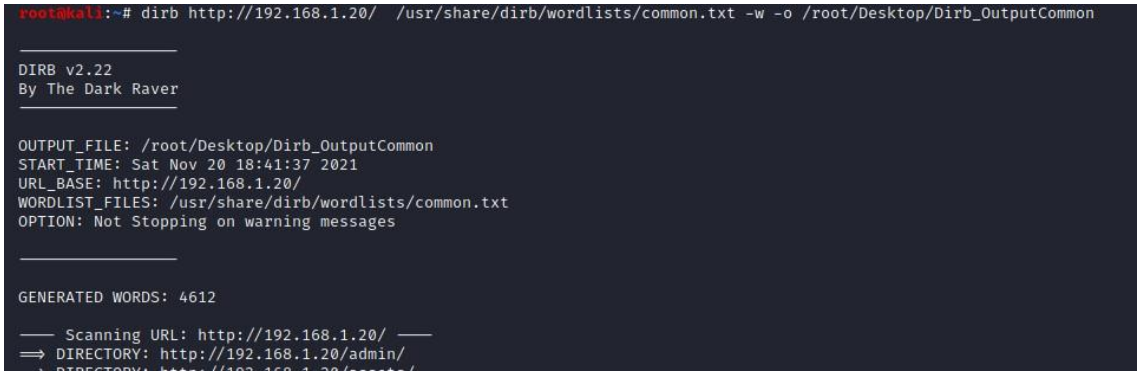


Figure 11: Dirb Command used

From the use of DirBuster and Dirb revealed important directories that were hidden including the admin login page and the subdirectories that lead on from the admin login page. The admin login page was able to be bypassed by browsing to one of the sub directories such as "/admin/ADMIN" where the admin pages were listed and able to be visited without proper authentication (Figure 12).

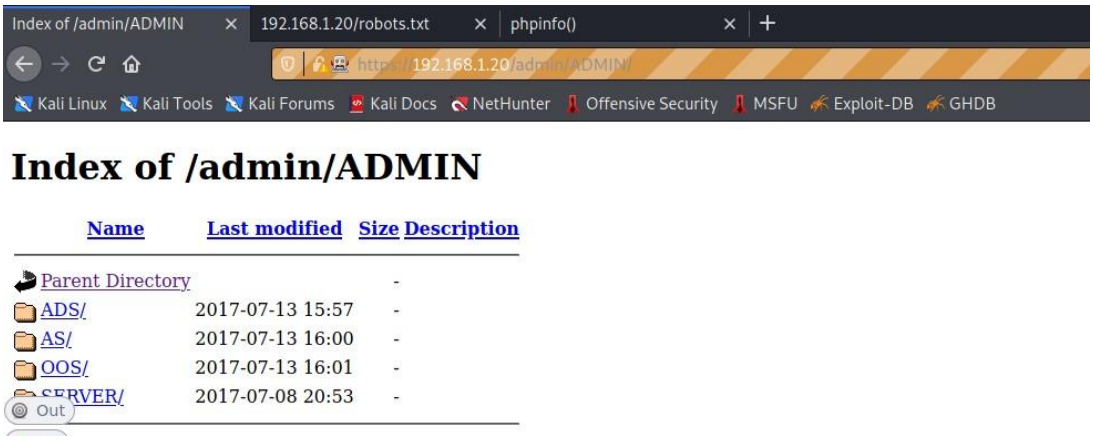


Figure 12: /admin/ADMIN directory

These pages contained sensitive data about customer account (Figure 13), products available on the site and allowed access to the announcement functionality that displays a message to users once they login. Although access was able to be achieved and details were able to be viewed without authentication, nothing was able to be edited or deleted without being redirected to the login page for proper authentication.

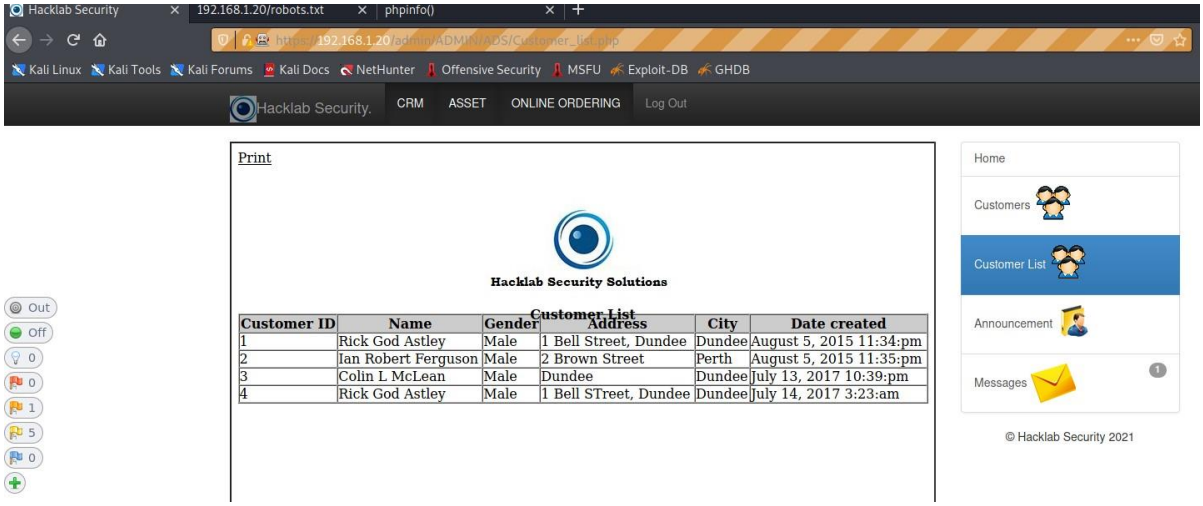


Figure 13: Customer Data

After authentication using a proper account, items were able to be added to cart and the checkout page was accessible. After checking out, the receipt for the order could be accessed from "/user_order.php". The receipt itself detailed how was due, what was ordered and the personal details of the customer who placed the order. The URL takes a variable named "id" and uses it to show the correct receipt. Changing this variable allowed for other customers receipts to be shown instead of only the correct users receipts (Figure 14).

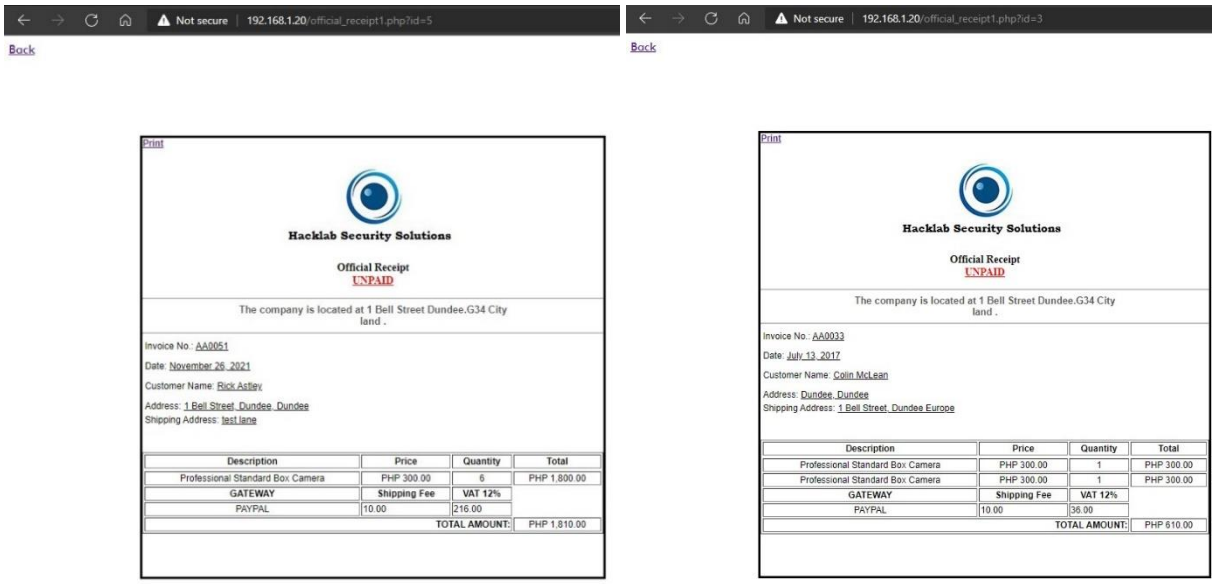


Figure 14: Receipt showing correct authenticated user and receipt that shouldn't be accessible

Dirb and Dirbuster (Dirb, 2021) also uncovered a database directory that contained a single .sql file, the file was then downloaded and looked at using Visual Studio Code and the contents were discovered (Figure 15). The file contained a complete dump of the database being used and revealed the layout for all the tables used in the website along with the details that were stored within including usernames, addresses and passwords that were hashed in MD5 without a salt. Details on the actual versions of phpMyAdmin being used were also in the file along with the server version and the version of php being used. The full contents of the database will be available in Appendix D.

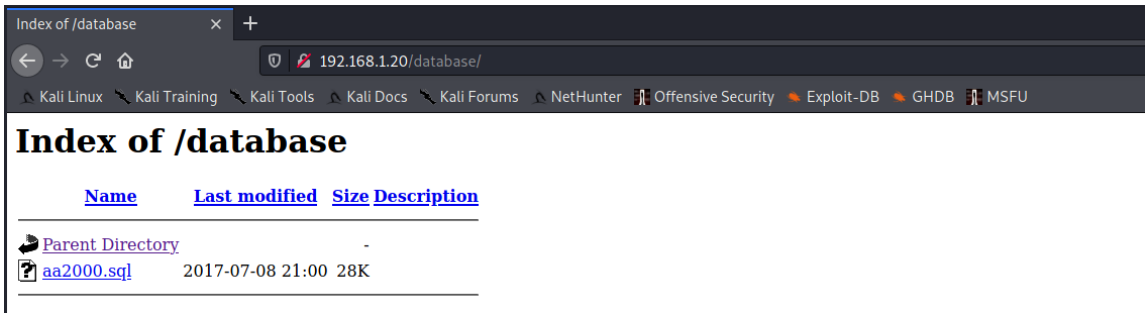


Figure 15: SQL database dump file

2.2.3 SUMMARY OF APPLICATION MAPPING

From the initial mapping phase, a large amount of useful information was found out about the website. The overall purpose of the website was quickly understood and the usage of both JavaScript and regular expressions were noted on the login and registration pages. Error messages displayed when accessing a webpage that didn't exist provided valued information along with the contents of the Robots.txt file which pointed towards an "info.php" page that revealed details on the operating system and modules loaded on the server. Through the usage of Dir and Dirbuster, multiple directories were found on the site and opened further avenues of investigation.

2.3 ANALYSE THE APPLICATION

2.3.1 IDENTIFY FUNCTIONALITY

The website was created to allow users to purchase security cameras from Hacklab Security Solutions. Visitors to the website can browse the selection of products and view the product details including price and a short description. New customers are asked for their gender, full name, email, password, date of birth along with and address and a contact number when they are registering for an account.

After accessing a valid account, a user could add and remove these products to their cart, change any personal details provided during registration, add or change the profile picture of their account, sent messages to an admin of the website and also access a list of previous orders placed on the site. When trying to access a webpage that requires authentication such as "/user_index.php" without being signed in, the webpage redirects you to the landing page "/index.php" for you to sign in. To access the admin page, different account details were required as regular account details did not work with the login. Admin details cracked using hydra were used to access the admin sections of the website. Once signed into the admin account, the admin section was split into 4 subsections (CRM = Customer Relations, ASSET = Asset management, ONLINE ORDERING = ordering management and CONFIGURATION = Database) each with its own functions (Figure 16).

Figure 16: Admin Section subsections

The customer management subsection allowed for the customer database to be searched for a certain customer, customer login accounts could be deleted or archived, a customer list could be printed that contained personal details, announcements could be added, deleted, or edited and messages from customers could be viewed and replied to.

The asset management section was used to search for products that were stocked, add, remove and delete products from the database and a product list could be printed off showing the price,

quantity, and when the product was added to the system. Assets belonging to the company could also be viewed and the categories could also be edited or deleted.

The subsection related to online ordering was used to observe pending orders that had been placed by customers and allowed for the orders to be either viewed, deleted, or confirmed. The page also had notifications to show when a new order had been placed and needed to be confirmed.

The final subsection was related to the database used on the site. From the page, admin logins could be deleted and edited along with the user type for each admin. An administrator list could also be printed that contained the usernames and user types for admin accounts and the database could also be backed up from this section. The recent archive activity for both customers and products could be accessed and logs from customers and admins logging in and out were available to view along with work changes such as any announcements that have been added.

2.3.2 IDENTIFY DATA ENTRY POINTS

The table below details the multiple entry points on the website that allow for user inputs to be processed by the website. Some entry points were protected by authentication such as user and admin login details.

Data Entry	Location on site
Email, Password	/index.php
F-name, M-name, L-name, email, password, address, contact number	/register.php
What, Image, Date, Where, Details	/admin/ADMIN/ADS/add_new_announcement.php
New Password, Image	/updatepassword.php
Name, email, subject, message	/Email.php
Message	/admin/ADMIN/ADS/messages.php
Name, price, quantity, description, image	/admin/ADMIN/AS/add_new_products.php
User type	/admin/ADMIN/SERVER/add_new_user_type.php
Employee Name, username, password	/admin/ADMIN/SERVER/add_new_user.php

2.3.3 IDENTIFY THE TECHNOLOGY USED

A range of different technologies was used for both the client-side and server-side of the website. Client-side technologies were identified by interacting with the website through normal usage and by examining the various page's source code available.

On the client-side, the use of java was found in relation to the image carousel from bootstrap that was displayed on the main page “index.php” and was found on the registration page when limiting the input on the password field and limiting age range.it was also used to force the first letter of all three names and the address of anyone who was registering.

An Nmap scan was used to identify was technologies were being used server-side (Chapter 15. Nmap Reference Guide | Nmap Network Scanning, n.d.). The scan revealed the open ports on the server and the services that were running on the ports related to them (Figure 17). Port 21 was open on the server and it relates to the FTP service, this was in use to more than likely allow the website owner to add or remove files such as new or updated webpages or any background files to the server.

Both ports 80 and 443 were open and they relate to HTTP and HTTPS respectively. The HTTP port was used to access the website that was to be tested. The HTTPS port was open and used to access the starting directory that was used to create the website but was ignored for the duration of the test due to being outside the scope of this investigation.

Port 3306 was also identified from the scans as being open and relates to the MySQL service which meant that an SQL database was in use on the server.

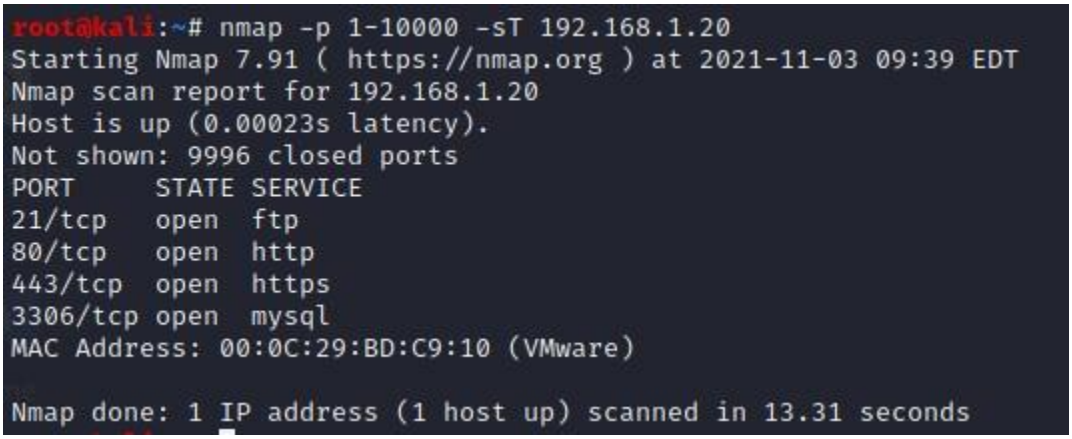


Figure 17: NMAP Scan of Open Ports

From the above mapping of the application section, it is revealed that the server was running a version of Apache webserver. The web server scanner Nikto was used next to scan the server for any other vulnerabilities that it may have. Nikto checks for a wide range of issues such as dangerous files, outdated versions, and configuration items such as having multiple index files (Sullo and Lodge, 2021). The results from the Nikto scan can be seen in Appendix E.

2.3.4 MAP THE ATTACK SURFACE

From the investigations into the website and the server so far, it was possible to roughly map and plan areas that may be vulnerable within the website. From using the website, it was noted that the website interacts with a SQL database on multiple different pages, this may be vulnerable to SQL injection. The customer mail section and the announcement section may be vulnerable to an XSS attack (Cross site scripting) and would affect a lot of users. Login and registration forms may allow for the brute force of details if no precautions or preventative measures have been implemented. The transfer of information from these pages may also be vulnerable if not implemented in the correct way.

2.3.5 SUMMARY OF APPLICATION ANALYSIS

From the analysis of the website the overall purpose of the website was clear in that it was designed to sell security cameras. Using knowledge obtained in the previous mapping stage, data entry points to the website were noted and the types of data entered at these points was also noted down. An Nmap scan revealed the open ports and services on the Server and a plan of the attack surface was mapped to keep track of the most susceptible areas of the website that could be targeted later in the test.

2.4 TEST CLIENT SIDE CONTROLS

2.4.1 TEST TRANSMISSION OF DATA VIA THE CLIENT

It was found that the product item pages used a variable in the URL to decide what product would be shown to the user. Numbers 1-11 returned valid products on screen that could be purchased whereas exceeding this range returned a store page about a product that was out of stock and was also listed as having its price set to zero. This was the same result for any number that was submitted in the URL over 11. Entering the string “blank” in the variable continued to return the same store page that was returned when the number exceeded 11 (Figure 18).

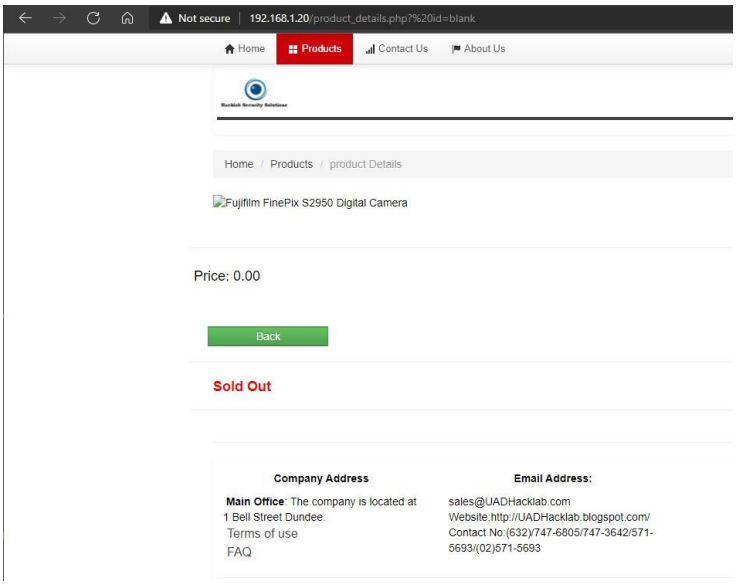


Figure 18: "blank" URL Variable

As seen earlier in section 2.2.2 customer receipts could also be accessed using the same process. After submitting a number variable that did not correspond with a receipt, a receipt would still be shown but would have no charges or products listed on it apart from the shipping fee. Like the product page above, the variable “blank” was submitted again and returned the same results that

were seen when providing a number that didn't correspond to a receipt. An error was shown in the product section of receipts that had no products listed and was in relation to the tax calculated for the transaction. The error was suspected to appear due to there being no cost listed that tax could be calculated on, hence the trying to tax an amount of zero caused the error.

2.4.2 TEST CLIENT-SIDE CONTROLS OVER USER INPUT

The limitations on the registration page that were controlled by JavaScript were tested in an attempt to subvert them. The tool Burp Suite was used to intercept the POST request and edit the submitted fields in an attempt to bypass the limitations (How to use Burp Suite for penetration testing, 2021). A new test account was registered, and the POST request was captured to be altered. A password of "TEST123" was initially set which complied with the JavaScript (Figure 19). Using Burp Suite this was changed to "TEST1" which did not follow the rules set (Figure 20). The webpage accepted the edited variables and registered the account with the edited password and allowed the new account to be used to sign in.

Figure 19: Initial POST request

Figure 20: Edited POST request

The same scenario was repeated to test that symbols could be added to passwords to bypass the other checks. Once again, a new account was registered with the password "TEST123" and was then changed, using the intercept feature on Burp Suite, to "TEST123!". The webpage again accepted the password, and it was successfully used to log in to the newly made account.

The same scenarios as above were then repeated once again, another test account was registered, and the POST request was intercepted to be edited. This time all submitted variables were removed and left blank except from the contact number as this caused the webpage to not accept the request as it expected a number (Figure 21). The webpage once again accepted the new account

with no submitted variables. The created account couldn't be accessed by logging in, but its existence was proven by navigating to the admin area and accessing the customer's table where a blank row was present (Figure 22).

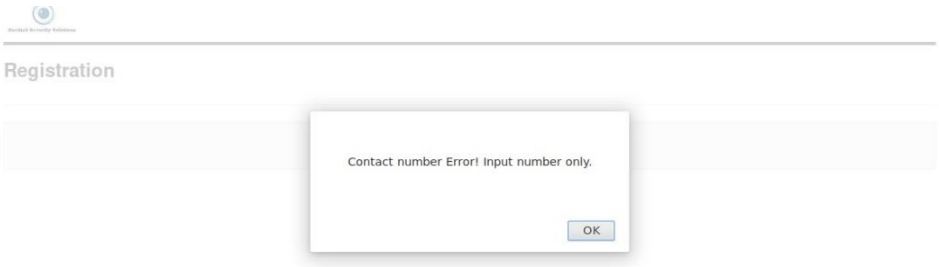


Figure 21: *Error due to Contact number*

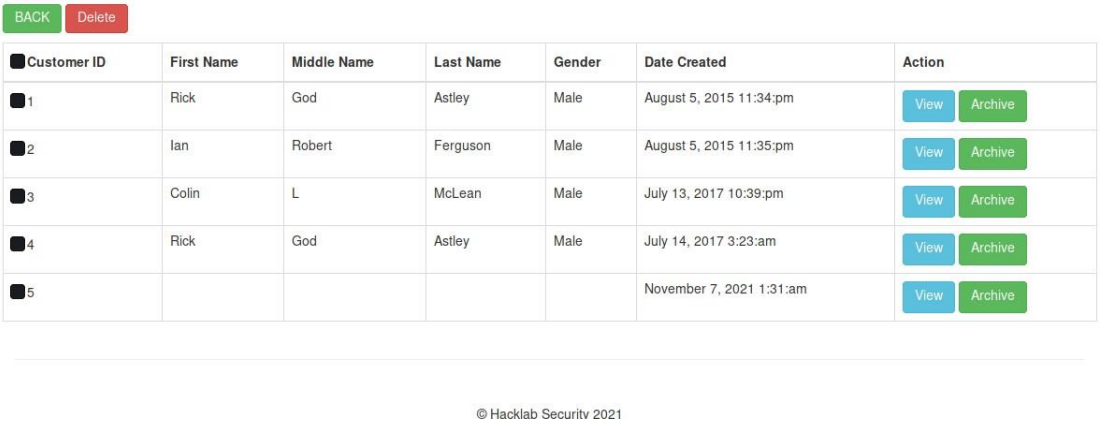


Figure 22: Blank row proving the existence of the blank account

2.4.3 SUMMARY OF CLIENT SIDE CONTROLS

After testing the client side controls it was found that the product details pages were controlled through a variable submitted in the URL, changing this variable to different numbers would display different products. After exceeding the number 11 the page returned a single page that displayed a product name with no stock or price, this was also seen when entering a string. The usage of JavaScript and regular expressions to enforce a limit on the passwords length and contents was bypassed using the intercept function on Burp Suite, which allowed a password to be submitted that broke the rules that were set.

2.5 TEST THE AUTHENTICATION MECHANISM

2.5.1 UNDERSTAND THE MECHANISM

There were several sections of the website that were identified as requiring authentication to be accessed. Authentication was required to access any account details and was required before being about to add anything to cart. Separate admin authentication was not required to access admin areas due to an exploit in the URL, further discussed in section 2.2.2, but was required to make any changes to the admin areas.

2.5.2 TEST PASSWORD QUALITY

As seen earlier, the website uses JavaScript to enforce a password length minimum and maximum of between 7 and 14 when entering the password during registration. Various different passwords were then set on test accounts to check for any quality checks. The top 10 worst passwords from the top 200 identified by Nordpass (Nordpass, 2021), that fit within the set rules for passwords, were used to check for any quality alerts that may be present on the website to alert the user to a weak password entry. The complete list of the 200 passwords can be viewed in Appendix F.

Incomplete validation of credentials was then checked to ensure the webpage was properly validating passwords submitted while logging in. The Hacklab account password was tested as a login attempt was tried using the full upper-case version of the password “HACKLAB” to test that case sensitivity was checked when validating the passwords given when logging in (Figure 23).

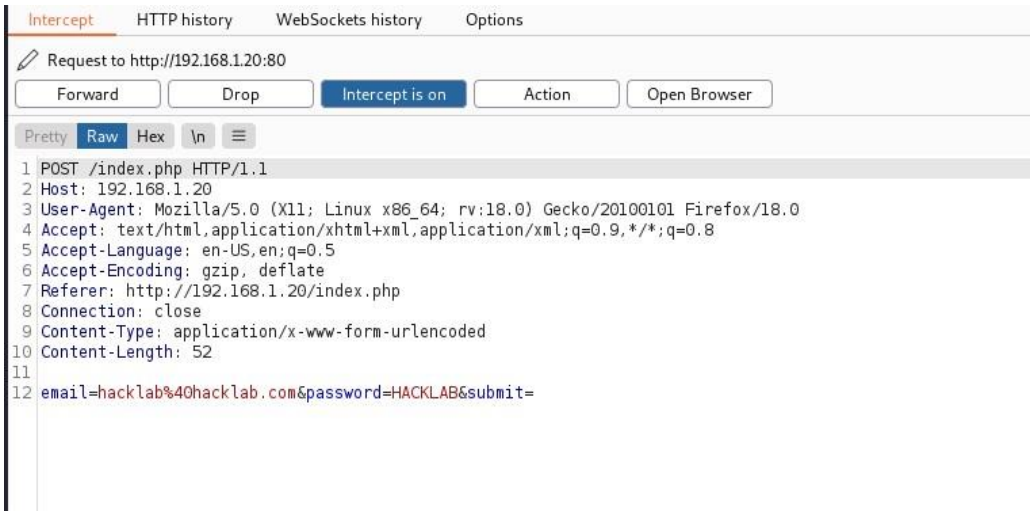


Figure 23: Uppercase password submission for Hacklab account

2.5.3 TEST RESILIENCE TO PASSWORD GUESSING

Using the provided Hacklab account, the account was attempted to be accessed using the correct username but an incorrect password. This was repeated 15 times, each with an incorrect password, to test for any potential password lockout that way have been used. After the 15 unsuccessful attempts the account was then accessed using the valid credentials and a successful login was achieved. This demonstrated that a lock out was likely not in use and no notification was displayed to the user after successful login to alert them to the attempts to access their account.

The software hydra (GitHub - vanhauser-thc/thc-hydra: hydra, 2021) was then used to try to guess any potential user logins that are currently unknown. Hydra was only used against the admin login as an unexpected error would occur if used against the normal customer login. The hydra test was hardcoded to use the username “admin” due to it being very common as a default username, and the rockyou.txt file was provided to be used as the source for passwords to be tried (Figure 24).

```
root@kali:~/Desktop# sudo hydra 192.168.1.20 -vV -l admin -P rockyou.txt http-post-form "/admin/index.php:username=admin&password=^PASS^&submit=:F=Please Check Your Username And Password"
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics a
nyway).
```

Figure 24: Hydra Command

After hydra had been running for about 30 seconds a successful login was detected and the test was stopped (Figure 25). The details that had been uncovered were then checked to ensure that they were indeed valid for the admin section and this allowed full access to any of the admin features that had previously been unable to be investigated.

```
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "gerald" - 682 of 14344398 [child 13] (0/0)
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "jordan1" - 683 of 14344398 [child 3] (0/0)
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "010203" - 684 of 14344398 [child 1] (0/0)
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "daddy1" - 685 of 14344398 [child 14] (0/0)
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "zachary" - 686 of 14344398 [child 10] (0/0)
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "daddysgirl" - 687 of 14344398 [child 0] (0/0)
[ATTEMPT] target 192.168.1.20 - login "admin" - pass "billabong" - 688 of 14344398 [child 4] (0/0)
[80][http-post-form] host: 192.168.1.20   login: admin   password: kelly
[STATUS] attack finished for 192.168.1.20 (waiting for children to complete tests)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-11-12 07:47:20
```

Figure 25: Successful Hydra attempt

2.5.4 TEST USERNAME UNIQUENESS

To test the handling of a username match when registering an account, a new account was registered with the username “TT”, a second account was then attempted to be registered using the same username but with a completely different password. This attempt was blocked by the website and an error message was displayed showing that an account with the current username already exists on the site (Figure 26).

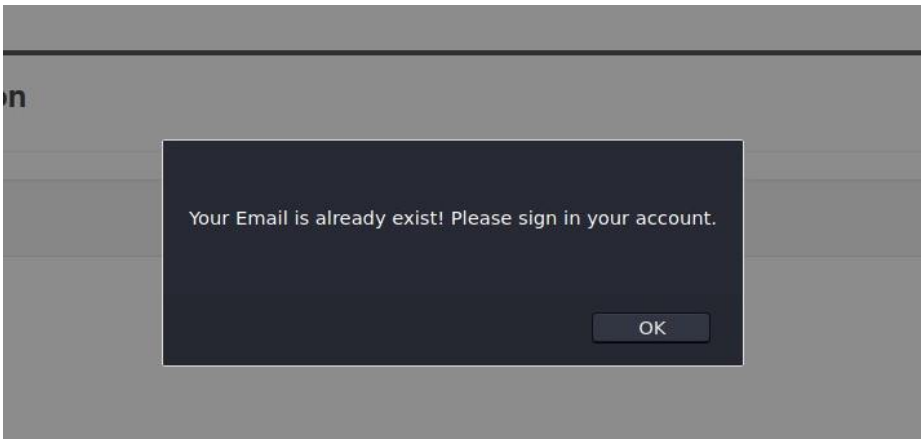


Figure 26: Registration error

The intercept function was used on Burp Suite to try to bypass the error message shown above. An account was registered with a different username and the request was intercepted and changed to match “TEST” (Figure 27). The attempt was blocked again, and the same error message was displayed on the website to alert us to an existing account.

Figure 27: Username changed using Intercept

2.5.5 CHECK FOR UNSAFE TRANSMISSION OF CREDENTIALS

Credentials submitted during both the login and registration processes were able to be viewed within the POST request after it was grabbed using Burp Suite. All credentials were able to be viewed in plain text within the request with no attempt to obfuscate the credentials or any attempt to secure them before transmission (Figure 28).

```
1  POST /index.php HTTP/1.1
2  Host: 192.168.1.20
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:18.0) Gecko/20100101 Firefox/18.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Referer: http://192.168.1.20/index.php
8  Cookie: PHPSESSID=uci5sjs1eq2g067nlanloi4086
9  Connection: close
10 Content-Type: application/x-www-form-urlencoded
11 Content-Length: 52
12
13 email=hacklab%40hacklab.com&password=hacklab&submit=
```

Figure 28: Plain text credentials after logging in

When accessing the admin section of the website using the credentials found in section 2.5.3 it was noted that the admin credentials were also able to be viewed in the POST request and no attempt to secure them was carried out before they were transmitted.

2.5.6 TEST FOR INSECURE STORAGE

When investigating the storage of details for the registered users it was found that the user's passwords were hashed once they reached the server. The passwords submitted after registering were transmitted and then hashed using the MD5 algorithm (E. Shacklett and Loshin, n.d.) before being stored in the database. This was proven by accessing the database backup found during section 2.2.2 which displayed the data stored within the tables (Figure 29).

```
birthday`, `Address`, `City`, `Contact_number`, `Gender`, `Email`, `Password`, `Date_created`, `status`) VALUES
. San Jose, Sitio IV, Dundee', 'Dundee', '09434138521', 'Male', 'sabellorichmon@yahoo.com', '11a00f3677902d1dec0aeccacc16d464', 'August 5, 2015 1
anila City', '09364987102', 'Male', 'benjiealfanta@yahoo.com', 'a432fa61bf0d91ad0c3d2b26ae8ace94', 'August 5, 2015 11:35:pm  ', 'active'),
 Rizal', '09109223103', 'Male', 'juliusfelicen@yahoo.com', 'fb154fdee061037d6f6bcec2eecfe688', 'August 12, 2015 4:07:pm  ', 'active'),
ity', '09364987102', 'Male', 'itchigo.aranzamendez@yahoo.com', '8eef495e2875ec79e82dd886e58f26bd', 'August 12, 2015 4:08:pm  ', 'active'),
Male', 'aa2000ent@gmail.com', 'dfc91587736b342423abefd7a2328de4', 'August 26, 2015 2:14:pm  ', 'active'),
San Jose, Sitio IV, Dundee', 'Dundee', '09364987102', 'Male', 'sabellorap@yahoo.com', '25f9e794323b453885f5181f1b624d0b', 'September 16, 2015 12:
```

Figure 29: MD5 hashed passwords in Database

Viewing the customer information pages within the admin section showed that the password for the user was obfuscated on the screen. The obfuscated password on the screen was easily accessible by viewing the page source code where the MD5 hash was available to be viewed and copied so that it could be cracked (Figure 30).

```
<div class="form-group">
    <label class="col-sm-2 control-label"for="inputPassword">Password</label>
    <div class="col-sm-10">
      <input type="password" name="password" class="form-control" name="midname" id="password" onchange="validation()"  value="7052cad6b415f4272c1986aa9a50a7c3" readonly/>
    </div>
  </div>
```

Figure 30: MD5 hash visible in Source code

After accessing the admin server page, it was possible to view the admin usernames and passwords from the user list section. Initially the User list page displays a blank table which was suspected to be empty. After viewing the source code for the page, it was discovered that details were stored in the table and that the text used had just been coloured white to blend into the background (Figure 31).

```
                    <tbody>
                        <tr class="del1">
                      <td><font color="white">ONLINE ORDERING Admin</font></td>
                    <td><font color="white">Benjie I. Alfanta</font></td>
                      <td><font color="white">BENJIE_OOS</font></td>
                       <td ><font color="white">e10adc3949ba59abbe56e057f20f883e</font> </td>
                                <td>
                      <a rel="tooltip"  title="Delete" id="<br />
<b>Notice</b>:  Undefined variable: id in <b>/opt/lampp/htdocs/studentsite/admin/ADMIN/SERVER/user.php</b> on line <b>88</b><br />
" onclick="return confirm('Are you sure you want to delete?')" href="delete_user.php?id=1" data-toggle="modal"    class="btn btn-danger"><i class="icon-trash icon-large"></i></a>
                       <a  rel="tooltip"  title="Edit" id="<br />
<b>Notice</b>:  Undefined variable: id in <b>/opt/lampp/htdocs/studentsite/admin/ADMIN/SERVER/user.php</b> on line <b>89</b><br />
"  href="edit_user.php?id=1" class="btn btn-info"><i class="icon-list icon-large"></i></a>
                                </td>

                                </tr>
                                <tr class="del2">
                      <td><font color="white">ASSET Admin</font></td>
                    <td><font color="white">Leo Aranzamendez</font></td>
                      <td><font color="white">hacklab</font></td>
                       <td ><font color="white">7052cad6b415f4272c1986aa9a50a7c3</font> </td>
```

Figure 31: Font colour set to white

Highlighting the table revealed the data that was stored within as well as it being visible in the source code. Both usernames and passwords for admin account were visible and the passwords were hashed using the MD5 algorithm, just like the customer details, which allowed them to be copied to be cracked. Each admin account was linked to a certain section of the admin area e.g. Asset Admin.

2.5.7 SUMMARY OF THE AUTHENTICATION MECHANISM

From the investigation of the authentication mechanism, it was clear that there were no quality checks being performed on submitted passwords and no notification was given when an insecure password was submitted. The website was also vulnerable to password guessing as the usage of hydra demonstrated through cracking a login for the admin section of the site that was valid. Usernames were found to be unique, and the same email/username was unable to be registered even with the use of the intercept function on Burp Suite. It was also discovered that credentials are not transmitted or stored safely as they were easily seen in site requests and were easily found within the admin section of the website where passwords were only hashed using MD5.

2.6 TEST THE SESSION MANAGEMENT MECHANISM

2.6.1 UNDERSTAND THE MECHANISM

The website made use of two different cookies that were assigned to the user only after logging in. After logging in as a customer the two cookies “Secret Cookie” and “PHPSESSID” were assigned to the user and used to validate authentication. After logging in to the admin area only the PHPSESSID was assigned to the user and no “Secret Cookie” was assigned. It was decided that the PHPSESSID cookie wasn’t relevant to testing (PHP Session & PHP Cookies with Example, n.d.) and was an arbitrary value set by the server as the Secret Cookie was thought to be the session token in use by the website.

2.6.2 TEST TOKENS FOR MEANING

After logging in and obtaining the secret cookie for the Hacklab account the cookie was tested to figure out meaning behind the construction of it. Using Burp Suite the cookie was identified to be encoded using base 64 (Figure 32).

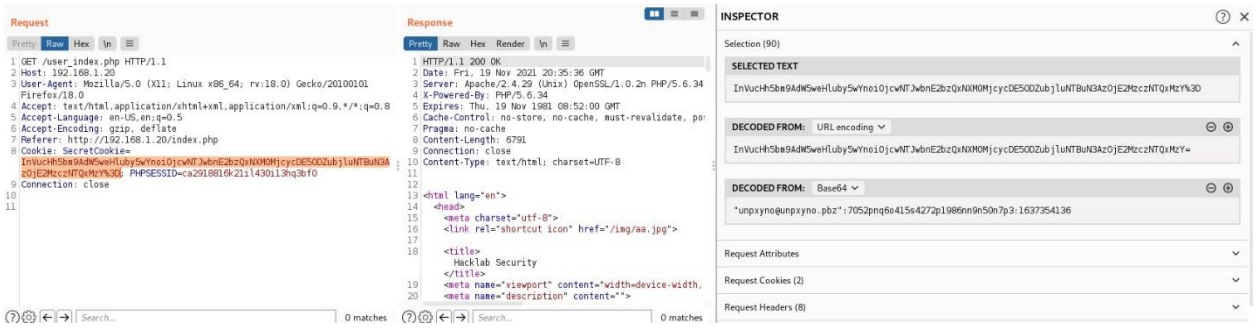


Figure 33: Burp Suite base 64 detection

After decoding using base 64, the overall structure of the cookie was evident but was still obfuscated using another type of encoding. After testing several other encoding types using the online tool CyberChef, ROT 13 was found to be the encoding used and fully revealed the content of

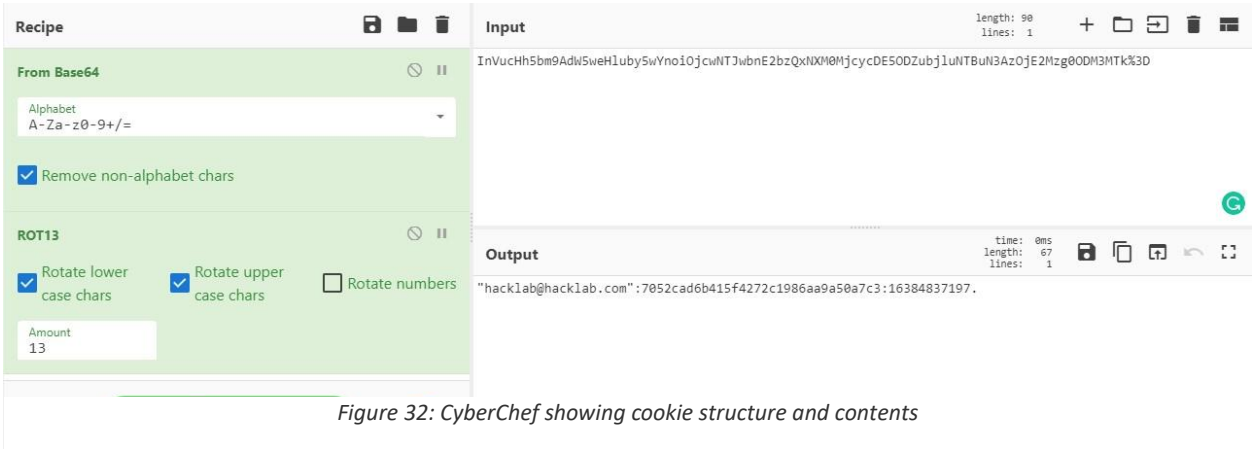


Figure 32: CyberChef showing cookie structure and contents

the secret cookie. Within the cookie was the username/email used to login, the MD5 hash of the account password and the time of login using Epoch time in seconds. The cookie followed the structure of “LoginEmail:MD5-HashedPassword:UnixTimeinSeconds” (Figure 33).

2.6.3 CHECK FOR INSECURE TRANSMISSION OF TOKEN

After successfully logging in and obtaining the Secret Cookie it was seen in the GET requests that followed, as the user browsed the site logged in, that the cookie was transmitted inside every request (Figure 34). After logging the user is directed to different webpages from a user that isn’t currently logged in. Where “index.php” would be the usual landing page for not being logged in, when navigating to the landing page while logged in directed the user to “user_index.php”

```
1  GET /user_index.php HTTP/1.1
2  Host: 192.168.1.20
3  Upgrade-Insecure-Requests: 1
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.159
   Safari/537.36
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;
   v=b3;q=0.9
6  Referer: http://192.168.1.20/index.php
7  Accept-Encoding: gzip, deflate
8  Accept-Language: en-US,en;q=0.9
9  Cookie: SecretCookie=InVucHh5bm9AdW5weHluby5wYnoiOjcwNTJwbnE2bzQxNXM0MjcycDE5ODZubjluNTBuN3AzOjE2Mzg0ODM3MTk%3D; PHPSESSID=
   avbiurre9clnanl0n5eokmp6t2
10 Connection: close
11
```

Figure 34: GET request containing cookie value

2.6.4 TEST SESSION TERMINATION

The session token was tested to identify any potential token expiration that may be active. The token was obtained by logging in on the 19/11/21 at 21:35pm and was tested at regular intervals to ascertain if it would expire. After 24hrs the token was still active, and it was presumed that if token expiration is in use then it is set to out with a full day. It could not be confirmed from testing if the token would eventually expire after the set time or if there was even an expiration time set.

After using the logout function on the website, the user was successfully logged out, but the session token was still visible in the requests made to the site. After logging out the termination of the session was tested by requesting a page requiring authentication by using the return button available in the browser to return to the previous page and by attempting to browse to “/user_index.php”. Both attempts failed to access the page that required authentication and it was decided that even though the token was still present in the requests it was unusable to provide authentication.

2.6.5 CHECK FOR CSRF

From the Zap report it was noted that several Anti-CSRF token alerts were detected by Zap and this suggested an obvious vulnerability for cross-site request forgery (Academy, 2021). A fake form that replicated the form that's submitted when changing a user's password. The form contained changed details that would change the password of an account, preventing the original owner from accessing it with their password. Using a python web server, the fake website was hosted and, while logged in to the test account provided with the password "Hacklab", the fake page was browsed to on a new tab. The from was triggered by pressing the "Save" button and then the fake form would be submitted, and the confirmation of details being changed was shown on screen (Figure 35).

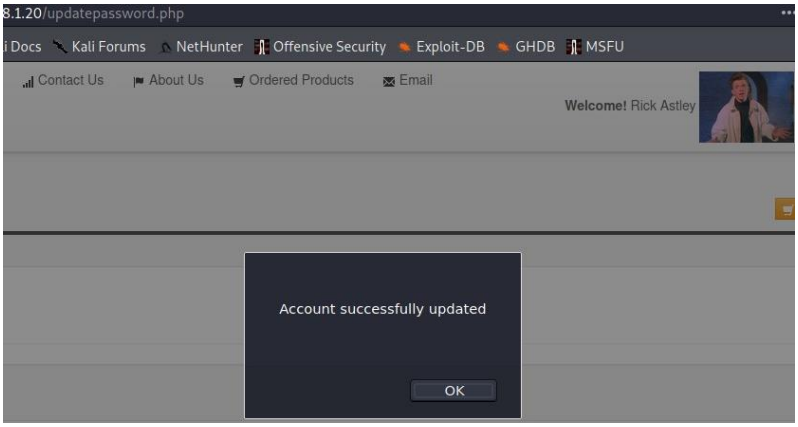


Figure 35: Account details Changed

The account login was then tested using the original password to ensure the change had occurred. The password was changed to "hacked1" and demonstrated how the website was vulnerable to an attack like this. The fake form constructed, and other steps of the process can be seen in Appendix G.

2.6.6 SUMMARY OF SESSION MANAGEMENT

After completing the testing of the session management of the website, the results showed a number of different vulnerabilities that impacted the site. The token "SecretCookie" in use on the website contained very sensitive login data and was easily captured and decoded during the test to reveal the login details on the account. The token used was transmitted in all requests after logging into the account and the token was still present in the request after the user had signed out. Following this a successful CSRF attack was completed where a fake change password form was hidden on a webpage and required a button to be pressed for the password to be changed to the

preset one in the form. This was hosted on a python web server and required an active login to be open in another tab.

2.7 TEST ACCESS CONTROLS

From mapping the website, different horizontal and vertical access controls were identified. Horizontal access controls were broken as it was found during testing that customers could access other customers receipts and view sensitive details related to the customer.

Vertical access controls were broken as regular customer accounts were able to gain access to admin sections of the website by browsing to "/admin/ADMIN". From here users hashed passwords, private details, and other sensitive data to both the website and its users could be accessed without having proper authentication.

2.8 TEST FOR INPUT BASED VULNERABILITIES

2.8.1 TEST FOR SQL INJECTION

Multiple areas on the website were identified from the initial scanning as being vulnerable to SQL injection, the login function was tested using " ' OR 1=1;-- " in an attempt to beat the requirement for a valid password to an account. A filtering script was in use as this input was detected by the website as being malicious and an alert was displayed once it was submitted (Figure 36).

Figure 36: SQL Injection Alert

The search functionality within the customers table contained in the admin section of the site was tested and used to identify the number of columns within the table being used, after testing using " ' ORDER BY 1# " and incrementing until an error is shown it was found that the table had 14 columns which matched what was previously discovered within the database backup that was found.

An application called SQLMAP was then used to automate the process of injection (sqlmap: automatic SQL injection and database takeover tool, 2021) with the initial injection location being the URL for the product details page as it used a variable to display the correct page. Through the

use of SQLMAP the complete set of tables for the website were identified and could easily be listed to be viewed and their contents dumped (Figure 37).

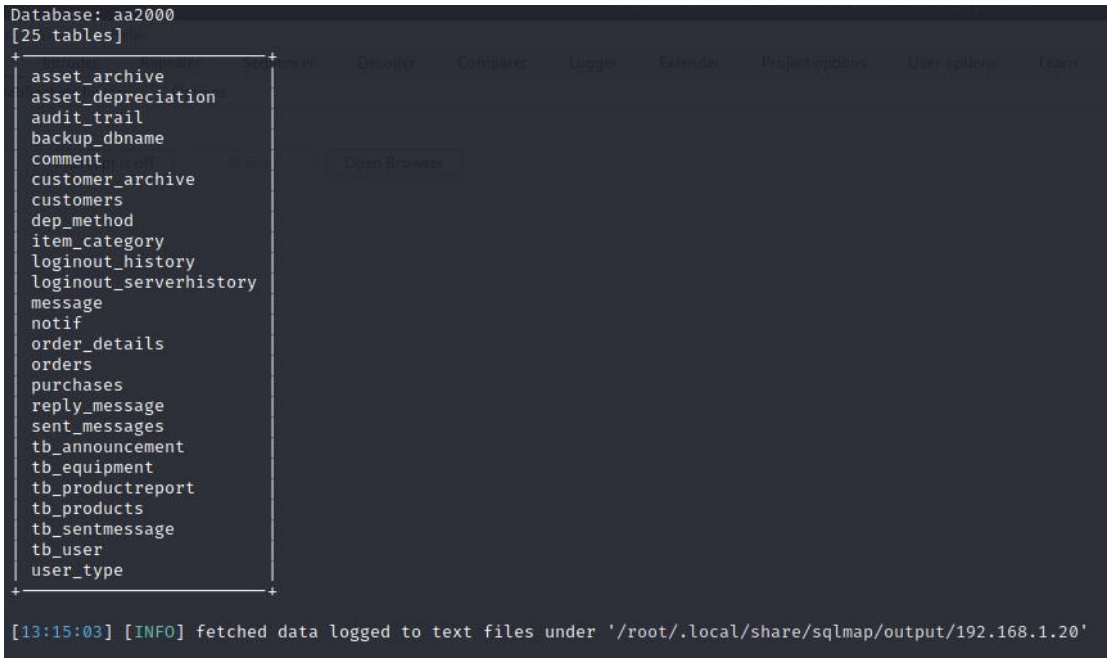


Figure 37: List of Tables being used

As well as dumping the contents of the tables, SQLMAP can also perform dictionary attacks on hashes that it detects. The admin user table was found in "tb_user" and the hashes were cracked using the dictionary attack available. The contents were then dumped along with the cracked hashes for each user within the table (Figure 38).

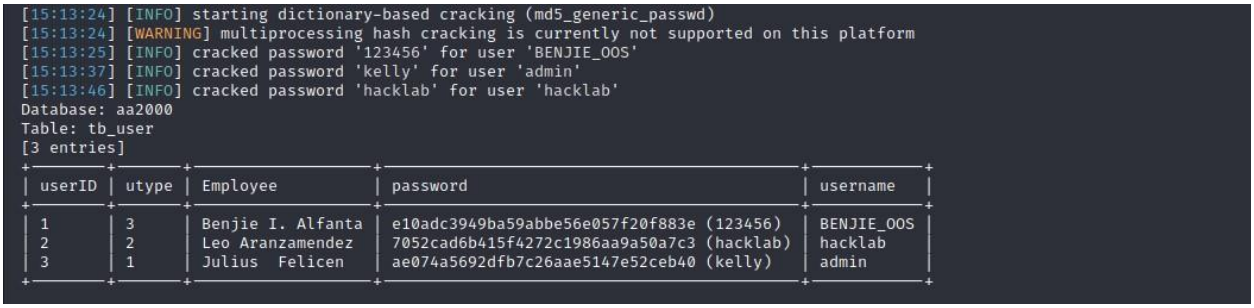


Figure 38: tb_user contents dumped after being cracked

2.8.2 TEST FOR XSS AND OTHER RESPONSE INJECTION

XSS (Cross site scripting) was tested for through several sections of the website. Multiple areas of the website were susceptible to stored Attacks that could be triggered when visiting the page (Cross Site Scripting (XSS) Software Attack | OWASP Foundation, 2021). The announcement section was accessed using the admin login discovered previously and allowed for an announcement to be

created that contained a simple script that would alert out “Hello”. This announcement would appear every time a user accessed their profile page on the site and was tested as an account was logged into and after selecting the profile page the script was triggered and was displayed on the screen (Figure 39).

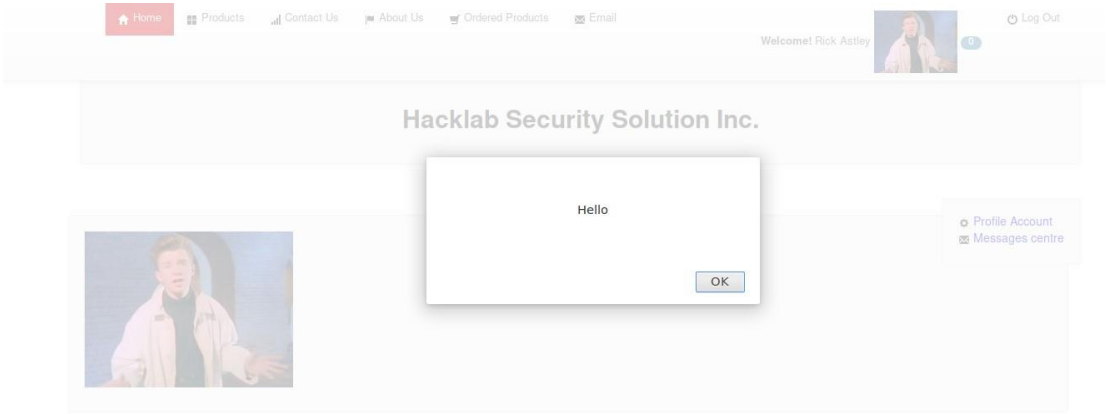


Figure 39: Stored Script activation on profile page

Another stored attack was created by accessing the admin section that handles the products displayed to be purchased where a fake product was created that had a script contained within the product description. During testing the script was placed in the products name and would trigger once the product was visible on the page, it was later moved to the product description, so the product page had to be visited for it to trigger. The script used would output the user’s current cookie to the screen when it was triggered (Figure 40). After setting up the fake product it was tested again by browsing to the product page and viewing the product where the script would trigger, and the user’s cookie would be visible on screen.

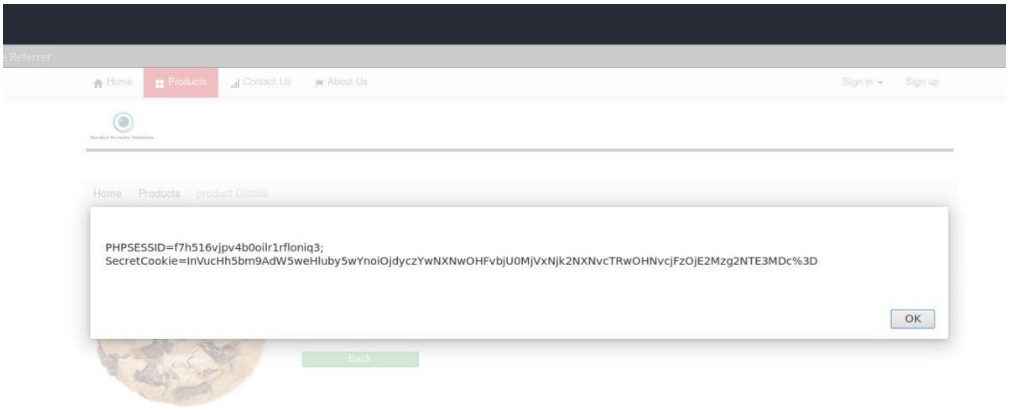


Figure 40: Cookie displayed due to script activation

The first script demonstrated on the user profile page that alerted the phrase “Hello” was created by Replacing the message section of the announcement with the script “<script>alert(“Hello”)</script>”. Using the following script “<script>alert(document.cookie)</script>” the user’s cookie was printed to the screen as seen on the product page created for the fake product.

2.8.3 TEST FOR PATH TRAVERSAL

To test path traversal, a webpage that accepted variables within the URL was found (Academy, 2021) and used for the test. The products page (/products.php?page=) was to test the possibility of path traversal to access the etc/passwd file location. Accessing the location through the URL was unsuccessful and was attempted again using URL encoding which also failed. Although no error was displayed on screen, no access to the location was completed (Figure 41).

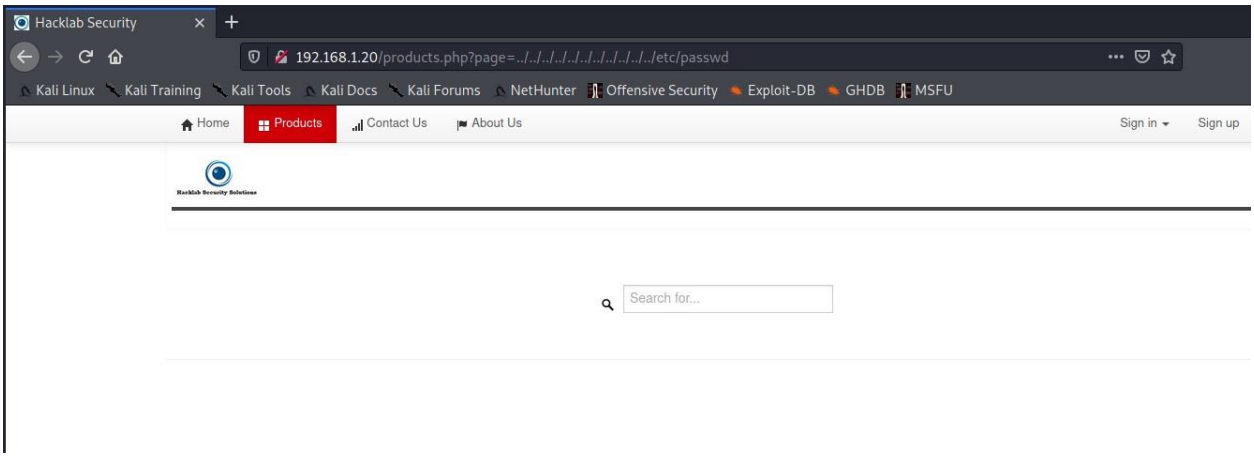


Figure 41: Path Traversal Attempt

Directories that were previously found during the mapping stage were able to be browsed and viewed freely. These directories include "/database" which revealed a copy of the SQL database being deployed on the site and "/pictures" which held the images that had been uploaded to the site through the profile picture change function available to signed in accounts (Figure 42).

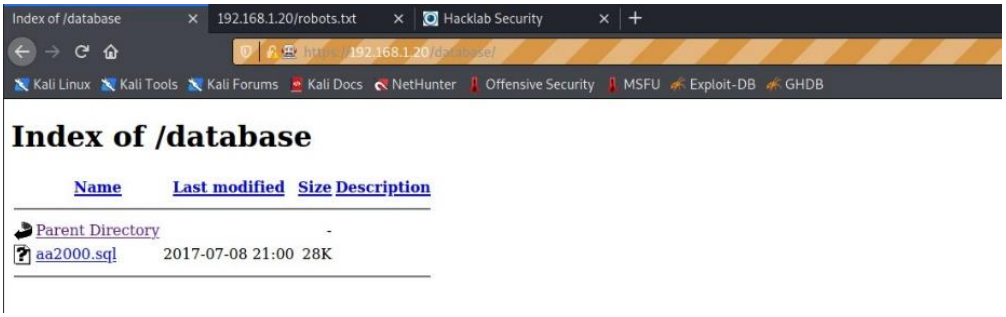


Figure 42: Access to /database directory

2.8.4 TEST FOR FILE INCLUSION

It was identified that there was an upload file functionality within the profile page which allowed for new profile pictures to be uploaded where they would be saved in the "/pictures" directory. A PHP shell was created, using weevely (Prodromou, 2020) that used the password "Hacklab" to access it, by entering the command "weevely generate hacklab /root/Desktop/weevely.php" within the kali machine provided. The shell was then uploaded through the profile page. The upload failed and an alert was shown that related to the file type that was being uploaded. Using Burp Suite and the intercept function, the content type of the upload was changed to represent a .jpg file as the file had been proven to be accepted (Figure 43).

```
------WebKitFormBoundaryLyVjDFrBN1bSQC10
Content-Disposition: form-data; name="uploadedfile"; filename="weevely.php"
Content-Type: image/jpg
```

Figure 43: Content type changed using intercept

This allowed the file to be uploaded to the website as a php file and be stored in the directory mentioned above (Figure 44). Browsing to the file location (192.168.1.20/pictures/weevely.php) activated the shell so that it could be accessed from another machine.

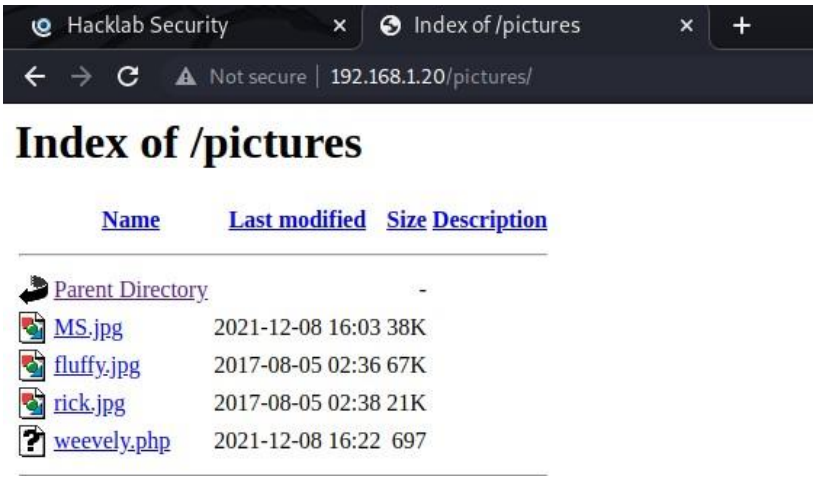


Figure 44: File upload to the /pictures directory

Opening a terminal and using the command "weevely http://192.168.1.20/pictures/weevely.php hacklab" opened the shell that had been activated and allowed access to the system (Figure 45). Using the shell important information could be found out about the server running the website. System information was found and included information on the operating system, client ip address and the document root (Figure 46).

Figure 45: Shell being accessed after activation

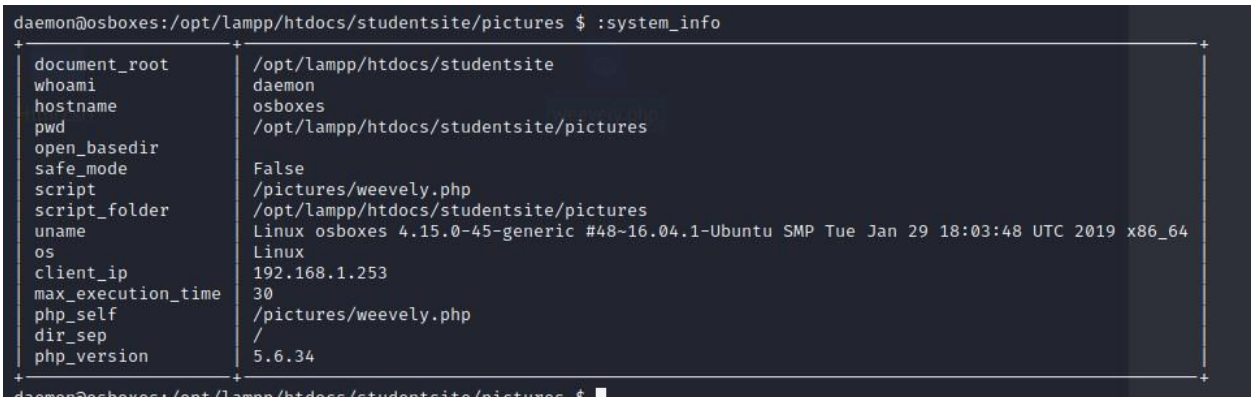


Figure 46: System info accessed through shell

2.8.5 SUMMARY OF INPUT BASED VULNERABILITY

After testing for input based vulnerabilities, a large number of different types of vulnerabilities were found. SQL injection was possible through the URL of a products details page, this allowed SQLMAP to be used to explore the database in use and dump the contents of different tables such as the customers and admin logins.

XSS was also possible using the admin function to create announcements that appeared in customers profile pages and through product pages where a fake product containing a script was created.

Path traversal was not possible through the URL in an attempt to access files such as “passwd” but some directories, found during the mapping stage, were available to be explored and files within them were able to be accessed and investigated.

The attempted file inclusion was successful as a weevely shell was able to be uploaded to the server and was activated and accessed from another machine where information on the server could be collected. The profile picture update functionality was vulnerable and was attacked using Burp Suite to bypass the file checks that was done before a successful upload.

2.9 TEST FOR LOGIC FLAWS

Testing for logic flaws was largely focused upon the transaction logic of the website from adding products to cart to checking out. The Tamper Data tool present within OWASP Mantra was used to identify any flaws. First a quantity of 10 of the same item from the store was added to the basket, tamper data was used to intercept the request before it had been added to the basket and allowed the opportunity to change variables within the request such as price and quantity (Figure 47).

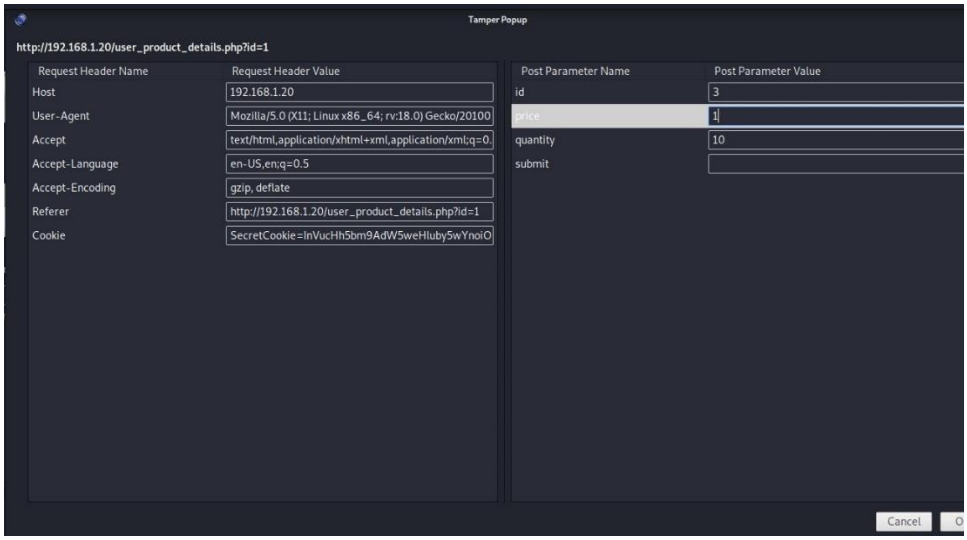


Figure 47: Tamper Data variables

The product added originally was £300 on the website but was altered to cost £1. The request was successful, and the basket represented the change that had been made with the basket totaling £10 instead of £3000 (What are Business Logic Flaws on Web Applications?, 2015). The process was repeated with a single change, after intercepting the request before it entered the basket the price was now changed to represent a negative number, essentially meaning if the purchase was successful then money would be due to the customer. The product was set to -£1000 again with a quantity of 10. The request was once again successful and was represented inside the basket when attempting to checkout showing that the transaction logic is flawed and vulnerable (Figure 48).

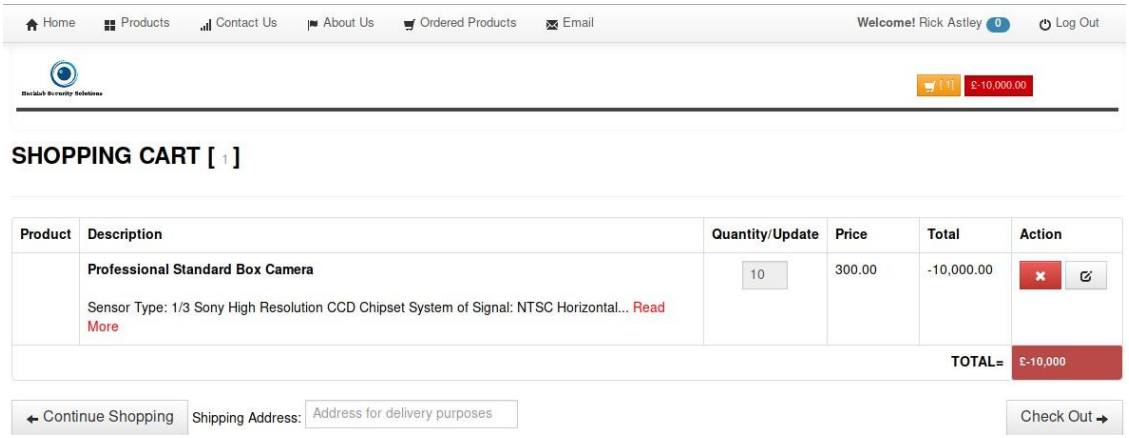


Figure 48: Negative total in basket after tamper

2.10 Miscellaneous Checks

It was discovered that the website wasn't using a valid SSL certificate and was vulnerable to several potential risks that not having a valid SSL certificate would create. Attackers would be able to see information sent in plain text due to the connection not being secure and so the data isn't transmitted securely. Due to having an invalid certificate the website could not be validated as being the official site and so would allow attackers to replicate the site with customers being unable to tell the difference between them as they would both lack the validation from the certificate.

2.11 Information Leakage

During testing a wide range of verbose error messages, that contained sensitive data, were shown or able to be seen. These errors contained important info such as server directories and operating systems being used on the server. Having these errors displayed allows for easier mapping of the website and for attacks to be planned much easier. 404 errors were shown when attempting to browse to a webpage that does not exist. This error page contained information on the server being used for the website (Figure 49).

Figure 49: Error 404

Error reports were also shown after an unsuccessful login to an account. The error displayed file locations and directories that were sensitive to the server and should not have been disclosed (Figure 50).

```
                    <br />
<b>Warning</b>:  include(sqlcm_filter.php): failed to open stream: No such file or directory in <b>/opt/lampp/htdocs/studentsite/index.php</b> on line <b>83</b><br />
<br />
<b>Warning</b>:  include(): Failed opening 'sqlcm_filter.php' for inclusion (include_path='.:/opt/lampp/lib/php') in <b>/opt/lampp/htdocs/studentsite/index.php</b> on line <b>83</b><br />
                <script type="text/javascript">
```

Figure 50: Source code of error shown on screen

3 DISCUSSION

3.1 SOURCE CODE ANALYSIS

After completing the initial testing of the web application, the source code was provided and was examined and analyzed. Using both Notepad++ and VScode the source code files for all the webpages were examined for any vulnerabilities that may be present. Any countermeasures to the vulnerabilities found will be included in the Vulnerabilities and Countermeasures section.

3.1.1 INFORMATION LEAKAGE

Sensitive information being leaked was looked at first and presented a dangerous vulnerability if any instances were to be located. Within the PHP files, the connection to an SQL server was handled through a separate “connect.php” file which contained the login info for the connection (Figure 51). This prevented any leakage of these details on the web pages where it was used. Leakage of this information was spotted within the “changepicture.php” file which failed to use the premade connection file which led to the details possibly being leaked (Figure 52).

connect.php

```
1  <?php
2      $con=mysql_connect("localhost","root","Thisisverysecret21") or die ("DOWN!");
3          if ($con) {
4              mysql_select_db("aa2000",$con);
5
6          }
7          else
8          {
9              die("DOWN");
10         }
11 ?>
```

Figure 51: connect.php file

```
# Update database
##############################################
$con=mysql_connect("localhost","root","Thisisverysecret21") or die ("DOWN!");
mysql_select_db($DBTable,$con);
```

Figure 52: Details leaked within php file (line 66-67)

Within the source code, it was seen that passwords for user account were being directly displayed from the server to the customer section of the admin area. Although being obfuscated on-screen and hashed using MD5, it was still possible to copy the un obfuscated hash from the page source and crack it to reveal the password (Figure 53).

```
  </div>
<div class="form-group">
    <label class="col-sm-2 control-label"for="inputPassword">Password</label>
    <div class="col-sm-10">
      <input type="password" name="password" class="form-control" name="midname" id="password" onchange="validation()"  value="<?php echo $row['Password'];?>" readonly/>
    </div>
  </div>
```

Figure 53: User password echoed in source code

After accessing the admin section of the site, a list of admin accounts located was found but the information stored within the table had been coloured white to hide it. The login details were being displayed on the webpage and were clear to see by highlighting the table or by viewing the page source. Although hashed with MD5 again, cracking the hashes took little time once they had been obtained.

Comments within the source code also provided a dangerous information leakage point and multiple instances of sensitive information being leaked directly onto the live page source were seen. The file “hidden.php” would present as a blank screen when navigated to but after opening the page source up a comment relating to a possible door code was found (Figure 54).

```
<!-- ***Note to self: Door entry number is 1846 -->
```

Figure 54: Hidden.php hidden comment

Another comment containing sensitive information was found inside the “user_products.php” page where a file path was detailed within the comment (Figure 55) and the information could potentially be used in an attack.

```
<!-- *** Note that the path is /home/tc/.local/bin:/usr/local/sbin:/usr/local/bin:/apps/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/local/java-sun/jre/bin:/etc/sysconfig/tcedir/ondemand -->
<?php include('include/connect.php');
```

Figure 55: Hidden file path comment

The last instance of comments containing sensitive information was noted on the “user_account2.php” page where contact details for a “Denis Smith” were present. An email address and a mobile number were contained within the comment along with a comment suggesting that “Dennis” potentially had access to the webpages and could have been responsible for the creation of webpages on behalf of the Client (Figure 56).

```
<!-- *** Denis Smith, d.smith@hacklab.com, phone number 01382 99999. Php expert. -->
```

Figure 56: Contact details displayed in comment

Within the “cookie.php” file the method of creating the “SecretCookie” token was revealed and proved the token was created using base 64 and rot 13 encoding as was previously tested. It was also confirmed that the cookie was using the username and password of the user along with the Unix timestamp (Unix Epoch) during its creation (Misja.com, no date), and in turn, leaking these details (Figure 57).

```php
<?php
$str=$username.':'.$password.':'.strtotime("now");$str = base64_encode(str_rot13($str)); setcookie("SecretCookie", $str);
?>
```

Figure 57: Cookie creation file

Inspecting the customer login from the index.php page shows that username enumeration was possible. Supplying a correct username during the login attempt would return a response of “Invalid Username or Password” whilst entering an incorrect username would respond with “Username not found” allowing for valid usernames to be successfully enumerated. Allowing for username enumerating enabled finding user accounts to be a suitable attack (Laverty, 2017).

3.1.2 XSS

No protections designed to prevent XSS attacks (Web Security Academy, no date) were recognised within the source code for the website. After examining an area that was tested such as the products page where an XSS exploit was created and exploited. Viewing the source code for both the webpage responsible for submitting the details of the new product to the server to be stored (Figure 58) and the webpage responsible for displaying these details to the user (Figure 59) showed that there was no cleaning in place when the product details are first submitted when creating the product initially.

```php
if (isset($_POST['submit'])) {

   $name = $_POST['name'];
    if(is_string($name)){
     $price = $_POST['price'];
     if(ctype_digit($price)){
    $quantity = $_POST['quantity'];
       if(ctype_digit($quantity)){

    $name = $_POST['name'];
     $price = $_POST['price'];
    $description = $_POST['description'];
    $quantity = $_POST['quantity'];

     date_default_timezone_set('Asia/Manila');
    $new =date('F j, Y g:i:a  ');

     //image
$image = addslashes(file_get_contents($_FILES['image']['tmp_name']));
$image_name = addslashes($_FILES['image']['name']);
$image_size = getimagesize($_FILES['image']['tmp_name']);

move_uploaded_file($_FILES["image"]["tmp_name"], "../SERVER/AS/products/" . $_FILES["image"]["name"]);
$location = "products/" . $_FILES["image"]["name"];

mysql_query("insert into tb_products
  (productID,name,price,image,details,quantity,date_created) values
('','$name','$price','$location','$description','$quantity','$new')")
or die(mysql_error());
```

Figure 58: Product details submitted during creation

```php
<hr class="soft clr"/>
<p>
<?php echo $row['details'];?>
</p>
</div>
```

Figure 59: Retrieving the details previously submitted

After browsing the newly created product page, the page retrieves the details from the server, and the malicious code is executed.

3.1.3 SQL INJECTION

SQL filtering (Portswigger, no date) was found to be active on the “index.php” webpage and from the source code, the filter in use was able to be seen. The filter was stored in another file called “sqlcm.php” and was included on the index page to filter the details being passed from the user login (Figure 60). From the filter, it's seen that once the username is submitted it is then checked against any of the matching items set and returns an output to the screen if any are detected. Although the filter was active, bypassing it was possible with the addition of extra spaces and changing of letter case used.

```
<?php  if(preg_match("[1=1|2=2|Select|Union|'b'= 'b'|1 =1|'a'='a']", $username)){ echo '<script language="javascript">'; echo 'alert ("Bad hacker.We are filtering input because of abuse!");'; echo 'window.location.href="index.php";'; echo '</script>'; die(); }  ?>
```

Figure 60: SQL Filter File

Within the source code, no prepared statements were seen to be used (Figure 61) and multiple instances of user direct input being included in SQL queries without being properly cleaned were noted including the user login page. After bypassing the filter being used, there was no other defense against SQL injection being used.

```
$username='"'.$username.'"';
$sqlquery="select * from customers where Email=(".$username.")";

                    $query = mysql_query($sqlquery) or die(mysql_error());
                    $rows = mysql_num_rows($query);
                    $row = mysql_fetch_array($query);
```

Figure 61: Lack of prepared statements

After inspecting the admin section “index.php” file, although no prepared statements are used an attempt to clean the input was made. The use of the user-created function “clean” created using several PHP functions (PHP, 2001) used to remove whitespace, special characters, and slashes (Figure 62) was used to sanitise the incoming input from the user.

```
function clean($str) {
        $str = @trim($str);
        if (get_magic_quotes_gpc()) {
            $str = stripslashes($str);
        }
        return mysql_real_escape_string($str);
    }
    $username = clean($_POST['username']);
    $password = clean($_POST['password']);
    $pass=md5($password);

    $query = mysql_query("select * from tb_user where username='$username' and password='$pass'") or die(mysql_error());
```

Figure 62: "clean" Function created

3.1.4 PATH TRAVERSAL AND LOCAL FILE INCLUSION

From the website testing, it was possible to navigate certain directories and from reviewing the source code it was confirmed that directory browsing was enabled. Examining the ".htaccess" file revealed, "Options +Indexes" which refers to the ability to browse directories that do not have default files such as "index.php" and allowed for the directories to be accessed and for additional information to be gathered (Oracle, 2004).

Path traversal was possible through the "affix.php" file which was used within the file "footer2.php" which was used on 11 web pages within the website. Utilizing the function of the "affix.php" file and inputting the directory location as the variable allowed for directories and files that were unavailable through the website to be accessed such as the passwd file (Figure 63).

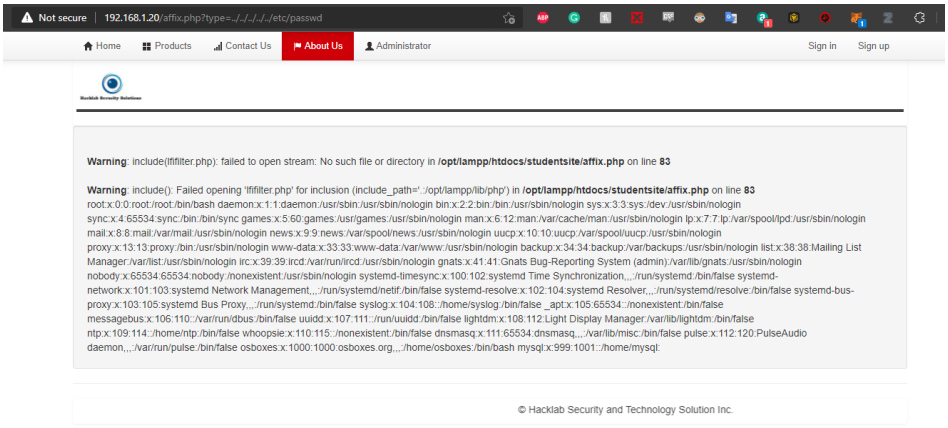


Figure 63: Passwd file accessed

With directory browsing allowed and the source files available, a hidden folder was found at the "192.168.0.20/error/" address. The folder was not accessible using a browser (Figure 64) and so browsing the source code for this folder revealed a backup for the SQL injection filter being used on the site.

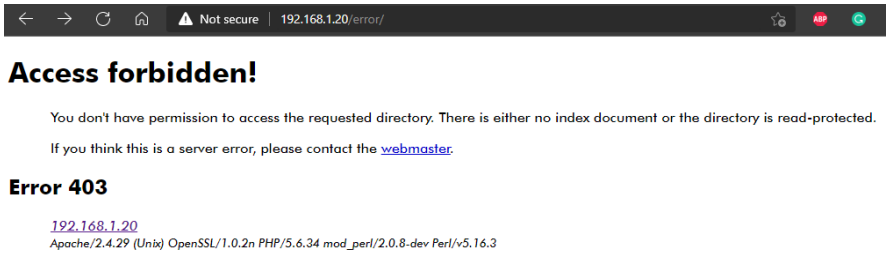


Figure 64: Folder inaccessible

3.1.5 FILE UPLOAD

After viewing the source code, the validation conditions that were being used to check the uploads from the user profile picture change were found. The conditions checked the file type, extension,

and size (Figure 65). The conditions were able to be bypassed by changing the file type within the request before it was submitted. Executing the file allowed for a shell to be activated on the server and system information was able to be gathered on the server running the site.

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

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

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

Figure 65: Filter Conditions

3.1.6 COOKIE ATTRIBUTES

After viewing the cookie creation file “cookie.php”, it was also noted that attributes on the cookie hadn’t been set when it was created. When looking at the “setcookie” function (Figure 66) and comparing it with the cookie creation file, it's seen that only the name and value of the cookie have been set.

setcookie(name, value, expire, path, domain, secure, httponly);

Figure 66: Cookie creation syntax

Importantly the “httponly” attribute of the cookie hasn’t been set during creation. This attribute helps protect against cross-site scripting as it prevents the cookie from being accessed by the malicious script being used (PHP, 2001).

3.2 VULNERABILITIES DISCOVERED AND COUNTERMEASURES

Once the testing phase had been completed and the source code had been assessed, all the found vulnerabilities were compiled into a list and possible solutions and/or countermeasures to solve the known vulnerabilities were researched and are recommended to be implemented into the site as soon as possible.

3.2.1 INFORMATION LEAKAGE ISSUES

3.2.1.1 ROBOTS.TXT AND PHPINFO

Browsing to “robots.txt” displayed information located within the file on the disallowed webpage (Google Developers, 2021). This file protects web pages from spidering attempts and limits web crawlers to avoid site overloads. The file showed a disallow rule in place on info.php (Figure 67) and prevent spidering techniques from being used on the page.

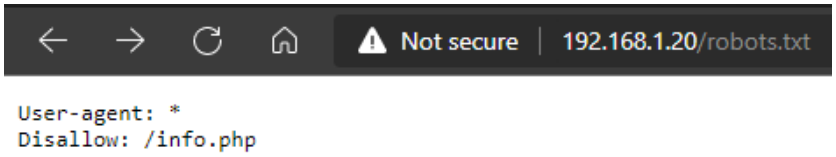


Figure 67: Robots.txt

This pointed towards information being located within the webpage that would be of interest during the test. Browsing to info.php revealed a large amount of important information relating to the server.

To fix this issue the info.php file inclusion should be removed from the robots.txt file as info.php shouldn’t be included in the live directory as it contains sensitive information and should only be used for testing and pre-release purposes to avoid leaking information on the system.

During testing the webpage “phpinfo.php” was also located and contained identical information to that contained in info.php. As the same as the vulnerability above, this webpage shouldn’t be listed within the active directory and shouldn’t be in use on the website to be freely browsed. The page should be removed from the active directory and only used when needing to access the information provided by it.

3.2.1.2 CUSTOMER PASSWORD DISPLAYED

Hashed versions of customer passwords were found to be accessible from within the admin section of the site due to the password being called from the database to be displayed on the customer detail page while being obfuscated. Accessing the page source revealed the passwords hashed version and it was possible to copy the hash and crack it.

To solve this issue, removing the displaying of the user’s password would eliminate any chance of it being seen and the password shouldn’t be accessible through the admin section of the site as it is insecure. Along with removing the password being displayed, the use of a different hashing algorithm that is secure, such as using bcrypt over MD5 which has been broken and proven to be insecure, along with the introduction of salting passwords before being hashed to reduce the chance of the password being cracked if it is stolen.

3.2.1.3 SERVER DETAILS LEAKED

Throughout the website, the use of the “connect.php” file was used to store the connection credential and was called whenever a connection was needed, as a result, the connection details were never exposed within a webpage and were hidden in the file. One instance was noted where the connection file wasn’t used, and the connection details were leaked to the active webpage. The changepicture.php webpage was responsible for leaking the details after not using the user-created connection file.

This single instance is easily fixed by swapping the connection details within the file to use the connection file currently used in all other pages with a connection. This ensures the details aren’t seen directly within the file and are stored within the connection file like other web pages.

3.2.1.4 HTTP USAGE

Throughout the webpage, HTTP is used instead of using HTTPS (Cloudflare, no date). This creates a vulnerability as while using HTTP the data being sent to and from the server is not encrypted and can be freely seen when communicating with the server. The data being sent to the server was also able to be tampered with as seen during the investigation and the usage of HTTPS would have prevented both attacks.

Using HTTPS within the site would fix the potential vulnerabilities that can impact HTTP. HTTPS uses an encryption protocol to encrypt all data being sent to and from the server. To use HTTPS an SSL certificate has to be active on the webpage ensuring HTTPS is used over HTTP. Although packets can still be sniffed while encrypted, the data gathered is useless compared to when they are sniffed from an HTTP connection.

Ensuring the whole website and everything within it uses HTTPS is important as mixing HTTPS and HTTP can create pages called mixed content pages (MDN, 2021) where even though the page is protected with HTTPS, certain items retrieved using HTTP leaves the website only partly encrypted as malicious attacks can still take place on HTTP items such as an HTTP image request.

3.2.1.5 COOKIE USAGE

Using user credentials to create the cookie for the website creates a serious vulnerability of leaking user data. The current cookie creation fails to set the httponly attribute and means that the cookie can be stolen and used in another attack such as an XXS attack.

Ensuring the httponly attribute and, if HTTPS is in use, the secure attributes are set can ensure the cookie will be better protected and can’t be stolen to be used in conjunction with another attack. The httponly attribute ensures that JavaScript cannot read the cookie in the case of an XSS attack. Using the secure attribute ensures that the cookie can only be sent over an HTTPS connection and removes the chance for eavesdroppers to grab the cookie.

3.2.1.6 DIRECTORY NAMING SCHEME

Within the webpage, vulnerable and sensitive directories not intended for customers to visit were uncovered through bruteforcing with wordlists. Areas such as the admin login page and the directory “database”, which contained a copy of the SQL database being used were discovered using dirb and dirbuster.

Naming these directories with unique names only known to those who require access can help eliminate the chance of the average customer or malicious attacker discovering the webpage. With the database directory, even if it had been renamed to be harder to find the chance of an attacker finding it would still exist. Storing a backup on a directory that is live on the server should be avoided as it doesn’t serve any benefit and can result in the loss of extremely sensitive data.

3.2.1.7 VERBOSE ERROR REPORTING

After providing the webpage with an invalid request, the error page that is produced displays additional information that is not related to the error. As well as displaying the error code, important information on the system being used is displayed such as software version numbers which allow for information to be easily gathered on the server (Figure 68).

Limiting the error page to only displaying information relating to the error such as the error code and description would eliminate the issue of displaying additional information that relates to the server and is unrelated to the current error.

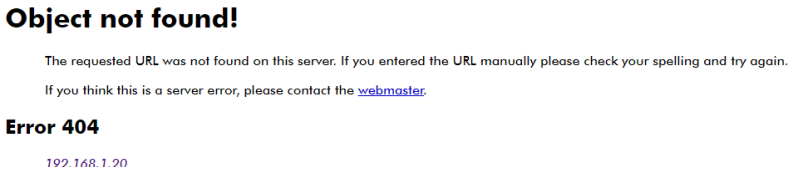


Figure 68: Error page displaying additional info

On-screen errors relating to SQL requests were also shown after an unsuccessful login. These errors revealed the filesystem setup being used as listed directory paths were contained within the error message. Disabling these errors on the live website would prevent this issue and error messages such as these should only be enabled during testing and shouldn’t be active on a live site.

3.2.1.8 HTML COMMENTS

Multiple comments not pertaining to the actual website code within the source files for the website were located. These comments included secure door codes, contact details for someone, and a system file path.

These comments should be removed quickly as they serve no purpose to the site and contain information that could be used to help assist any attacks on the server and any physical locations in relation to the door code. Personal details have been leaked within one of the comments and this information could allow for the person mentioned to be a victim of a potential scam or phishing attempt separate from the attack on the website.

3.2.2 AUTHENTICATION

3.2.2.1 USERNAME ENUMERATION

Customer account usernames were able to be enumerated due to the responses from the server in relation to a correct and incorrect username submission. This would allow for a program such as hydra to try a wordlist against the website and return a list of usernames that potentially would be useable. This would lead to further attacks using the usernames found to brute force passwords for them.

To prevent this, the error message shown after a successful username or an unsuccessful username should be the same and should state “Invalid Username or Password” to avoid giving away any information on what details may or may not be correct (OWASP Foundation, no date).

3.2.2.2 PASSWORD QUALITY

During the user registration on the website, a password length limit of between 7-14 characters is set. This limit allows for easier password brute-forcing due to the fact the length of the password is locked to a max length of 14. Minimum password lengths work well to avoid small passwords, but the maximum limit should be removed to allow for longer passwords and to avoid limiting passwords to be easier to crack.

From the testing phase, it was seen that even after submitting very weak passwords that fit the rules set, no additional quality checks were done. Although passwords met the rules set, they were still very insecure and were easily guessed using a wordlist. Additional quality checks to check for repeating numbers, potential dates and other things such as popular names and well-known items should be implemented to perhaps alert the user while signing up that it is a weak password or allowing the user to create the account and then alerting them that the password, they chose should be updated to be stronger. This change would make it harder for passwords to be cracked should usernames ever be obtained (Avast, 2018).

3.2.2.3 PASSWORD LOCKOUT

Currently, the website operates no password lockout/ account lockout which allowed for an application such as hydra to be used to successfully brute force a list of users and passwords commonly used. Without a policy that suspends logging into an account for "x" number of minutes after several incorrect login attempts, once a successful username has been found attackers can attempt to login to that account without any delay mechanism.

Implementing a lockout policy after a few incorrect logins that suspend logging into an account for several minutes can prevent these attacks or at least slow them down (Ncsc.gov.uk, 2018). Using a compounding time out would increase security and after a few lockouts on the same account, the timeout could be several hours after starting at only several minutes.

Alerting a user to any lockouts that had occurred on their account after they successfully login would allow users the chance to change their details in relation to an attack that may have occurred but failed.

3.2.2.4 ADMIN LOGIN

The password cracked for the admin section was "kelly" and was found to be very weak and easily found on a wordlist. Ensuring properly built and chosen passwords are used or even passphrases to limit the chance of the passwords for the admin section of the website appearing on a wordlist used in a brute force attack. Ensuring passwords meet the requirements applied to regular users' passwords should be implemented at an absolute minimum due to the importance of the admin area and the consequences of unauthorised access.

The implementation of a 2FA system used in connection with the admin login in page would prevent unauthorised access even if the credentials had been obtained (Authy, no date). After successfully logging in a code could be requested using an authenticator app or login approval could be requested using apps to ensure access isn't achieved by anyone who should have access. This method would also act as an alert if credentials have been stolen as a notification could be received when a code is requested, or an email could alert an admin to an unusual login from a certain location.

3.2.3 SESSION MANAGEMENT

3.2.3.1 SESSION TERMINATION

Testing the termination time of active sessions revealed that there was likely no termination time set. During the testing stage, an active session was left for 24 hours while being regularly tested to confirm if it was still active or not. No session timeout/termination is set up by default and so including a session termination timing should be considered.

Setting a termination time as it can limit the time a potential attacker has to hijack the session. With no timeout, the attacker has no time limit to acquiring access to the session. This would be especially important in admin areas as these areas may be targeted more due to the increased privileges and access the session would allow.

3.2.3.2 CSRF

The site was vulnerable to a CSRF attack and an attack was demonstrated during the testing and was successful in locking the user out of their account. A fake form that was created allowed for an attacker to change details of another user's account to ones they had set without ever having access to the user's account, all that was required was for the user to browse to the link while logged into the website.

To prevent this type of attack, anti-CSRF tokens should be used as it provides a unique, secret, and unpredictable value that is used to verify that the requests are from the correct server. The server generates the token and sends it to the user to be used in further requests, and when another request is made the tokens are compared, and if there is no match or if the token is missing then the request to the server is denied (Web Academy, no date).

3.2.4 INPUT BASED VULNERABILITIES

3.2.4.1 XSS

Several sections of the website were vulnerable to cross-site scripting attacks and these attacks were demonstrated within the testing phase above. Issues relating to these vulnerabilities were discovered within the source code of the website.

Implementing the usage of the function "htmlentities" will neutralise the injected code by converting any character with an HTML equivalent to HTML entities effectively disabling the potentially malicious script that had been placed (PHP, 2001). Sanitising all entries using this command will ensure that scripts, even when injected into areas of the website, will be disabled, and will simply be output as HTML instead of being successfully run.

3.2.4.2 SQL INJECTION

Although a filter had been created and was in use, it was relatively basic and could be bypassed in order to allow for a successful attack. Allowing for an attack to be successful on the database used on the website allowed for everything stored within the database to be leaked and could allow for the website to be completely exposed.

Introducing the use of prepared statements where any SQL queries are used would eliminate the risk of an attack and ensure no data can be extracted from the database that isn't meant to be seen (PHP, 2001). Prepared statements allow for the desired query to be already set and simply inserts the user-supplied data into the query, eliminating the possibility of extracting information. During an SQL injection attack on prepared statements, the query or logic isn't changed or altered so the maliciously injected code has no impact on the query and the attack fails to progress.

3.2.4.3 DIRECTORY TRAVERSAL

With the ability to browse directories within the site being enabled within the ".htaccess" file allowed for directories such as database, which included a copy of the database structure being used, and the picture directory which contained files upload using the changepicture file upload available from the user profile page. Allowing users to browse directories that didn't contain a landing page allowed for information that would have prevented attacks to be released.

Changing the contents of the ".htaccess" file to be "Options -Indexes" will prevent sites without an index or landing page to be browsed like they currently are. Due to the browsing ability that was available the upload PHP file used in the shell exploit was able to be viewed within the directory to confirm that the upload was successful.

3.2.4.4 LOCAL FILE INCLUSION

Directory traversal to a local file was possible and sensitive files were able to be accessed that compromised the server being used. Files not listed on the website were able to be accessed such as the passwd file for the system and allowed for the credentials to possibly be stolen.

A possible countermeasure to this is to whitelist the files that are live and that are designed to be accessed when a request that isn't whitelisted is received, no information is able to be extracted as the request is not processed and isn't able to be used maliciously.

3.2.4.5 FILE UPLOAD

Utilising the vulnerability within the file upload function on the user profile page allowed for a shell PHP file to be uploaded onto the server. The PHP file was then able to be activated and a shell was created on the server with access to the server information.

To prevent this, filtering the file upload types to only allow image types would prevent straight PHP files from being uploaded. Although this filtering was spotted within the source code suspected errors prevented it from limiting the upload during the testing. Operating a renaming scheme that provides new uploads with a randomly generated ID number would prevent malicious files from being activated as the known file name wouldn't be the same as it was when the attacker uploaded it.

Verifying file types on upload and not relying on just the extension to conduct checks would prevent files from being uploaded with extensions that don't match the true nature of the file and would prevent them from bypassing the filter.

3.2.5 LOGIC FLAWS

Flaws within the shopping functionality logic allowed for products to be purchased at a negative value by tampering with the data after adding items to the basket. This could result in the company having to pay out on real orders using this vulnerability and could be exploited to obtain products for much less than their value.

Implementing checks with the server during an important stage of the shopping checkout journey to validate the pricing of products along with the quantity to ensure negative values aren't present and that no malicious values had been entered.

Defining max and minimum values for fields can also prevent value and quantity from being maliciously tweaked as well as calculating a hash of the transaction ad using it to compare further down the checkout experience to ensure that nothing had been tweaked to the condition of the shopping basket.

3.2.6 GENERAL DISCUSSION

From the investigation carried out on the Hacklab Security Solutions website, it was quickly found that the website contained a significant number of vulnerabilities that could potentially be exploited. Exploits including disclosure of information, SQL injection, and XSS among others significantly impacted the website's ability to operate securely.

From the amount of potential information gathered on the site including sensitive information pertaining to the company, the systems and technology being used, and admin and customer private details including logins could result in a series of potential fines and penalties depending on the leaked information.

Some attempts had been made to prevent attacks but due to the implementations of these attempted countermeasures, many of them were able to be bypassed or did not perform their intended function. These attempts included the usage of an SQL filter which would have prevented certain attacks but due to the construction of the filter bypassing it would be possible.

Overall, the website in its current state should be considered to be extremely unsafe and unfit for usage due to the issues and dangers to potential customers along with the risks that operating a website of this security comes with and the implications that can come from running a site in this condition such as fines and reputational damage to the company (ITgovernance, 2018). The suggested countermeasures should be introduced at the earliest date and will ensure the website functions properly for both customers and employees and keeps both parties' information secure.

3.3 FUTURE WORK

After implementing the countermeasures and solutions to the vulnerabilities and problems discovered and disclosed during the test, the provided website would be considered safe for public usage. Before launching the newly updated website, another test would be provided to ensure the fixes had been implemented into the website correctly and to ensure no other exploits had developed as a result of fixing the already displayed issues.

Further testing could be carried out in the future should Hacklab Security Solutions choose to expand on their current active site or redesign the entirety of the website to ensure it is safe for the public domain.

REFERENCES PART 1

Interpol.int. (2021) *INTERPOL report shows alarming rate of cyberattacks during COVID-19*. [online] Available at: <https://www.interpol.int/en/News-and-Events/News/2020/INTERPOL-report-shows-alarming-rate-of-cyberattacks-during-COVID-19> [Accessed 7 December 2021].

Unctad.org. (2021) *Global e-commerce jumps to $26.7 trillion, COVID-19 boosts online sales | UNCTAD*. [online] Available at: <https://unctad.org/news/global-e-commerce-jumps-267-trillion-covid-19-boosts-online-sales> [Accessed 7 December 2021].

Patterson, D., (2021) *Cybercrime is thriving during the pandemic, driven by surge in phishing and ransomware*. [online] Cbsnews.com. Available at: <https://www.cbsnews.com/news/ransomware-phishing-cybercrime-pandemic/> [Accessed 7 December 2021].

Stuttard, D. and Pinto, M., (2011) *The web application hacker's handbook*. 2nd ed. Hoboken, N.J.: Wiley.

Kali Linux Tools (2021) Kali Linux Tool List [Online] Available at: <https://www.kali.org/tools/> [Accessed 20 November 2021].

Google Developers (2021) *Robots.txt Introduction and Guide | Google Search Central*. [online] Available at: <https://developers.google.com/search/docs/advanced/robots/intro> [Accessed 1 December 2021].

2021. *Dirb*. [online] Available at: <https://www.kali.org/tools/dirb/> [Accessed 10 December 2021].

Nmap.org. n.d. *Chapter 15. Nmap Reference Guide | Nmap Network Scanning*. [online] Available at: <https://nmap.org/book/man.html> [Accessed 10 December 2021].

Sullo, C. and Lodge, D., (2021) *Nikto2 | CIRT.net*. [Online] Available at: <https://cirt.net/Nikto2> [Accessed 30 November 2021].

Portswigger.net. *How to use Burp Suite for penetration testing*. [online] Available at: <https://portswigger.net/burp/documentation/desktop/penetration-testing> [Accessed 10 December 2021].

Nordpass.com (2021) Top 200 worst passwords [online] Available at: <https://nordpass.com/most-common-passwords-list/> [Accessed 1 December 2021].

GitHub (2021) *GitHub - vanhauser-thc/thc-hydra: hydra*. [online] Available at: <https://github.com/vanhauser-thc/thc-hydra> [Accessed 14 December 2021].

E. Shacklett, M. and Loshin, P., n.d. *What is MD5 (MD5 Message-Digest Algorithm)?*. [online] SearchSecurity. Available at: <https://www.techtarget.com/searchsecurity/definition/MD5> [Accessed 11 December 2021].

Guru99. *PHP Session & PHP Cookies with Example*. [online] Available at: <https://www.guru99.com/cookies-and-sessions.html> [Accessed 12 December 2021].

Academy, W., *What is CSRF (Cross-site request forgery)? Tutorial & Examples | Web Security Academy*. [online] Portswigger.net. Available at: <https://portswigger.net/web-security/csrf> [Accessed 5 December 2021].

Sqlmap.org. 2021. *sqlmap: automatic SQL injection and database takeover tool*. [online] Available at: <https://sqlmap.org/> [Accessed 6 December 2021].

Owasp.org. (2021) *Cross Site Scripting (XSS) Software Attack | OWASP Foundation*. [online] Available at: <https://owasp.org/www-community/attacks/xss/> [Accessed 4 December 2021].

Academy, W., *What is directory traversal, and how to prevent it? | Web Security Academy*. [online] Portswigger.net. Available at: <https://portswigger.net/web-security/file-path-traversal> [Accessed 3 December 2021].

Prodromou, A., (2020) Web Shells in Action. [Blog] *Web Security Zone*, Available at: <https://www.acunetix.com/blog/articles/web-shells-action-introduction-web-shells-part-4/> [Accessed 6 December 2021].

VAADATA Le Blog, (2015) What are Business Logic Flaws on Web Applications?. Available at: <https://www.vaadata.com/blog/what-are-business-logic-flaws-on-web-applications/> [Accessed 8 December 2021].

REFERENCES PART 2

Misja.com (no date) *Epoch Converter*. [online] Available at: <https://www.epochconverter.com/> [Accessed 12 January 2022].

Laverty, P., (2017) What Is User Enumeration?. [Blog] Available at: <https://www.rapid7.com/blog/post/2017/06/15/about-user-enumeration/> [Accessed 13 January 2022].

Portswigger.net. (no date) *What is cross-site scripting (XSS) and how to prevent it? | Web Security Academy*. [online] Available at: <https://portswigger.net/web-security/cross-site-scripting> [Accessed 12 January 2022].

Portswigger.net. (no date) *SQL Injection: Bypassing Common Filters*. [online] Available at: <https://portswigger.net/support/sql-injection-bypassing-common-filters> [Accessed 12 January 2022].

Php.net. (2001). *PHP: Sanitize filters - Manual*. [online] Available at: <https://www.php.net/manual/en/filter.filters.sanitize.php> [Accessed 14 January 2022].

Docs.oracle.com. (2004). *Apache Tutorial: .htaccess files - Apache HTTP Server*. [online] Available at: <https://docs.oracle.com/cd/B14099_19/web.1012/q20206/howto/htaccess.html> [Accessed 13 January 2022].

Php.net. 2001. *PHP: setcookie - Manual*. [online] Available at: <https://www.php.net/manual/en/function.setcookie.php> [Accessed 13 January 2022].

Google Developers. 2021. *Robots.txt Introduction and Guide | Google Search Central | Google Developers*. [online] Available at: <https://developers.google.com/search/docs/advanced/robots/intro#:~:text=txt-,A%20robots.,or%20password%2Dprotect%20the%20page.> [Accessed 13 January 2022].

Cloudflare (no date) *Why is HTTP not secure? | HTTP vs. HTTPS*. [online] Available at: <https://www.cloudflare.com/en-gb/learning/ssl/why-is-http-not-secure/> [Accessed 13 January 2022].

Developer.mozilla.org. 2021. *Mixed content - Web security | MDN*. [online] Available at: <https://developer.mozilla.org/en-US/docs/Web/Security/Mixed_content> [Accessed 13 January 2022].

Owasp.org. (no date) *WSTG - Latest | OWASP Foundation*. [online] Available at: <https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/03-Identity_Management_Testing/04-Testing_for_Account_Enumeration_and_Guessable_User_Account> [Accessed 13 January 2022].

Avast (2018) How to create a strong password. [Blog] Available at: <https://blog.avast.com/strong-password-ideas> [Accessed 14 January 2022].

Ncsc.gov.uk. 2018. *Password policy: updating your approach*. [online] Available at: <https://www.ncsc.gov.uk/collection/passwords/updating-your-approach> [Accessed 14 January 2022].

Authy. (no date) *What Is Two-Factor Authentication (2FA)? - Authy*. [online] Available at: <https://authy.com/what-is-2fa/> [Accessed 14 January 2022].

Academy, W., (no date) *CSRF tokens | Web Security Academy*. [online] Portswigger.net. Available at: <https://portswigger.net/web-security/csrf/tokens> [Accessed 14 January 2022].

Php.net. (2001) *PHP: htmlentities - Manual*. [online] Available at: <https://www.php.net/manual/en/function.htmlentities.php> [Accessed 15 January 2022].

Php.net. (2001) *PHP: Prepared Statements - Manual*. [online] Available at: <https://www.php.net/manual/en/mysqli.quickstart.prepared-statements.php> [Accessed 15 January 2022].

Itgovernance.co.uk. (2018) *GDPR Penalties and Fines | What's the Maximum Fine?*. [online] Available at: <https://www.itgovernance.co.uk/dpa-and-gdpr-penalties> [Accessed 15 January 2022].

APPENDICES PART 1

APPENDIX A – ZAP URL RESULTS

http://192.168.1.20/
http://192.168.1.20/Email.php
http://192.168.1.20/aboutus.php
http://192.168.1.20/admin
http://192.168.1.20/admin/ADMIN
http://192.168.1.20/admin/ADMIN/
http://192.168.1.20/admin/ADMIN/SERVER
http://192.168.1.20/admin/ADMIN/SERVER/ADS
http://192.168.1.20/admin/ADMIN/SERVER/ADS/upload
http://192.168.1.20/admin/ADMIN/SERVER/ADS/upload/
http://192.168.1.20/admin/ADMIN/SERVER/ADS/upload/rick.jpg
http://192.168.1.20/admin/ADMIN/SERVER/AS
http://192.168.1.20/admin/ADMIN/SERVER/AS/products
http://192.168.1.20/admin/ADMIN/SERVER/AS/products/
http://192.168.1.20/admin/ADMIN/SERVER/AS/products/1.JPG
http://192.168.1.20/admin/ADMIN/SERVER/AS/products/2.JPG
http://192.168.1.20/admin/ADMIN/SERVER/AS/products/3.JPG
http://192.168.1.20/admin/ADMIN/SERVER/AS/products/4.JPG
http://192.168.1.20/assets
http://192.168.1.20/assets/
http://192.168.1.20/assets/bootstrap.min.css
http://192.168.1.20/assets/bootstrap.min.js
http://192.168.1.20/assets/css
http://192.168.1.20/assets/css/
http://192.168.1.20/assets/css/bootstrap-responsive.css
http://192.168.1.20/assets/css/bootstrap.css
http://192.168.1.20/assets/css/docs.css
http://192.168.1.20/assets/img
http://192.168.1.20/assets/img/
http://192.168.1.20/assets/img/glyphicons-halflings-white.png
http://192.168.1.20/assets/img/glyphicons-halflings.png
http://192.168.1.20/assets/img/search.png
http://192.168.1.20/assets/jquery.min.js
http://192.168.1.20/assets/js
http://192.168.1.20/assets/js/
http://192.168.1.20/assets/js/application.js
http://192.168.1.20/assets/js/bootsshoptgl.js
http://192.168.1.20/assets/js/bootstrap-affix.js
http://192.168.1.20/assets/js/bootstrap-alert.js
http://192.168.1.20/assets/js/bootstrap-button.js
http://192.168.1.20/assets/js/bootstrap-carousel.js
http://192.168.1.20/assets/js/bootstrap-collapse.js

http://192.168.1.20/assets/js/bootstrap-dropdown.js
http://192.168.1.20/assets/js/bootstrap-modal.js
http://192.168.1.20/assets/js/bootstrap-popover.js
http://192.168.1.20/assets/js/bootstrap-scrollspy.js
http://192.168.1.20/assets/js/bootstrap-tab.js
http://192.168.1.20/assets/js/bootstrap-tooltip.js
http://192.168.1.20/assets/js/bootstrap-transition.js
http://192.168.1.20/assets/js/bootstrap-typeahead.js
http://192.168.1.20/assets/js/google-code-prettify
http://192.168.1.20/assets/js/google-code-prettify/
http://192.168.1.20/assets/js/google-code-prettify/prettify.css
http://192.168.1.20/assets/js/google-code-prettify/prettify.js
http://192.168.1.20/assets/js/jquery.js
http://192.168.1.20/assets/js/jquery.lightbox-0.5.js
http://192.168.1.20/assets/style.css
http://192.168.1.20/bootstrap
http://192.168.1.20/bootstrap.min.js
http://192.168.1.20/bootstrap/
http://192.168.1.20/bootstrap/css
http://192.168.1.20/bootstrap/css/
http://192.168.1.20/bootstrap/css/bootstrap.min.css
http://192.168.1.20/bootstrap/fonts
http://192.168.1.20/bootstrap/fonts/
http://192.168.1.20/bootstrap/fonts/glyphicons-halflings-regular.woff
http://192.168.1.20/bootstrap/fonts/glyphicons-halflings-regular.woff2
http://192.168.1.20/contact.php
http://192.168.1.20/docs.min.js
http://192.168.1.20/img
http://192.168.1.20/img/
http://192.168.1.20/img/5.jpg
http://192.168.1.20/img/CCTV.jpg
http://192.168.1.20/img/Hacklab.jpg
http://192.168.1.20/img/aa.jpg
http://192.168.1.20/img/aalogo.jpg
http://192.168.1.20/index.php
http://192.168.1.20/jquery.min.js
http://192.168.1.20/js
http://192.168.1.20/js/incrementing.js
http://192.168.1.20/pictures
http://192.168.1.20/pictures/
http://192.168.1.20/pictures/rick.jpg
http://192.168.1.20/product_details.php?%20id=2
http://192.168.1.20/product_details.php?=
http://192.168.1.20/products.php
http://192.168.1.20/style.css
http://192.168.1.20/updatepassword.php
http://192.168.1.20/user_aboutus.php
http://192.168.1.20/user_account2.php

http://192.168.1.20/user_index.php
http://192.168.1.20/user_mail.php
http://192.168.1.20/user_order.php
http://192.168.1.20/user_product_details.php?=
http://192.168.1.20/user_product_details.php?id=2
http://192.168.1.20/user_products.php

APPENDIX B – DIRBUSTER SCAN FULL RESULT

DirBuster 1.0-RC1 - Report

http://www.owasp.org/index.php/Category:OWASP_DirBuster_Project

Report produced on Wed Nov 03 16:48:45 EDT 2021

---

http://192.168.1.20:80

Directories found during testing:

Dirs found with a 200 response:

/img/

/

/icons/

/admin/

/assets/

/assets/img/

/pictures/

/assets/js/

/assets/js/google-code-prettify/

/assets/bootstrap/

/assets/css/

/admin/assets/

/admin/assets/img/

/assets/bootstrap/css/

/assets/bootstrap/fonts/

/assets/bootstrap/js/

/admin/assets/css/

/admin/assets/js/

/database/

/icons/small/

/admin/assets/css/locales/

/admin/assets/js/google-code-prettify/

/include/

/admin/include/

/admin/ADMIN/

/admin/ADMIN/SERVER/AS/

/admin/ADMIN/SERVER/AS/products/

/admin/ADMIN/ADS/

/admin/ADMIN/AS/

/admin/ADMIN/SERVER/ADS/

/admin/ADMIN/OOS/

/admin/ADMIN/SERVER/OOS/

/admin/ADMIN/SERVER/ADS/js/

/admin/ADMIN/SERVER/AS/js/

/admin/ADMIN/SERVER/js/

/admin/ADMIN/AS/js/

/admin/ADMIN/ADS/js/

/admin/ADMIN/SERVER/OOS/js/

/admin/ADMIN/OOS/js/

/admin/ADMIN/SERVER/ADS/js/locales/

/admin/ADMIN/SERVER/AS/js/locales/

/admin/ADMIN/SERVER/js/locales/

/admin/ADMIN/ADS/js/locales/

/admin/ADMIN/AS/js/locales/

/admin/ADMIN/SERVER/OOS/js/locales/

/admin/ADMIN/OOS/js/locales/

/admin/ADMIN/SERVER/ADS/upload/

/admin/ADMIN/SERVER/AS/css/

/admin/ADMIN/SERVER/css/

/admin/ADMIN/AS/css/

/admin/ADMIN/OOS/css/

/admin/ADMIN/SERVER/AS/css/locales/

/admin/ADMIN/SERVER/OOS/css/

/admin/ADMIN/SERVER/css/locales/

/admin/ADMIN/ADS/css/

/admin/ADMIN/SERVER/ADS/css/

/admin/ADMIN/AS/css/locales/

/admin/ADMIN/OOS/css/locales/

/admin/ADMIN/SERVER/OOS/css/locales/

/admin/ADMIN/ADS/css/locales/

/admin/ADMIN/SERVER/ADS/css/locales/

/bootstrap/

/bootstrap/bootstrap/

/bootstrap/css/

/bootstrap/fonts/

/bootstrap/js/

/bootstrap/bootstrap/css/

/bootstrap/bootstrap/js/

/bootstrap/js/vendor/

/bootstrap/bootstrap/js/google-code-prettify/

Dirs found with a 403 response:

/cgi-bin/

/error/

/error/include/

/phpmyadmin/

Dirs found with a 302 response:

/admin/ADMIN/SERVER/

Files found during testing:

Files found with a 200 responce:

/privacy.php

/contact.php

/login.php

/products.php

/register.php

/index.php

/terms.php

/aboutus.php

/header.php

/map.php

/mail.php

/faqs.php

/info.php

/footer.php

/admin/index.php

/maps.php

/admin/header.php

/docs.min.js

/jquery.min.js

/forgotpass.php

/bootstrap.min.js

/affix.php

/assets/js/jquery.js

/assets/js/google-code-prettify/prettify.js

/assets/bootstrap.min.css

/admin/footer.php

/assets/js/application.js

/assets/bootstrap.min.js

/assets/js/bootstrap-transition.js

/assets/js/bootstrap-modal.js

/assets/js/bootstrap-scrollspy.js

/assets/jquery.min.js

/assets/js/bootstrap-alert.js

/assets/offcanvas.css

/assets/js/bootstrap-dropdown.js

/assets/style.css

/assets/js/bootstrap-tab.js

/assets/js/bootstrap-tooltip.js

/assets/js/google-code-prettify/prettify.css

/assets/js/bootstrap-button.js

/assets/js/bootstrap-collapse.js

/assets/js/bootsshoptgl.js

/assets/js/bootstrap-carousel.js

/assets/js/bootstrap-typeahead.js

/assets/js/bootstrap-popover.js

/assets/js/bootstrap-affix.js

/assets/js/bootstrap.js

/assets/js/jquery.lightbox-0.5.js

/assets/css/bootstrap-responsive.css

/assets/js/bootstrap.min.js

/assets/js/bootstrap.min.tmp.js

/assets/css/bootstrap-theme.min.css

/assets/js/docs.min.js

/assets/css/bootstrap.css

/assets/js/ie-emulation-modes-warning.js

/assets/js/ie10-viewport-bug-workaround.js

/assets/css/bootstrap.min.css

/admin/assets/style.css

/assets/css/carousel.css

/assets/js/jquery.min.js

/assets/css/docs.css

/assets/js/jquery.ui.custom.js

/assets/bootstrap/css/bootstrap-theme.css

/assets/js/scg.js

/assets/css/docs.min.css

/assets/css/font-awesome.min.css

/assets/bootstrap/css/bootstrap-theme.css.map

/assets/bootstrap/fonts/glyphicons-halflings-regular.eot

/assets/bootstrap/js/bootstrap.js

/assets/bootstrap/css/bootstrap-theme.min.css

/assets/bootstrap/js/bootstrap.min.js

/assets/bootstrap/fonts/glyphicons-halflings-regular.svg

/assets/bootstrap/js/npm.js

/assets/bootstrap/fonts/glyphicons-halflings-regular.ttf

/assets/bootstrap/css/bootstrap.css

/assets/bootstrap/fonts/glyphicons-halflings-regular.woff

/assets/bootstrap/fonts/glyphicons-halflings-regular.woff2

/assets/bootstrap/css/bootstrap.min.css

/assets/bootstrap/css/bootstrap.css.map

/admin/assets/css/boots.min.css

/admin/assets/css/bootstrap-datetimepicker.js

/announce.php

/admin/assets/js/application.js

/admin/assets/css/bootstrap-datetimepicker.min.css

/admin/assets/js/bootsshoptgl.js

/admin/assets/css/bootstrap-responsive.css

/admin/assets/js/bootstrap-affix.js

/admin/assets/css/bootstrap-theme.css.map

/database/aa2000.sql

/admin/assets/js/bootstrap-alert.js

/admin/assets/css/bootstrap-theme.min.css

/admin/assets/js/bootstrap-button.js

/query.php

/admin/assets/css/bootstrap.css

/admin/assets/js/bootstrap-carousel.js

/admin/assets/js/bootstrap-collapse.js

/admin/assets/css/bootstrap.css.map

/admin/assets/css/bootstrap.min.css

/admin/assets/css/bootstrap2.css

/admin/assets/js/bootstrap-dropdown.js

/admin/assets/css/bootstrap2.min.css

/admin/assets/js/bootstrap-modal.js

/admin/assets/css/docs.css

/admin/assets/js/bootstrap-popover.js

/admin/assets/css/font-awesome.css

/admin/assets/js/bootstrap-scrollspy.js

/admin/assets/css/justified-nav.css

/admin/assets/js/bootstrap-tab.js

/admin/assets/css/offcanvas.css

/admin/assets/js/bootstrap-tooltip.js

/admin/assets/css/style.css

/admin/assets/js/bootstrap-transition.js

/admin/assets/js/bootstrap-typeahead.js

/admin/assets/js/bootstrap.js

/admin/assets/js/bootstrap.min.js

/admin/assets/js/bootstrap.min.tmp.js

/admin/assets/js/jquery.js

/admin/assets/js/jquery.lightbox-0.5.js

/admin/assets/js/jquery.ui.custom.js

/admin/assets/js/google-code-prettify/prettify.css

/admin/assets/js/google-code-prettify/prettify.js

/include/connect.php

/function.php

/admin/include/connect.php

/cookie.php

/header2.php

/username.php

/instructions.php

/product_details.php

/admin/ADMIN/SERVER/AS/index.php

/admin/ADMIN/ADS/index.php

/admin/ADMIN/SERVER/ADS/index.php

/admin/ADMIN/AS/index.php

/admin/ADMIN/OOS/index.php

/admin/ADMIN/SERVER/OOS/index.php

/admin/ADMIN/SERVER/AS/asset.php

/admin/ADMIN/SERVER/user.php

/admin/ADMIN/SERVER/Recovery.php

/admin/ADMIN/SERVER/AS/reports.php

/admin/ADMIN/SERVER/customer_archive.php

/admin/ADMIN/SERVER/AS/reports1.php

/admin/ADMIN/SERVER/Customer_loginout.php

/admin/ADMIN/SERVER/AS/products/1.JPG

/admin/ADMIN/SERVER/AS/equipment.php

/admin/ADMIN/SERVER/AS/products/2.JPG

/admin/ADMIN/ADS/Customers.php

/admin/ADMIN/SERVER/AS/products/3.JPG

/admin/ADMIN/AS/asset.php

/admin/ADMIN/SERVER/AS/configuration.php

/admin/ADMIN/ADS/Customer_list.php

/admin/ADMIN/SERVER/AS/products/4.JPG

/admin/ADMIN/SERVER/ADS/Customer_list.php

/admin/ADMIN/ADS/announcement.php

/admin/ADMIN/SERVER/AS/fixedasset_report.php

/admin/ADMIN/OOS/orders.php

/admin/ADMIN/AS/reports.php

/admin/ADMIN/SERVER/OOS/orders.php

/admin/ADMIN/SERVER/ADS/announcement.php

/admin/ADMIN/ADS/messages.php

/admin/ADMIN/SERVER/AS/header.php

/admin/ADMIN/SERVER/ADS/messages.php

/admin/ADMIN/SERVER/AS/products/5.JPG

/admin/ADMIN/SERVER/OOS/reports.php

/admin/ADMIN/OOS/view_order_notif.php

/admin/ADMIN/OOS/reports.php

/admin/ADMIN/AS/reports1.php

/admin/ADMIN/SERVER/OOS/view_order_notif.php

/admin/ADMIN/SERVER/user_type.php

/admin/ADMIN/SERVER/AS/products/6.JPG

/admin/ADMIN/SERVER/ADS/js/jquery.min.js

/admin/ADMIN/OOS/jquery-1.10.2.js

/admin/ADMIN/SERVER/AS/print_products.php

/admin/ADMIN/SERVER/ADS/js/bootstrap.min.js

/admin/ADMIN/SERVER/AS/products/7.JPG

/admin/ADMIN/SERVER/asset_archive.php

/admin/ADMIN/SERVER/admin_report.php

/admin/ADMIN/SERVER/AS/products/8.JPG

/admin/ADMIN/SERVER/Administrator_loginout.php

/admin/ADMIN/SERVER/backup.php

/admin/ADMIN/SERVER/AS/add_new_equipment.php

/admin/ADMIN/SERVER/AS/products/9.JPG

/admin/ADMIN/SERVER/AS/js/DT_bootstrap.js

/admin/ADMIN/SERVER/AS/js/jquery.dataTables.js

/admin/ADMIN/SERVER/Audit_trail.php

/admin/ADMIN/SERVER/js/jquery.dataTables.js

/admin/ADMIN/SERVER/print_database.php

/admin/ADMIN/SERVER/AS/products/10.JPG

/admin/ADMIN/SERVER/js/DT_bootstrap.js

/admin/ADMIN/SERVER/AS/js/bootstrap.js

/admin/ADMIN/SERVER/AS/js/jquery-1.7.2.min.js

/admin/ADMIN/SERVER/OOS/pending_order.php

/admin/ADMIN/SERVER/AS/add_new_category.php

/admin/ADMIN/SERVER/js/bootstrap.js

/admin/ADMIN/OOS/pending_order.php

/admin/ADMIN/AS/print_products.php

/admin/ADMIN/SERVER/AS/products/11.JPG

/admin/ADMIN/SERVER/OOS/confirmed_order.php

/admin/ADMIN/OOS/confirmed_order.php

/admin/ADMIN/SERVER/js/jquery-1.7.2.min.js

/admin/ADMIN/SERVER/print_customer_monitoring.php

/admin/ADMIN/SERVER/ADS/js/DT_bootstrap.js

/admin/ADMIN/SERVER/print_customer_archive.php

/admin/ADMIN/SERVER/ADS/messages_box.php

/admin/ADMIN/SERVER/OOS/print_orders.php

/admin/ADMIN/SERVER/ADS/js/jquery.dataTables.js

/admin/ADMIN/SERVER/AS/jquery.min.js

/admin/ADMIN/ADS/messages_box.php

/admin/ADMIN/SERVER/ADS/header.php

/admin/ADMIN/OOS/print_orders.php

/admin/ADMIN/SERVER/AS/print_products1.php

/admin/ADMIN/SERVER/ADS/js/bootstrap.js

/admin/ADMIN/SERVER/AS/js/bootstrap.min.js

/admin/ADMIN/AS/js/jquery.dataTables.js

/admin/ADMIN/AS/js/DT_bootstrap.js

/admin/ADMIN/SERVER/AS/print_assetlist.php

/admin/ADMIN/SERVER/ADS/js/bootstrap-datetimepicker.js

/admin/ADMIN/AS/header.php

/admin/ADMIN/SERVER/ADS/js/jquery-1.7.2.min.js

/admin/ADMIN/AS/js/bootstrap.js

/admin/ADMIN/SERVER/print_asset_archive.php

/admin/ADMIN/ADS/js/jquery.dataTables.js

/admin/ADMIN/SERVER/print_admin_report.php

/admin/ADMIN/SERVER/header.php

/admin/ADMIN/SERVER/ADS/print_Customerlist.php

/admin/ADMIN/ADS/js/DT_bootstrap.js

/admin/ADMIN/OOS/js/jquery.dataTables.js

/admin/ADMIN/SERVER/OOS/js/bootstrap.js

/admin/ADMIN/SERVER/OOS/js/jquery.dataTables.js

/admin/ADMIN/ADS/js/bootstrap.js

/admin/ADMIN/SERVER/js/bootstrap-datetimepicker.js

/admin/ADMIN/AS/jquery.min.js

/admin/ADMIN/SERVER/AS/js/bootstrap-datetimepicker.js

/admin/ADMIN/AS/js/jquery-1.7.2.min.js

/admin/ADMIN/SERVER/OOS/js/jquery-1.7.2.min.js

/admin/ADMIN/AS/print_products1.php

/admin/ADMIN/SERVER/AS/js/jquery.min.js

/admin/ADMIN/AS/js/bootstrap.min.js

/admin/ADMIN/SERVER/js/jquery.min.js

/admin/ADMIN/ADS/js/jquery-1.7.2.min.js

/admin/ADMIN/OOS/js/bootstrap.js

/admin/ADMIN/ADS/print_Customerlist.php

/admin/ADMIN/SERVER/print_all_changes.php

/admin/ADMIN/OOS/js/jquery-1.7.2.min.js

/admin/ADMIN/SERVER/js/locales/bootstrap-datetimepicker.fr.js

/admin/ADMIN/AS/js/bootstrap-datetimepicker.js

/admin/ADMIN/ADS/js/bootstrap-datetimepicker.js

/admin/ADMIN/SERVER/OOS/js/DT_bootstrap.js

/admin/ADMIN/OOS/js/DT_bootstrap.js

/admin/ADMIN/SERVER/OOS/js/bootstrap-datetimepicker.js

/admin/ADMIN/OOS/js/bootstrap-datetimepicker.js

/admin/ADMIN/SERVER/OOS/js/bootstrap.min.js

/admin/ADMIN/SERVER/AS/footer.php

/admin/ADMIN/SERVER/OOS/js/jquery.min.js

/admin/ADMIN/OOS/js/jquery.min.js

/admin/ADMIN/OOS/header.php

/admin/ADMIN/ADS/header.php

/admin/ADMIN/SERVER/OOS/header.php

/admin/ADMIN/AS/footer.php

/admin/ADMIN/SERVER/ADS/footer.php

/admin/ADMIN/SERVER/footer.php

/admin/ADMIN/SERVER/account.php

/admin/ADMIN/OOS/footer.php

/admin/ADMIN/ADS/footer.php

/admin/ADMIN/SERVER/OOS/footer.php

/admin/ADMIN/SERVER/ADS/upload/4.JPG

/admin/ADMIN/SERVER/ADS/upload/5.JPG

/admin/ADMIN/SERVER/AS/css/boots.min.css

/admin/ADMIN/SERVER/css/boots.min.css

/admin/ADMIN/SERVER/AS/css/bootstrap-datetimepicker.js

/admin/ADMIN/SERVER/css/bootstrap-datetimepicker.js

/admin/ADMIN/SERVER/AS/css/bootstrap-datetimepicker.min.css

/admin/ADMIN/SERVER/AS/css/bootstrap-theme.css.map

/admin/ADMIN/SERVER/css/bootstrap-datetimepicker.min.css

/admin/ADMIN/SERVER/AS/css/bootstrap-theme.min.css

/admin/ADMIN/SERVER/css/bootstrap-theme.css.map

/admin/ADMIN/SERVER/css/bootstrap-theme.min.css

/admin/ADMIN/SERVER/AS/css/bootstrap.css

/admin/ADMIN/SERVER/css/bootstrap.css

/admin/ADMIN/SERVER/AS/css/bootstrap.css.map

/admin/ADMIN/SERVER/AS/css/bootstrap.min.css

/admin/ADMIN/SERVER/css/bootstrap.css.map

/admin/ADMIN/AS/css/boots.min.css

/admin/ADMIN/SERVER/css/bootstrap.min.css

/admin/ADMIN/SERVER/AS/css/bootstrap2.css

/admin/ADMIN/SERVER/css/bootstrap2.css

/admin/ADMIN/AS/css/bootstrap-datetimepicker.js

/admin/ADMIN/SERVER/AS/css/bootstrap2.min.css

/admin/ADMIN/AS/css/bootstrap-datetimepicker.min.css

/admin/ADMIN/SERVER/css/bootstrap2.min.css

/admin/ADMIN/SERVER/AS/css/font-awesome.css

/admin/ADMIN/OOS/css/boots.min.css

/admin/ADMIN/SERVER/css/font-awesome.css

/admin/ADMIN/AS/css/bootstrap-theme.css.map

/admin/ADMIN/OOS/css/bootstrap-datetimepicker.js

/admin/ADMIN/SERVER/AS/css/justified-nav.css

/admin/ADMIN/SERVER/css/justified-nav.css

/admin/ADMIN/OOS/css/bootstrap-datetimepicker.min.css

/admin/ADMIN/AS/css/bootstrap-theme.min.css

/admin/ADMIN/SERVER/AS/css/style.css

/admin/ADMIN/OOS/css/bootstrap-theme.css.map

/admin/ADMIN/AS/css/bootstrap.css

/admin/ADMIN/SERVER/css/style.css

/admin/ADMIN/OOS/css/bootstrap-theme.min.css

/admin/ADMIN/AS/css/bootstrap.css.map

/admin/ADMIN/AS/css/bootstrap.min.css

/admin/ADMIN/OOS/css/bootstrap.css

/admin/ADMIN/AS/css/bootstrap2.css

/admin/ADMIN/OOS/css/bootstrap.css.map

/admin/ADMIN/AS/css/bootstrap2.min.css

/admin/ADMIN/OOS/css/bootstrap.min.css

/admin/ADMIN/AS/css/font-awesome.css

/admin/ADMIN/OOS/css/bootstrap2.css

/admin/ADMIN/AS/css/justified-nav.css

/admin/ADMIN/OOS/css/bootstrap2.min.css

/admin/ADMIN/OOS/css/font-awesome.css

/admin/ADMIN/OOS/css/justified-nav.css

/admin/ADMIN/AS/css/style.css

/admin/ADMIN/SERVER/OOS/css/boots.min.css

/admin/ADMIN/SERVER/OOS/css/bootstrap-datetimepicker.js

/admin/ADMIN/OOS/css/style.css

/admin/ADMIN/SERVER/OOS/css/bootstrap-datetimepicker.min.css

/admin/ADMIN/SERVER/OOS/css/bootstrap-theme.css.map

/admin/ADMIN/SERVER/OOS/css/bootstrap-theme.min.css

/admin/ADMIN/ADS/css/boots.min.css

/admin/ADMIN/ADS/css/bootstrap-datetimepicker.js

/admin/ADMIN/SERVER/OOS/css/bootstrap.css

/admin/ADMIN/ADS/css/bootstrap-datetimepicker.min.css

/admin/ADMIN/SERVER/ADS/css/boots.min.css

/admin/ADMIN/ADS/css/bootstrap-theme.css.map

/admin/ADMIN/SERVER/OOS/css/bootstrap.css.map

/admin/ADMIN/ADS/css/bootstrap-theme.min.css

/admin/ADMIN/SERVER/ADS/css/bootstrap-datetimepicker.js

/admin/ADMIN/SERVER/OOS/css/bootstrap.min.css

/admin/ADMIN/ADS/css/bootstrap.css

/admin/ADMIN/SERVER/ADS/css/bootstrap-datetimepicker.min.css

/admin/ADMIN/SERVER/OOS/css/bootstrap2.css

/admin/ADMIN/SERVER/ADS/css/bootstrap-theme.css.map

/admin/ADMIN/SERVER/OOS/css/bootstrap2.min.css

/admin/ADMIN/ADS/css/bootstrap.css.map

/admin/ADMIN/SERVER/ADS/css/bootstrap-theme.min.css

/admin/ADMIN/ADS/css/bootstrap.min.css

/admin/ADMIN/SERVER/ADS/css/bootstrap.css

/admin/ADMIN/SERVER/OOS/css/font-awesome.css

/admin/ADMIN/SERVER/ADS/css/bootstrap.css.map

/admin/ADMIN/SERVER/ADS/css/bootstrap.min.css

/admin/ADMIN/SERVER/OOS/css/justified-nav.css

/admin/ADMIN/ADS/css/bootstrap2.css

/admin/ADMIN/SERVER/ADS/css/bootstrap2.css

/admin/ADMIN/ADS/css/bootstrap2.min.css

/admin/ADMIN/SERVER/ADS/css/bootstrap2.min.css

/admin/ADMIN/SERVER/OOS/css/style.css

/admin/ADMIN/ADS/css/font-awesome.css

/admin/ADMIN/ADS/css/justified-nav.css

/admin/ADMIN/SERVER/ADS/css/font-awesome.css

/admin/ADMIN/SERVER/ADS/css/justified-nav.css

/admin/ADMIN/ADS/css/style.css

/admin/ADMIN/SERVER/ADS/css/style.css

/admin/ADMIN/SERVER/ADS/query.php

/hidden.php

/admin/ADMIN/ADS/query.php

/admin/ADMIN/SERVER/AS/function.php

/admin/ADMIN/AS/function.php

/admin/ADMIN/SERVER/ADS/function.php

/admin/ADMIN/SERVER/function.php

/admin/ADMIN/OOS/function.php

/admin/ADMIN/ADS/function.php

/admin/ADMIN/SERVER/OOS/function.php

/admin/ADMIN/OOS/sample.php

/admin/ADMIN/SERVER/OOS/sample.php

/admin/ADMIN/AS/equipment.php

/admin/ADMIN/AS/add_new_equipment.php

/admin/ADMIN/AS/fixedasset_report.php

/admin/ADMIN/AS/configuration.php

/admin/ADMIN/AS/print_assetlist.php

/admin/ADMIN/AS/add_new_category.php

/footer2.php

/admin/ADMIN/SERVER/AS/view_product.php

/admin/ADMIN/AS/view_product.php

/topheader.php

/bootstrap/carousel.css

/bootstrap/cover.css

/bootstrap/style.css

/bootstrap/theme.css

/bootstrap/css/boots.min.css

/bootstrap/fonts/glyphicons-halflings-regular.eot

/bootstrap/css/bootstrap-theme.css

/bootstrap/fonts/glyphicons-halflings-regular.svg

/bootstrap/js/application.js

/bootstrap/fonts/glyphicons-halflings-regular.ttf

/bootstrap/css/bootstrap-theme.css.map

/bootstrap/css/bootstrap-theme.min.css

/bootstrap/fonts/glyphicons-halflings-regular.woff

/bootstrap/js/bootstrap.js

/bootstrap/bootstrap/css/bootstrap-responsive.css

/bootstrap/js/bootstrap.min.js

/bootstrap/css/bootstrap.css

/bootstrap/bootstrap/css/bootstrap.css

/bootstrap/css/bootstrap.css.map

/bootstrap/css/bootstrap.min.css

/bootstrap/js/customizer.js

/bootstrap/bootstrap/css/bootstrap.min.css

/bootstrap/js/customize.min.js

/bootstrap/js/docs.min.js

/bootstrap/bootstrap/js/application.js

/bootstrap/js/ie8-responsive-file-warning.js

/bootstrap/bootstrap/css/docs.css

/bootstrap/bootstrap/js/bootsshoptgl.js

/bootstrap/css/bootstrap2.css

/bootstrap/js/raw-files.min.js

/bootstrap/bootstrap/js/bootstrap-affix.js

/bootstrap/css/font-awesome.css

/bootstrap/css/justified-nav.css

/bootstrap/bootstrap/js/bootstrap-alert.js

/bootstrap/bootstrap/js/bootstrap-button.js

/bootstrap/css/style.css

/bootstrap/bootstrap/js/bootstrap-carousel.js

/bootstrap/bootstrap/js/bootstrap-collapse.js

/bootstrap/bootstrap/js/bootstrap-dropdown.js

/bootstrap/bootstrap/js/bootstrap-modal.js

/bootstrap/bootstrap/js/bootstrap-popover.js

/bootstrap/js/vendor/blob.js

/bootstrap/bootstrap/js/bootstrap-scrollspy.js

/bootstrap/bootstrap/js/bootstrap-tab.js

/bootstrap/js/vendor/filesaver.js

/bootstrap/bootstrap/js/bootstrap-tooltip.js

/bootstrap/js/vendor/holder.js

/bootstrap/bootstrap/js/bootstrap-transition.js

/bootstrap/js/vendor/jszip.min.js

/bootstrap/bootstrap/js/bootstrap-typeahead.js

/bootstrap/bootstrap/js/bootstrap.js

/bootstrap/js/vendor/less.min.js

/bootstrap/bootstrap/js/bootstrap.min.js

/bootstrap/js/vendor/uglify.min.js

/bootstrap/bootstrap/js/bootstrap.min.tmp.js

/bootstrap/bootstrap/js/jquery.js

/bootstrap/bootstrap/js/jquery.lightbox-0.5.js

/bootstrap/bootstrap/js/jquery.min.js

/bootstrap/bootstrap/js/jquery.ui.custom.js

/bootstrap/bootstrap/js/google-code-prettify/prettify.css

/bootstrap/bootstrap/js/google-code-prettify/prettify.js

/admin/ADMIN/SERVER/account_edit.php

/phpinfo.php

Files found with a 302 responce:

/user_index.php

/logout.php

/user_products.php

/user_contact.php

/user_aboutus.php

/user_order.php

/Email.php

/user_account2.php

/product_summary.php

/updatepassword.php

/user_mail.php

/include/session.php

/admin/include/session.php

/admin/logout.php

/admin/ADMIN/SERVER/index.php

/admin/ADMIN/SERVER/ADS/Customers.php

/admin/ADMIN/SERVER/AS/add_new_products.php

/admin/ADMIN/SERVER/AS/delete_product.php

/admin/ADMIN/ADS/Delete_Customer.php

/admin/ADMIN/SERVER/ADS/Delete_Customer.php

/admin/ADMIN/AS/add_new_products.php

/admin/ADMIN/ADS/add_new_announcement.php

/admin/ADMIN/AS/delete_product.php

/admin/ADMIN/SERVER/ADS/add_new_announcement.php

/admin/ADMIN/SERVER/print_administrator_monitoring.php

/admin/ADMIN/SERVER/delete_customer_archive.php

/admin/ADMIN/SERVER/ADS/reply.php

/admin/ADMIN/ADS/reply.php

/admin/ADMIN/SERVER/AS/Archive.php

/admin/ADMIN/AS/Archive.php

/admin/ADMIN/SERVER/ADS/Archive.php

/admin/ADMIN/ADS/Archive.php

/admin/ADMIN/SERVER/sessions.php

APPENDIX C – DIRB SCAN FULL RESULT

DIRB v2.22

By The Dark Raver

OUTPUT_FILE: /root/Desktop/Dirb_OutputCommon

START_TIME: Sat Nov 20 18:33:22 2021

URL_BASE: http://192.168.1.20/

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

OPTION: Not Stopping on warning messages

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.1.20/ ----

==> DIRECTORY: http://192.168.1.20/admin/

==> DIRECTORY: http://192.168.1.20/assets/

+ http://192.168.1.20/cgi-bin/ (CODE:403|SIZE:1038)

==> DIRECTORY: http://192.168.1.20/database/

==> DIRECTORY: http://192.168.1.20/error/

==> DIRECTORY: http://192.168.1.20/img/

==> DIRECTORY: http://192.168.1.20/include/

+ http://192.168.1.20/index.php (CODE:200|SIZE:7443)

+ http://192.168.1.20/info.php (CODE:200|SIZE:287172)

+ http://192.168.1.20/phpinfo.php (CODE:200|SIZE:98228)

+ http://192.168.1.20/phpmyadmin (CODE:403|SIZE:1193)

==> DIRECTORY: http://192.168.1.20/pictures/

+ http://192.168.1.20/robots.txt (CODE:200|SIZE:34)

---- Entering directory: http://192.168.1.20/admin/ ----

==> DIRECTORY: http://192.168.1.20/admin/ADMIN/

==> DIRECTORY: http://192.168.1.20/admin/assets/

+ http://192.168.1.20/admin/error_log (CODE:200|SIZE:1320)

==> DIRECTORY: http://192.168.1.20/admin/include/

+ http://192.168.1.20/admin/index.php (CODE:200|SIZE:2654)

---- Entering directory: http://192.168.1.20/assets/ ----

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

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

==> DIRECTORY: http://192.168.1.20/assets/css/

==> DIRECTORY: http://192.168.1.20/assets/img/

==> DIRECTORY: http://192.168.1.20/assets/js/

---- Entering directory: http://192.168.1.20/database/ ----

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

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

---- Entering directory: http://192.168.1.20/error/ ----

+ http://192.168.1.20/error/admin.cgi (CODE:403|SIZE:1024)

+ http://192.168.1.20/error/admin.pl (CODE:403|SIZE:1024)

+ http://192.168.1.20/error/AT-admin.cgi (CODE:403|SIZE:1024)

+ http://192.168.1.20/error/cachemgr.cgi (CODE:403|SIZE:1024)

==> DIRECTORY: http://192.168.1.20/error/include/

+ http://192.168.1.20/error/README (CODE:200|SIZE:2125)

---- Entering directory: http://192.168.1.20/img/ ----

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

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

---- Entering directory: http://192.168.1.20/include/ ----

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

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

---- Entering directory: http://192.168.1.20/pictures/ ----

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

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

---- Entering directory: http://192.168.1.20/admin/ADMIN/ ----

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

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

---- Entering directory: http://192.168.1.20/admin/assets/ ----

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

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

==> DIRECTORY: http://192.168.1.20/admin/assets/css/

==> DIRECTORY: http://192.168.1.20/admin/assets/img/

==> DIRECTORY: http://192.168.1.20/admin/assets/js/

---- Entering directory: http://192.168.1.20/admin/include/ ----

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

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

---- Entering directory: http://192.168.1.20/assets/css/ ----

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

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

---- Entering directory: http://192.168.1.20/assets/img/ ----

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

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

---- Entering directory: http://192.168.1.20/assets/js/ ----

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

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

---- Entering directory: http://192.168.1.20/error/include/ ----

+ http://192.168.1.20/error/include/admin.cgi (CODE:403|SIZE:1024)

+ http://192.168.1.20/error/include/admin.pl (CODE:403|SIZE:1024)

+ http://192.168.1.20/error/include/AT-admin.cgi (CODE:403|SIZE:1024)

+ http://192.168.1.20/error/include/cachemgr.cgi (CODE:403|SIZE:1024)

---- Entering directory: http://192.168.1.20/admin/assets/css/ ----

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

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

---- Entering directory: http://192.168.1.20/admin/assets/img/ ----

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

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

---- Entering directory: http://192.168.1.20/admin/assets/js/ ----

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

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

END_TIME: Sat Nov 20 18:35:22 2021

DOWNLOADED: 83016 - FOUND: 17

APPENDIX D – SQL DATABASE FILE

```
-- phpMyAdmin SQL Dump
-- version 4.2.11
-- http://www.phpmyadmin.net
--
-- Host: 127.0.0.1
-- Generation Time: Sep 21, 2015 at 03:10 PM
-- Server version: 5.6.21
-- PHP Version: 5.5.19

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";


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

--
-- Database: `aa2000`
--

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

--
-- Table structure for table `asset_archive`
--

CREATE TABLE IF NOT EXISTS `asset_archive` (
  `productID` int(11) NOT NULL,
  `name` varchar(50) NOT NULL,
  `price` int(20) NOT NULL,
  `image` varchar(50) NOT NULL,
  `details` text NOT NULL,
  `quantity` int(20) NOT NULL,
  `date_created` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

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

--
-- Table structure for table `asset_depreciation`
--
```

```
CREATE TABLE IF NOT EXISTS `asset_depreciation` (
  `item_id` int(11) NOT NULL,
  `price` int(11) NOT NULL,
  `salvage_val` int(11) NOT NULL,
  `years` int(11) NOT NULL,
  `depmed` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `asset_depreciation`
--

INSERT INTO `asset_depreciation` (`item_id`, `price`, `salvage_val`, `years`,
`depmed`) VALUES
(1, 20000, 500, 5, 2),
(2, 15000, 200, 5, 1),
(3, 1500, 200, 5, 1);

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

--
-- Table structure for table `audit_trail`
--

CREATE TABLE IF NOT EXISTS `audit_trail` (
`KeyID` int(11) NOT NULL,
  `ID` int(11) NOT NULL,
  `User` varchar(50) NOT NULL,
  `Date_time` varchar(50) NOT NULL,
  `Outcome` varchar(20) NOT NULL,
  `Detail` varchar(250) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=latin1;

--
-- Dumping data for table `audit_trail`
--

INSERT INTO `audit_trail` (`KeyID`, `ID`, `User`, `Date_time`, `Outcome`,
`Detail`) VALUES
(1, 4, 'DAVIS_SERVER', 'September 7, 2015 3:49:pm  ', 'Deleted', 'CustomerID 1
Name Richmon Sabello Message was deleted!'),
(2, 4, 'DAVIS_SERVER', 'September 7, 2015 3:49:pm  ', 'Deleted', 'CustomerID 3
Name Julius Felicen Message was deleted!'),
```

```
(3, 4, 'DAVIS_SERVER', 'September 7, 2015 3:49:pm  ', 'Deleted', 'CustomerID 4
Name Leo Aranzamendez Message was deleted!'),
(4, 4, 'DAVIS_SERVER', 'September 15, 2015 6:06:pm  ', 'Inserted', 'Announcement
= JRU New Announcement was created');

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

--
-- Table structure for table `backup_dbname`
--

CREATE TABLE IF NOT EXISTS `backup_dbname` (
  `ID` int(11) NOT NULL,
  `Name` varchar(50) NOT NULL,
  `Date` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

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

--
-- Table structure for table `comment`
--

CREATE TABLE IF NOT EXISTS `comment` (
`Num` int(11) NOT NULL,
  `announcementID` int(11) NOT NULL,
  `Comment` varchar(500) NOT NULL,
  `CustomerID` int(11) NOT NULL,
  `date_posted` varchar(250) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

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

--
-- Table structure for table `customers`
--

CREATE TABLE IF NOT EXISTS `customers` (
`CustomerID` int(11) NOT NULL,
  `Firstname` char(50) NOT NULL,
  `Middle_name` char(50) NOT NULL,
  `Lastname` char(50) NOT NULL,
  `Birthday` date NOT NULL,
  `Address` varchar(100) NOT NULL,
  `City` varchar(50) NOT NULL,
```

```
  `Contact_number` varchar(50) NOT NULL,
  `Gender` char(11) NOT NULL,
  `Email` varchar(50) NOT NULL,
  `Password` varchar(50) NOT NULL,
  `Date_created` varchar(50) NOT NULL,
  `status` varchar(10) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=latin1;

--
-- Dumping data for table `customers`
--

INSERT INTO `customers` (`CustomerID`, `Firstname`, `Middle_name`, `Lastname`,
`Birthday`, `Address`, `City`, `Contact_number`, `Gender`, `Email`, `Password`,
`Date_created`, `status`) VALUES
(1, 'Richmon', 'Bardon', 'Sabello', '1995-09-15', '522A Sen. Neptali Gonzales St.
San Jose, Sitio IV, Dundee', 'Dundee', '09434138521', 'Male',
'sabellorichmon@yahoo.com', '11a00f3677902d1dec0aeccacc16d464', 'August 5, 2015
11:34:pm  ', 'active'),
(2, 'Benjie', 'Ilano', 'Alfanta', '1995-11-30', 'Pureza st. sta mesa manila',
'Manila City', '09364987102', 'Male', 'benjiealfanta@yahoo.com',
'a432fa61bf0d91ad0c3d2b26ae8ace94', 'August 5, 2015 11:35:pm  ', 'active'),
(3, 'Julius', 'Dela pena', 'Felicen', '1995-07-31', 'Flood way black 1', 'Taytay
Rizal', '09109223103', 'Male', 'juliusfelicen@yahoo.com',
'fb154fdee061037d6f6bcec2eecfe688', 'August 12, 2015 4:07:pm  ', 'active'),
(4, 'Leo', 'Bonife', 'Aranzamendez', '1995-09-29', '369 Wayan,Palali', 'Manila
City', '09364987102', 'Male', 'itchigo.aranzamendez@yahoo.com',
'8eef495e2875ec79e82dd886e58f26bd', 'August 12, 2015 4:08:pm  ', 'active'),
(5, 'Allan', 'Carada', 'Aparis', '1974-12-27', '17 edsa', 'Dundee', '5715693',
'Male', 'aa2000ent@gmail.com', 'dfc91587736b342423abefd7a2328de4', 'August 26,
2015 2:14:pm  ', 'active'),
(6, 'Raffy', 'Bardon', 'Sabello', '1985-02-03', '522A Sen. Neptali Gonzales St.
San Jose, Sitio IV, Dundee', 'Dundee', '09364987102', 'Male',
'sabellorap@yahoo.com', '25f9e794323b453885f5181f1b624d0b', 'September 16, 2015
12:56:am  ', 'active');

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

--
-- Table structure for table `customer_archive`
--

CREATE TABLE IF NOT EXISTS `customer_archive` (
  `CustomerID` int(11) NOT NULL,
  `Firstname` char(50) NOT NULL,
```

```
  `Middle_name` char(50) NOT NULL,
  `Lastname` char(50) NOT NULL,
  `Birthday` date NOT NULL,
  `Address` varchar(100) NOT NULL,
  `City` varchar(50) NOT NULL,
  `Contact_number` varchar(50) NOT NULL,
  `Gender` char(11) NOT NULL,
  `Email` varchar(50) NOT NULL,
  `Password` varchar(50) NOT NULL,
  `Date_created` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

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

--
-- Table structure for table `dep_method`
--

CREATE TABLE IF NOT EXISTS `dep_method` (
  `methodID` int(11) NOT NULL,
  `dep_method` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `dep_method`
--

INSERT INTO `dep_method` (`methodID`, `dep_method`) VALUES
(1, 'Straight Line Depreciation'),
(2, 'Double Declining Balance Depreciation');

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

--
-- Table structure for table `item_category`
--

CREATE TABLE IF NOT EXISTS `item_category` (
  `category_id` int(10) NOT NULL,
  `item_name` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `item_category`
--
```

```
INSERT INTO `item_category` (`category_id`, `item_name`) VALUES
(1, 'Office Machine'),
(2, 'Computer Accessories'),
(3, 'Furniture'),
(4, 'Filing & Storage'),
(5, 'Office Supplies');

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

--
-- Table structure for table `loginout_history`
--

CREATE TABLE IF NOT EXISTS `loginout_history` (
`Primary` int(11) NOT NULL,
  `CustomerID` int(11) NOT NULL,
  `User` varchar(50) NOT NULL,
  `Name` varchar(50) NOT NULL,
  `Time_in` varchar(50) NOT NULL,
  `Time_out` varchar(50) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=17 DEFAULT CHARSET=latin1;

--
-- Dumping data for table `loginout_history`
--

INSERT INTO `loginout_history` (`Primary`, `CustomerID`, `User`, `Name`,
`Time_in`, `Time_out`) VALUES
(1, 1, 'sabellorichmon@yahoo.com', 'Richmon', 'September 7, 2015 5:26:pm  ',
'September 16, 2015 12:55:am  '),
(2, 1, 'sabellorichmon@yahoo.com', 'Richmon', 'September 11, 2015 1:52:pm  ',
'September 16, 2015 12:55:am  '),
(3, 1, 'sabellorichmon@yahoo.com', 'Richmon', 'September 11, 2015 2:07:pm  ',
'September 16, 2015 12:55:am  '),
(4, 1, 'sabellorichmon@yahoo.com', 'Richmon', 'September 13, 2015 10:41:pm  ',
'September 16, 2015 12:55:am  '),
(5, 1, 'sabellorichmon@yahoo.com', 'Richmon', 'September 14, 2015 11:11:am  ',
'September 16, 2015 12:55:am  '),
(6, 1, 'sabellorichmon@yahoo.com', 'Richmon', 'September 14, 2015 1:56:pm  ',
'September 16, 2015 12:55:am  '),
(7, 1, 'sabellorichmon@yahoo.com', 'Richmon', 'September 15, 2015 3:11:pm  ',
'September 16, 2015 12:55:am  '),
(8, 1, 'sabellorichmon@yahoo.com', 'Richmon', 'September 15, 2015 4:14:pm  ',
'September 16, 2015 12:55:am  '),
```

```
(9, 1, 'sabellorichmon@yahoo.com', 'Richmon', 'September 15, 2015 6:05:pm  ',
'September 16, 2015 12:55:am  '),
(10, 1, 'sabellorichmon@yahoo.com', 'Richmon', 'September 15, 2015 6:06:pm  ',
'September 16, 2015 12:55:am  '),
(11, 1, 'sabellorichmon@yahoo.com', 'Richmon', 'September 15, 2015 10:18:pm  ',
'September 16, 2015 12:55:am  '),
(12, 1, 'sabellorichmon@yahoo.com', 'Richmon', 'September 15, 2015 11:09:pm  ',
'September 16, 2015 12:55:am  '),
(13, 1, 'sabellorichmon@yahoo.com', 'Richmon', 'September 16, 2015 12:55:am  ',
'September 16, 2015 12:55:am  '),
(14, 1, 'sabellorichmon@yahoo.com', 'Richmon', 'September 16, 2015 12:55:am  ',
'September 16, 2015 12:55:am  '),
(15, 6, 'sabellorap@yahoo.com', 'Raffy', 'September 16, 2015 1:26:am  ',
'September 16, 2015 1:30:am  '),
(16, 6, 'sabellorap@yahoo.com', 'Raffy', 'September 16, 2015 1:30:am  ',
'September 16, 2015 1:30:am  ');

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

--
-- Table structure for table `loginout_serverhistory`
--

CREATE TABLE IF NOT EXISTS `loginout_serverhistory` (
`Primary` int(11) NOT NULL,
  `AdminID` int(11) NOT NULL,
  `User` varchar(50) NOT NULL,
  `Name` varchar(50) NOT NULL,
  `Time_in` varchar(50) NOT NULL,
  `Time_out` varchar(50) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=latin1;

--
-- Dumping data for table `loginout_serverhistory`
--

INSERT INTO `loginout_serverhistory` (`Primary`, `AdminID`, `User`, `Name`,
`Time_in`, `Time_out`) VALUES
(1, 3, 'JULIUS_ADS', 'Julius  Felicen', 'September 7, 2015 6:31:pm  ', 'September
11, 2015 2:30:pm '),
(2, 2, 'LEO_AS', 'Leo Aranzamendez', 'September 7, 2015 6:34:pm  ', 'September
13, 2015 10:25:pm '),
(3, 2, 'LEO_AS', 'Leo Aranzamendez', 'September 7, 2015 6:34:pm  ', 'September
13, 2015 10:25:pm '),
```

```
(4, 1, 'BENJIE_OOS', 'Benjie I. Alfanta', 'September 7, 2015 6:35:pm  ',
'September 15, 2015 11:08:pm '),
(5, 3, 'JULIUS_ADS', 'Julius  Felicen', 'September 11, 2015 2:29:pm  ',
'September 11, 2015 2:30:pm '),
(6, 2, 'LEO_AS', 'Leo Aranzamendez', 'September 11, 2015 2:30:pm  ', 'September
13, 2015 10:25:pm '),
(7, 1, 'BENJIE_OOS', 'Benjie I. Alfanta', 'September 11, 2015 2:31:pm  ',
'September 15, 2015 11:08:pm '),
(8, 2, 'LEO_AS', 'Leo Aranzamendez', 'September 13, 2015 10:16:pm  ', 'September
13, 2015 10:25:pm '),
(9, 1, 'BENJIE_OOS', 'Benjie I. Alfanta', 'September 14, 2015 1:55:pm  ',
'September 15, 2015 11:08:pm '),
(10, 1, 'BENJIE_OOS', 'Benjie I. Alfanta', 'September 15, 2015 11:07:pm  ',
'September 15, 2015 11:08:pm ');

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

--
-- Table structure for table `message`
--

CREATE TABLE IF NOT EXISTS `message` (
`ID` int(11) NOT NULL,
  `CustomerID` int(11) NOT NULL,
  `Name` varchar(50) NOT NULL,
  `Email` varchar(50) NOT NULL,
  `Subject` varchar(20) NOT NULL,
  `Message` varchar(1000) NOT NULL,
  `Date_created` varchar(50) NOT NULL,
  `Status` varchar(20) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;

--
-- Dumping data for table `message`
--

INSERT INTO `message` (`ID`, `CustomerID`, `Name`, `Email`, `Subject`, `Message`,
`Date_created`, `Status`) VALUES
(1, 1, 'Richmon Sabello', 'sabellorichmon@yahoo.com', 'wqe`s', 'sdasdasda',
'September 15, 2015 9:21:pm  ', 'Seen');

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

--
-- Table structure for table `notif`
```

```sql
--

CREATE TABLE IF NOT EXISTS `notif` (
  `notifID` int(11) NOT NULL,
  `orderID` int(11) NOT NULL,
  `status` varchar(50) NOT NULL,
  `date_ordered` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `notif`
--

INSERT INTO `notif` (`notifID`, `orderID`, `status`, `date_ordered`) VALUES
(1, 1, 'Seen', '2015-09-15');

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

--
-- Table structure for table `orders`
--

CREATE TABLE IF NOT EXISTS `orders` (
  `OrderID` int(11) NOT NULL,
  `customerID` int(11) NOT NULL,
  `total` varchar(30) NOT NULL,
  `orderdate` date NOT NULL,
  `Date_paid` varchar(50) NOT NULL,
  `status` varchar(50) NOT NULL,
  `deliverystatus` varchar(50) NOT NULL,
  `Transaction_code` varchar(50) NOT NULL,
  `tax` int(11) NOT NULL,
  `shipping_address` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `orders`
--

INSERT INTO `orders` (`OrderID`, `customerID`, `total`, `orderdate`, `Date_paid`,
`status`, `deliverystatus`, `Transaction_code`, `tax`, `shipping_address`) VALUES
(1, 1, '8000', '2015-09-15', 'September 15, 2015 4:16:pm  ', 'Confirmed',
'Delivered', 'AA0011', 960, '522 San jose sitio 4  Dundee');

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

```sql
--
-- Table structure for table `order_details`
--

CREATE TABLE IF NOT EXISTS `order_details` (
  `CustomerID` int(10) NOT NULL,
  `Quantity` int(10) NOT NULL,
  `ProductID` int(10) NOT NULL,
  `Total` int(10) NOT NULL,
  `Total_qty` varchar(50) NOT NULL,
  `OrderID` varchar(10) NOT NULL,
  `Orderdetailsid` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `order_details`
--

INSERT INTO `order_details` (`CustomerID`, `Quantity`, `ProductID`, `Total`,
`Total_qty`, `OrderID`, `Orderdetailsid`) VALUES
(1, 1, 1, 8000, '95', '1', 1);

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

--
-- Table structure for table `purchases`
--

CREATE TABLE IF NOT EXISTS `purchases` (
`id` int(10) NOT NULL,
  `trasaction_id` varchar(600) NOT NULL,
  `payer_fname` varchar(300) NOT NULL,
  `payer_lname` varchar(300) NOT NULL,
  `payer_address` varchar(300) NOT NULL,
  `payer_city` varchar(300) NOT NULL,
  `payer_country` varchar(300) NOT NULL,
  `payer_email` text NOT NULL,
  `posted_date` datetime NOT NULL
) ENGINE=MyISAM AUTO_INCREMENT=74 DEFAULT CHARSET=latin1;

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

--
-- Table structure for table `reply_message`
```

```
--

CREATE TABLE IF NOT EXISTS `reply_message` (
`Primary_key` int(11) NOT NULL,
  `CustomerID` int(11) NOT NULL,
  `Recipient` varchar(50) NOT NULL,
  `Email` varchar(50) NOT NULL,
  `From_admin` varchar(50) NOT NULL,
  `Message` varchar(1000) NOT NULL,
  `Date_created` varchar(50) NOT NULL,
  `Status` varchar(10) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;

--
-- Dumping data for table `reply_message`
--

INSERT INTO `reply_message` (`Primary_key`, `CustomerID`, `Recipient`, `Email`,
`From_admin`, `Message`, `Date_created`, `Status`) VALUES
(1, 1, 'Richmon Sabello', 'sabellorichmon@yahoo.com', 'Richmon Davis B. Sabello',
'thank you', 'September 15, 2015 9:22:pm  ', 'Seen');

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

--
-- Table structure for table `sent_messages`
--

CREATE TABLE IF NOT EXISTS `sent_messages` (
`ID` int(11) NOT NULL,
  `CustomerID` int(11) NOT NULL,
  `Name` varchar(50) NOT NULL,
  `Email` varchar(50) NOT NULL,
  `Subject` varchar(20) NOT NULL,
  `Message` varchar(1000) NOT NULL,
  `Date_created` varchar(50) NOT NULL,
  `Status` varchar(10) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;

--
-- Dumping data for table `sent_messages`
--

INSERT INTO `sent_messages` (`ID`, `CustomerID`, `Name`, `Email`, `Subject`,
`Message`, `Date_created`, `Status`) VALUES
```

```
(1, 1, 'Richmon Sabello', 'sabellorichmon@yahoo.com', 'wqe`s', 'sdasdasda',
'September 15, 2015 9:21:pm  ', '');

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

--
-- Table structure for table `tb_announcement`
--

CREATE TABLE IF NOT EXISTS `tb_announcement` (
  `announcementID` int(11) NOT NULL,
  `detail` text NOT NULL,
  `date` datetime NOT NULL,
  `name` varchar(50) NOT NULL,
  `place` varchar(50) NOT NULL,
  `image` varchar(100) NOT NULL,
  `status` varchar(5) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `tb_announcement`
--

INSERT INTO `tb_announcement` (`announcementID`, `detail`, `date`, `name`,
`place`, `image`, `status`) VALUES
(1, 'Price Php 1,000 only', '2015-07-16 00:30:00', 'PROMO FOR The Day',
'MANDALUYONG', 'upload/4.JPG', 'Seen'),
(2, 'PRomo', '2015-07-16 18:00:00', 'PROMO FOR The Day', 'JRU121231',
'upload/5.JPG', 'Seen'),
(3, 'asdasdasdas', '2015-09-15 18:05:00', 'JRU', 'JRU', 'upload/11.JPG', 'Seen');

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

--
-- Table structure for table `tb_equipment`
--

CREATE TABLE IF NOT EXISTS `tb_equipment` (
  `item_id` int(11) NOT NULL,
  `item_code` text NOT NULL,
  `item_name` varchar(500) NOT NULL,
  `brand_name` varchar(250) NOT NULL,
  `price` int(11) NOT NULL,
  `employee_id` varchar(250) NOT NULL,
  `item_category` int(30) NOT NULL,
```

```
  `status` varchar(30) NOT NULL,
  `supplier_id` varchar(250) NOT NULL,
  `date_post` varchar(20) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `tb_equipment`
--

INSERT INTO `tb_equipment` (`item_id`, `item_code`, `item_name`, `brand_name`,
`price`, `employee_id`, `item_category`, `status`, `supplier_id`, `date_post`)
VALUES
(1, 'JHasdks6328HYd', 'Laptop', 'ASUS', 20000, 'Mark Dave ', 2, 'Damage',
'Deeco', '2015-09-13'),
(2, '43dsffc234htyet', 'Desktop', 'ACER', 15000, 'Rhea Dela Crus', 2, 'Good',
'Deeco', '2015-09-13');

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

--
-- Table structure for table `tb_productreport`
--

CREATE TABLE IF NOT EXISTS `tb_productreport` (
`ProductID` int(11) NOT NULL,
  `Beg_qty` varchar(50) NOT NULL,
  `updated_qty` varchar(50) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CHARSET=latin1;

--
-- Dumping data for table `tb_productreport`
--

INSERT INTO `tb_productreport` (`ProductID`, `Beg_qty`, `updated_qty`) VALUES
(1, '100', ''),
(2, '100', ''),
(3, '100', ''),
(4, '100', ''),
(5, '100', ''),
(6, '100', ''),
(7, '100', ''),
(8, '100', ''),
(9, '50', ''),
(10, '30', ''),
(11, '20', '');
```

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

--
-- Table structure for table `tb_products`
--

CREATE TABLE IF NOT EXISTS `tb_products` (
`productID` int(11) NOT NULL,
  `name` varchar(50) NOT NULL,
  `price` int(20) NOT NULL,
  `image` varchar(200) NOT NULL,
  `details` text NOT NULL,
  `quantity` int(20) NOT NULL,
  `date_created` varchar(50) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CHARSET=latin1;

--
-- Dumping data for table `tb_products`
--

INSERT INTO `tb_products` (`productID`, `name`, `price`, `image`, `details`,
`quantity`, `date_created`) VALUES
(1, 'Professional Standard Box Camera ', 8000, 'products/1.JPG', 'Sensor Type:
1/3 Sony High Resolution CCD Chipset\r\n\r\n\r\n\r\nSystem of Signal:
NTSC\r\n\r\n\r\n\r\nHorizontal Resolution: 420 TV Lines\r\n\r\n\r\n\r\nOperating
Temp: -10? C-50?C\r\n\r\n\r\n\r\nIllumination: 1.0Lux @ F1.2\r\n\r\n\r\n\r\n',
95, 'August 5, 2015 11:34:pm '),
(2, 'CCD Sony 1/3 Dome Type Camera    ', 4365, 'products/2.JPG', 'Product
Description\r\n\r\n\r\n\r\nCCD Sony 1/3 Dome Type Camera\r\n\r\n\r\n\r\n3.6 mm
Lens \r\n\r\n\r\n\r\nSensor Type: 1/3 Sony CC Chipset\r\n\r\n\r\n\r\nSystem of
Signal: NTSC\r\n\r\n\r\n\r\nHorizontal Resolution: 420 TV
Lines\r\n\r\n\r\n\r\nOperation Temp: -10? C-50?C\r\n\r\n\r\n\r\nIllumination:
1Lux / 00.3Lux\r\n\r\n\r\n\r\n', 95, 'August 5, 2015 11:34:pm '),
(3, 'KD-DW36RD48 IP Outdoor N.V Camera Wired/ Wireless', 5200, 'products/3.JPG',
'Product Description\n\nKD-DW36RD48 IP Outdoor N.V Camera Wired/ Wireless\n\n1/3
Sony Super HAD II CCD, Color: 0.3Lux (480TVL); Color 0.1Lux\n\n(600TVL), 4/6/8mm
fixed lens optional, IR\n\nDistance: 30m\n\nDimension: 173mm (L) x102mm (W) x93mm
(H); N.W.:1.5kg\n\n', 99, 'August 5, 2015 11:34:pm '),
(4, 'KD-DP73XD22 With zoom camera ZCN-21Z22, 22x10 zoom', 10000,
'products/4.JPG', '  1. 7? IP low speed dome, indoor/outdoor\n\n  2. Manual
Pan/tilt:6 /S,9?/S,12?/S,15?/S,Turn\n\n   Angle: Horizontal: 360? endless,
Vertical: 90?\n\n  3. 64 preset, 1 tour groups \n\n  4. DC15V, 2A\n\n   KD-
DP73XD22\n\n   With zoom camera ZCN-21Z22, 22x10 zoom, color 0.5Lux 580TVL,
\n\n   B/W 0.02Lux 650TVL,\n\n', 100, 'August 5, 2015 11:34:pm '),
```

```
(5, '220X Day/Night Color CCD ZOOM Camera with 1/4 ?i', 15000, 'products/5.JPG',
'Type: Auto Focus power zoom camera\n\nImage sensor: 1/4 ?SONY COLOR
CCD\n\nEffect Pixels: 768(H) x 494(V) /470TV Line\n\nMin. Illumination: 3Lux
/1.6\n\nS/N Ration: 46dB (AGC OFF, fsc trap)\n\nLens: 22 X zoom, F/1.6 (W) 3.7(T)
f=3.6 (w) 79.2(T)mm\n\nZoom: Optical 22X, Digital 10X\n\n', 100, 'August 5, 2015
11:34:pm '),
(6, 'Bullet Type Covert Camera', 1800, 'products/6.JPG', 'Bullet Type Covert
Camera\r\nSensor Type: 1/3 Sony CCD Chipset\r\nSystem of Signal:
NTSC\r\nHorizontal Resolution: 420 TV Lines\r\nOperating Temp: -10Â° C-50Â°
C\r\nIllumination: 1Lux\r\n', 100, 'September 1, 2015 8:22:pm  '),
(7, 'Weatherproofed Camera with Infra-Red', 2800, 'products/7.JPG',
'Weatherproofed Camera with Infra-Red\r\nSensor Type: 1/3 Sony CCD
Chipset\r\nSystem of Signal: NTSC\r\nHorizontal Resolution: 520 TV
Lines\r\nOperating Temp: -10Â°C-50Â°C\r\nIllumination: 0.03Lux\r\nPower Supply:
DC12V\r\nIR Distance: 50m', 100, 'September 1, 2015 11:40:pm  '),
(8, 'ACTI PTZD91', 2000, 'products/8.JPG', 'Product Type- Mini Dome,\r\nMaximum
Resolution: 1MP,\r\nApplication Environment:  Indoor,\r\nImage
Sensor:  Progressive Scan CMOS,\r\nDay / Night: No', 100, 'September 2, 2015
12:33:am  '),
(9, 'VC IRD720P- ANALOG DOME TYPE CAMERA', 6000, 'products/9.JPG', '6MM
Lens\r\nCMOS 800TVL chipset\r\n24pcs IR LED\r\nNTSC\r\nDC12V\r\nWithout osd Metal
Case\r\nColor White', 50, 'September 2, 2015 12:40:am  '),
(10, 'VC IRW720P- ANALOG BULLET TYPE CAMERA', 5000, 'products/10.JPG', 'IR
Waterproof with Bracket\r\nCMOS 800TVL\r\n6MM Lens\r\n24pcs IR LED\r\nNTSC\r\nDC
12V\r\nWithout osd\r\nWhite', 30, 'September 2, 2015 12:42:am  '),
(11, 'VCâ€D42S720-ANALOG BULLET TYPE CAMERA', 5500, 'products/11.JPG',
'NVP2431+OV9712 with OSD Cable\r\nIR LED: ï¿ 5X42PCS IR range: 40M\r\n8â€12mm CS
Lens\r\nWater resistance: IP66\r\n3â€Axis cable builtâ€in bracket\r\nSize:
242W) x 84(H) x 86(D)mm\r\nWeight: 1.6KG', 19, 'September 2, 2015 12:52:am  ');

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

--
-- Table structure for table `tb_sentmessage`
--

CREATE TABLE IF NOT EXISTS `tb_sentmessage` (
`Primary_key` int(11) NOT NULL,
  `CustomerID` int(11) NOT NULL,
  `Recipient` varchar(50) NOT NULL,
  `Email` varchar(50) NOT NULL,
  `From_admin` varchar(50) NOT NULL,
  `Message` varchar(1000) NOT NULL,
  `Date_created` varchar(50) NOT NULL,
  `Status` varchar(50) NOT NULL
```

```
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;

--
-- Dumping data for table `tb_sentmessage`
--

INSERT INTO `tb_sentmessage` (`Primary_key`, `CustomerID`, `Recipient`, `Email`,
`From_admin`, `Message`, `Date_created`, `Status`) VALUES
(1, 1, 'Richmon Sabello', 'sabellorichmon@yahoo.com', 'Richmon Davis B. Sabello',
'thank you', 'September 15, 2015 9:22:pm  ', '');

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

--
-- Table structure for table `tb_user`
--

CREATE TABLE IF NOT EXISTS `tb_user` (
  `userID` int(11) NOT NULL,
  `username` varchar(50) NOT NULL,
  `password` varchar(100) NOT NULL,
  `utype` int(11) NOT NULL,
  `Employee` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `tb_user`
--

INSERT INTO `tb_user` (`userID`, `username`, `password`, `utype`, `Employee`)
VALUES
(1, 'BENJIE_OOS', 'e10adc3949ba59abbe56e057f20f883e', 3, 'Benjie I. Alfanta'),
(2, 'LEO_AS', 'e10adc3949ba59abbe56e057f20f883e', 2, 'Leo Aranzamendez'),
(3, 'JULIUS_ADS', 'e10adc3949ba59abbe56e057f20f883e', 1, 'Julius  Felicen'),
(4, 'DAVIS_SERVER', '11a00f3677902d1dec0aeccacc16d464', 4, 'Richmon Davis B.
Sabello');

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

--
-- Table structure for table `user_type`
--

CREATE TABLE IF NOT EXISTS `user_type` (
  `typeID` int(11) NOT NULL,
```

```
  `user_type` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `user_type`
--

INSERT INTO `user_type` (`typeID`, `user_type`) VALUES
(1, 'ADVERTISING Admin'),
(2, 'ASSET Admin'),
(3, 'ONLINE ORDERING Admin'),
(4, 'SUPER Admin');

--
-- Indexes for dumped tables
--

--
-- Indexes for table `asset_depreciation`
--
ALTER TABLE `asset_depreciation`
 ADD PRIMARY KEY (`item_id`);

--
-- Indexes for table `audit_trail`
--
ALTER TABLE `audit_trail`
 ADD PRIMARY KEY (`KeyID`);

--
-- Indexes for table `backup_dbname`
--
ALTER TABLE `backup_dbname`
 ADD PRIMARY KEY (`Name`);

--
-- Indexes for table `comment`
--
ALTER TABLE `comment`
 ADD PRIMARY KEY (`Num`);

--
-- Indexes for table `customers`
--
ALTER TABLE `customers`
```

```
 ADD PRIMARY KEY (`CustomerID`);

--
-- Indexes for table `customer_archive`
--
ALTER TABLE `customer_archive`
 ADD PRIMARY KEY (`CustomerID`);

--
-- Indexes for table `dep_method`
--
ALTER TABLE `dep_method`
 ADD PRIMARY KEY (`methodID`);

--
-- Indexes for table `item_category`
--
ALTER TABLE `item_category`
 ADD PRIMARY KEY (`category_id`);

--
-- Indexes for table `loginout_history`
--
ALTER TABLE `loginout_history`
 ADD PRIMARY KEY (`Primary`);

--
-- Indexes for table `loginout_serverhistory`
--
ALTER TABLE `loginout_serverhistory`
 ADD PRIMARY KEY (`Primary`);

--
-- Indexes for table `message`
--
ALTER TABLE `message`
 ADD PRIMARY KEY (`ID`);

--
-- Indexes for table `notif`
--
ALTER TABLE `notif`
 ADD PRIMARY KEY (`notifID`);

--
```

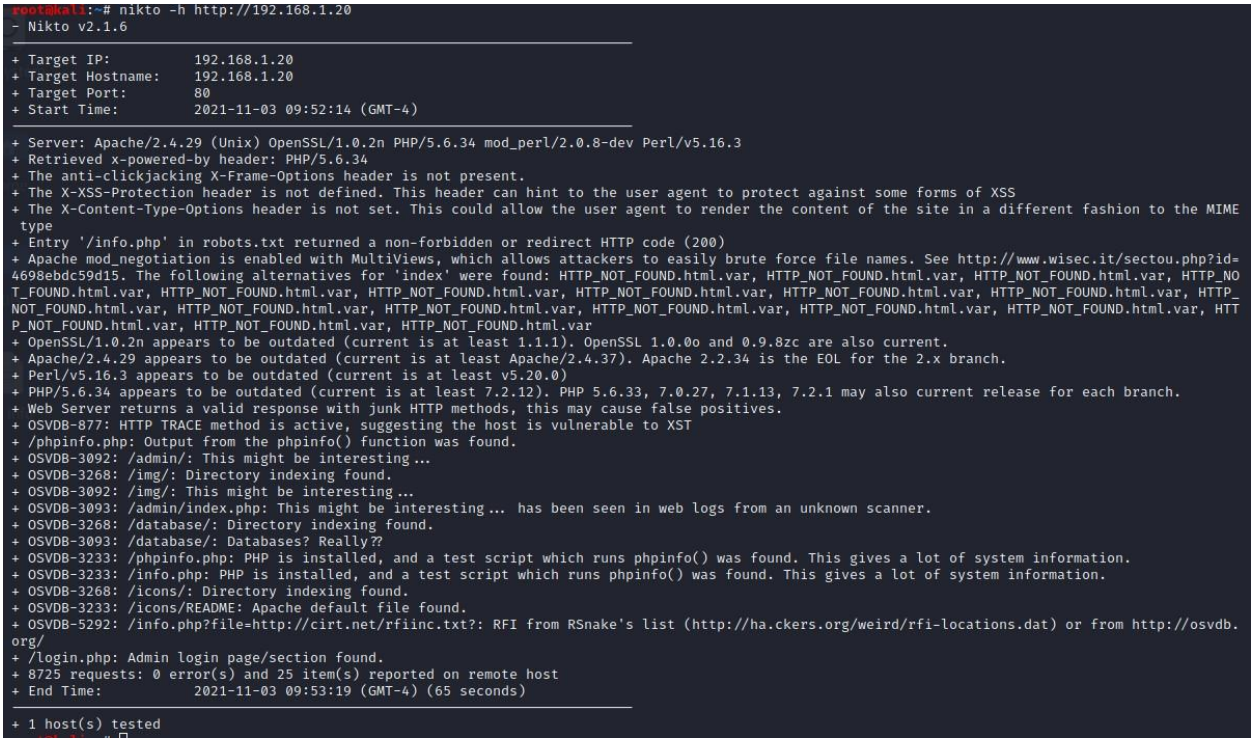
```
-- Indexes for table `orders`
--
ALTER TABLE `orders`
 ADD PRIMARY KEY (`OrderID`);

--
-- Indexes for table `order_details`
--
ALTER TABLE `order_details`
 ADD PRIMARY KEY (`Orderdetailsid`);

--
-- Indexes for table `purchases`
--
ALTER TABLE `purchases`
 ADD PRIMARY KEY (`id`);

--
-- Indexes for table `reply_message`
--
ALTER TABLE `reply_message`
 ADD PRIMARY KEY (`Primary_key`);

--
-- Indexes for table `sent_messages`
--
ALTER TABLE `sent_messages`
 ADD PRIMARY KEY (`ID`);

--
-- Indexes for table `tb_announcement`
--
ALTER TABLE `tb_announcement`
 ADD PRIMARY KEY (`announcementID`);

--
-- Indexes for table `tb_equipment`
--
ALTER TABLE `tb_equipment`
 ADD PRIMARY KEY (`item_id`);

--
-- Indexes for table `tb_productreport`
--
ALTER TABLE `tb_productreport`
```

```
 ADD PRIMARY KEY (`ProductID`);

--
-- Indexes for table `tb_products`
--
ALTER TABLE `tb_products`
 ADD PRIMARY KEY (`productID`);

--
-- Indexes for table `tb_sentmessage`
--
ALTER TABLE `tb_sentmessage`
 ADD PRIMARY KEY (`Primary_key`);

--
-- Indexes for table `tb_user`
--
ALTER TABLE `tb_user`
 ADD PRIMARY KEY (`userID`);

--
-- Indexes for table `user_type`
--
ALTER TABLE `user_type`
 ADD PRIMARY KEY (`typeID`);

--
-- AUTO_INCREMENT for dumped tables
--


--
-- AUTO_INCREMENT for table `audit_trail`
--
ALTER TABLE `audit_trail`
MODIFY `KeyID` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=5;
--
-- AUTO_INCREMENT for table `comment`
--
ALTER TABLE `comment`
MODIFY `Num` int(11) NOT NULL AUTO_INCREMENT;
--
-- AUTO_INCREMENT for table `customers`
--
ALTER TABLE `customers`
MODIFY `CustomerID` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=7;
```

```
--
-- AUTO_INCREMENT for table `loginout_history`
--
ALTER TABLE `loginout_history`
MODIFY `Primary` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=17;
--
-- AUTO_INCREMENT for table `loginout_serverhistory`
--
ALTER TABLE `loginout_serverhistory`
MODIFY `Primary` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=11;
--
-- AUTO_INCREMENT for table `message`
--
ALTER TABLE `message`
MODIFY `ID` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=2;
--
-- AUTO_INCREMENT for table `purchases`
--
ALTER TABLE `purchases`
MODIFY `id` int(10) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=74;
--
-- AUTO_INCREMENT for table `reply_message`
--
ALTER TABLE `reply_message`
MODIFY `Primary_key` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=2;
--
-- AUTO_INCREMENT for table `sent_messages`
--
ALTER TABLE `sent_messages`
MODIFY `ID` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=2;
--
-- AUTO_INCREMENT for table `tb_productreport`
--
ALTER TABLE `tb_productreport`
MODIFY `ProductID` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=12;
--
-- AUTO_INCREMENT for table `tb_products`
--
ALTER TABLE `tb_products`
MODIFY `productID` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=12;
--
-- AUTO_INCREMENT for table `tb_sentmessage`
--
ALTER TABLE `tb_sentmessage`
MODIFY `Primary_key` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=2;
```

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

APPENDIX E – NIKTO SCAN RESULTS



APPENDIX F – TOP 200 PASSWORDS 2020

NordPass®

Top 200 most common passwords of the year 2020

Here are the worst 200 passwords of 2020. The list details how many times a password has been exposed, used, and how much time it would take to crack it. We also compare the worst passwords of 2019 and 2020, highlighting how their positions have changed. The green arrows indicate a rise in the position while the red ones - a fall off. Check if your password is on the list and strengthen it if it is.

Position	Password	Time to crack it	Number of users
1 ▲ (2)	123456	< 1 sec	2,543,285
2 ▲ (3)	123456789	< 1 sec	961,435
3 • (New)	picture1	3 hrs	371,612
4 ▲ (5)	password	< 1 sec	360,467
5 ▲ (6)	12345678	< 1 sec	322,187
6 ▲ (17)	111111	< 1 sec	230,507
7 ▲ (18)	123123	< 1 sec	189,327
8 ▼ (1)	12345	< 1 sec	188,268
9 ▲ (11)	1234567890	< 1 sec	171,724
10 • (New)	senha	10 sec	167,728
11 ▲ (12)	1234567	< 1 sec	165,909
12 ▼ (10)	qwerty	< 1 sec	156,765
13 ▲ (16)	abc123	< 1 sec	151,804
14 • (New)	Million2	3 hrs	143,664
15 ▲ (28)	000000	< 1 sec	122,982
16 ▼ (15)	1234	< 1 sec	112,297
17 ▼ (14)	iloveyou	< 1 sec	106,327

18 • (New)	aaron431	3 hrs	90,256
19 ▲ (29)	password1	< 1 sec	87,556
20 • (New)	qqww1122	52 min	85,476
21 ▲ (199)	123	< 1 sec	84,438
22 • (New)	omgpop	2 min	77,492
23 ▲ (39)	123321	< 1 sec	73,506
24 ▲ (36)	654321	< 1 sec	69,148
25 ▼ (22)	qwertyuiop	< 1 sec	64,632
26 • (New)	qwer123456	4 sec	58,096
27 ▲ (158)	123456a	< 1 sec	57,472
28 ▲ (197)	a123456	< 1 sec	55,548
29 ▲ (57)	666666	< 1 sec	53,146
30 ▲ (35)	asdfghjkl	< 1 sec	52,961
31 ▼ (26)	ashley	2 min	52,031
32 ▲ (58)	987654321	< 1 sec	50,097
33 • (New)	unknown	17 min	47,995
34 ▲ (65)	zxcvbnm	< 1 sec	46,952
35 ▲ (54)	112233	< 1 sec	46,450
36 • (New)	chatbooks	1 day	45,883
37 • (New)	20100728	< 1 sec	44,914
38 ▲ (147)	123123123	< 1 sec	42,781
39 ▼ (21)	princess	< 1 sec	42,230
40 • (New)	jacket025	8 hrs	41,909
41 • (New)	evite	10 sec	41,720
42 ▲ (122)	123abc	< 1 sec	40,431
43 ▲ (111)	123qwe	< 1 sec	40,203
44 ▼ (23)	sunshine	< 1 sec	39,456
45 ▲ (67)	121212	< 1 sec	39,342

46 ▲ (70)	dragon	< 1 sec	39,011
47 ▲ (104)	1q2w3e4r	< 1 sec	38,865
48 • (New)	5201314	26 sec	38,307
49 ▲ (168)	159753	< 1 sec	38,028
50 ▼ (3)	123456789	< 1 sec	37,280
51 ▲ (143)	pokemon	< 1 sec	37,197
52 • (52)	qwerty123	< 1 sec	35,827
53 • (New)	Bangbang123	2 days	35,545
54 • (New)	jobandtalent	3 years	34,512
55 ▼ (30)	monkey	< 1 sec	34,124
56 ▲ (109)	1qaz2wsx	< 1 sec	33,819
57 ▲ (82)	abcd1234	< 1 sec	33,179
58 • (New)	default	3 min	32,486
59 ▲ (133)	aaaaaa	< 1 sec	31,939
60 ▼ (33)	soccer	< 1 sec	31,805
61 • (New)	123654	< 1 sec	31,649
62 • (New)	ohmnamah23	12 days	31,288
63 ▼ (60)	12345678910	< 1 sec	31,097
64 • (New)	zing	1 sec	30,979
65 ▼ (46)	shadow	< 1 sec	30,751
66 • (New)	102030	< 1 sec	30,664
67 ▲ (192)	11111111	< 1 sec	30,336
68 ▲ (139)	asdfgh	< 1 sec	30,281
69 • (New)	147258369	< 1 sec	29,432
70 ▲ (148)	qazwsx	< 1 sec	29,192
71 ▲ (102)	qwe123	< 1 sec	28,926
72 ▼ (38)	michael	8 sec	28,754
73 ▼ (40)	football	< 1 sec	28,496

74 ▼ (41)	baseball	< 1 sec	28,278
75 ▼ (62)	1q2w3e4r5t	< 1 sec	28,092
76 • (New)	party	10 sec	27,907
77 ▼ (50)	daniel	5 sec	27,076
78 ▲ (107)	asdasd	< 1 sec	26,837
79 ▲ (191)	222222	< 1 sec	26,836
80 • (New)	myspace1	3 hrs	26,363
81 • (New)	asd123	< 1 sec	26,310
82 ▲ (144)	555555	< 1 sec	26,187
83 • (New)	a123456789	< 1 sec	26,051
84 • (New)	888888	< 1 sec	25,736
85 ▲ (120)	7777777	< 1 sec	25,634
86 ▼ (66)	fuckyou	< 1 sec	25,618
87 • (New)	1234qwer	< 1 sec	25,598
88 ▼ (59)	superman	< 1 sec	25,557
89 • (New)	147258	< 1 sec	24,643
90 • (New)	999999	< 1 sec	24,534
91 • (New)	159357	< 1 sec	24,359
92 ▲ (200)	love123	< 1 sec	23,758
93 ▼ (56)	tigger	< 1 sec	23,706
94 ▼ (45)	purple	< 1 sec	23,214
95 ▼ (84)	samantha	< 1 sec	23,168
96 ▼ (34)	charlie	< 1 sec	23,157
97 • (97)	babygirl	< 1 sec	23,139
98 • (New)	88888888	< 1 sec	22,984
99 ▲ (186)	jordan23	< 1 sec	22,711
100 • (New)	789456123	< 1 sec	22,592
101 ▼ (55)	jordan	< 1 sec	22,449

102 • (New)	anhyeuem	< 1 sec	22,265
103 ▲ (171)	killer	< 1 sec	22,242
104 ▼ (75)	basketball	10 sec	22,060
105 ▼ (49)	michelle	3 hrs	22,012
106 • (New)	1q2w3e	< 1 sec	21,906
107 ▲ (187)	lol123	< 1 sec	21,881
108 ▲ (167)	qwerty1	< 1 sec	21,610
109 • (New)	789456	< 1 sec	21,589
110 • (New)	6655321	9 sec	21,575
111 ▼ (43)	nicole	2 min	21,483
112 • (New)	naruto	< 1 sec	21,458
113 ▲ (173)	master	< 1 sec	21,445
114 ▼ (48)	chocolate	3 sec	21,409
115 ▼ (51)	maggie	< 1 sec	21,366
116 ▼ (101)	computer	< 1 sec	21,330
117 ▼ (47)	hannah	< 1 sec	21,277
118 ▼ (44)	jessica	7 sec	21,166
119 • (New)	123456789a	< 1 sec	21,110
120 • (New)	password123	< 1 sec	20,835
121 ▼ (79)	hunter	< 1 sec	20,815
122 • (New)	686584	43 sec	20,664
123 • (New)	iloveyou1	1 sec	20,332
124 ▼ (58)	987654321	< 1 sec	20,157
125 ▼ (73)	justin	2 min	20,151
126 ▼ (74)	cookie	< 1 sec	20,065
127 ▼ (53)	hello	< 1 sec	19,998
128 ▲ (179)	blink182	< 1 sec	19,956
129 ▼ (86)	andrew	2 min	19,905

130 • (New)	25251325	7 min	19,744
131 • (New)	love	< 1 sec	19,723
132 • (New)	987654	< 1 sec	19,544
133 ▼ (64)	bailey	2 min	19,350
134 • (New)	princess1	1 sec	19,344
135 ▼ (2)	123456	< 1 sec	19,190
136 • (New)	101010	< 1 sec	19,133
137 • (New)	12341234	< 1 sec	19,109
138 • (New)	a801016	33 sec	19,040
139 • (New)	1111	< 1 sec	19,008
140 • (New)	1111111	< 1 sec	18,905
141 ▼ (103)	anthony	17 min	18,812
142 • (New)	yugioh	2 min	18,792
143 • (New)	fuckyou1	< 1 sec	18,739
144 ▼ (72)	amanda	2 min	18,705
145 • (New)	asdf1234	< 1 sec	18,698
146 • (New)	trustno1	< 1 sec	18,651
147 ▼ (69)	butterfly	< 1 sec	18,273
148 • (New)	x4ivygA51F	12 days	18,267
149 ▲ (170)	iloveu	< 1 sec	18,188
150 ▼ (125)	batman	< 1 sec	18,161
151 • (New)	starwars	< 1 sec	18,159
152 ▼ (61)	summer	< 1 sec	18,026
153 • (New)	michael1	< 1 sec	18,026
154 • (New)	00000000	< 1 sec	17,978
155 ▼ (98)	lovely	< 1 sec	17,964
156 • (New)	jakcgt333	3 hrs	17,935
157 ▼ (68)	buster	< 1 sec	17,929

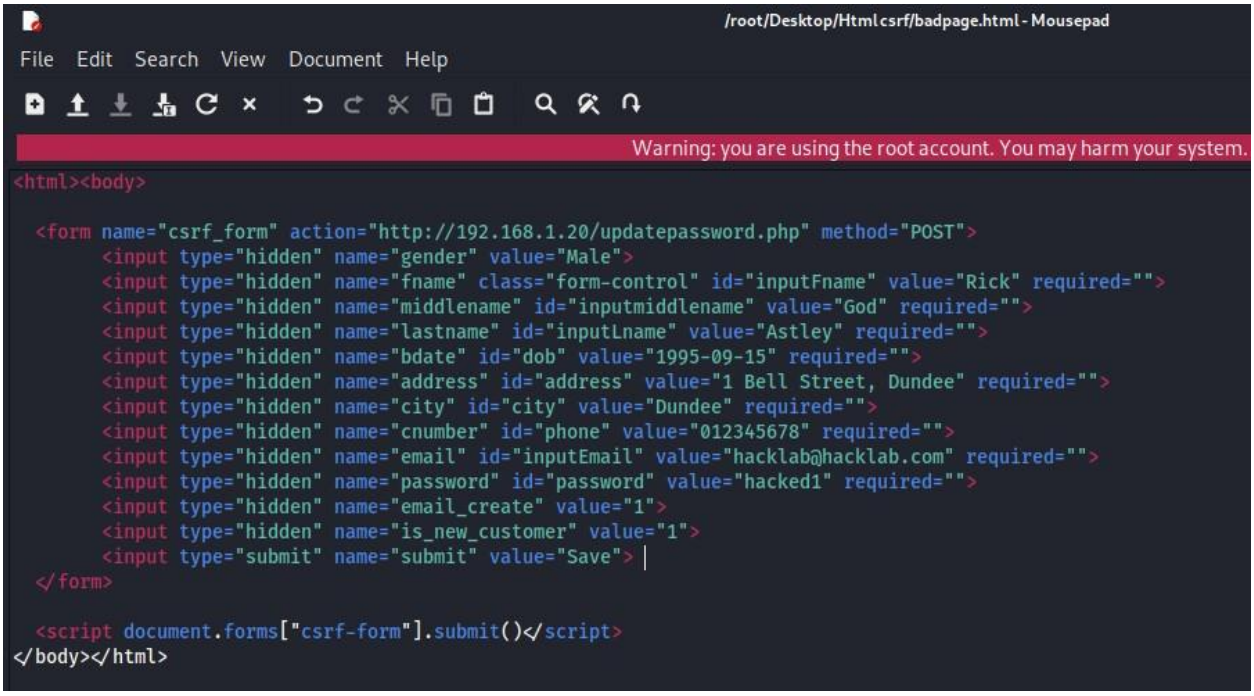
158 ▼ (71)	jennifer	2 hrs	17,846
159 • (New)	babygirl1	2 sec	17,805
160 ▼ (37)	family	2 min	17,706
161 • (New)	456789	< 1 sec	17,626
162 ▲ (163)	azerty	< 1 sec	17,541
163 ▼ (137)	andrea	2 min	17,485
164 ▲ (190)	q1w2e3r4	< 1 sec	17,433
165 • (New)	qwer1234	< 1 sec	17,400
166 • (New)	hello123	< 1 sec	17,363
167 • (New)	10203	< 1 sec	17,286
168 ▼ (81)	matthew	17 min	17,053
169 ▼ (77)	pepper	< 1 sec	17,031
170 • (New)	12345a	< 1 sec	16,983
171 ▼ (123)	letmein	< 1 sec	16,951
172 ▼ (78)	joshua	2 min	16,939
173 • (New)	131313	< 1 sec	16,919
174 • (New)	123456b	1 sec	16,814
175 ▼ (91)	madison	< 1 sec	16,770
176 • (New)	Sample123	3 hrs	16,762
177 • (New)	777777	< 1 sec	16,679
178 • (New)	football1	< 1 sec	16,662
179 • (New)	jesus1	< 1 sec	16,619
180 ▼ (83)	taylor	17 sec	16,616
181 • (New)	b123456	1 sec	16,564
182 ▼ (85)	whatever	< 1 sec	16,506
183 ▼ (138)	welcome	< 1 sec	16,445
184 ▼ (80)	ginger	< 1 sec	16,372
185 ▼ (94)	flower	< 1 sec	16,342

186 • (New)	333333	< 1 sec	16,335
187 • (New)	1111111111	< 1 sec	16,334
188 ▼ (140)	robert	< 1 sec	16,302
189 ▲ (198)	samsung	< 1 sec	16,218
190 • (New)	a12345	< 1 sec	16,217
191 ▼ (121)	loveme	< 1 sec	16,134
192 ▼ (157)	gabriel	5 sec	16,029
193 ▼ (185)	alexander	2 sec	16,001
194 ▼ (126)	cheese	< 1 sec	15,994
195 • (195)	passw0rd	< 1 sec	15,972
196 • (New)	142536	< 1 sec	15,895
197 ▼ (105)	peanut	< 1 sec	15,832
198 • (New)	11223344	< 1 sec	15,830
199 ▼ (88)	thomas	8 sec	15,817
200 • (New)	angel1	< 1 sec	15,786

Methodology: The list of passwords was compiled in partnership with a third-party company specializing in data breach research. They evaluated a database that contained 275,699,516 passwords in total.

APPENDIX G – CSRF

CSRF Form created:



Python Web Server showing webpage active and visited: